Version October 31, 2025
Preprint typeset using LaTeX style openjournal v. 09/06/15

THE RAY TRACING SAMPLER: BAYESIAN SAMPLING OF NEURAL NETWORKS FOR EVERYONE

Peter Behroozi[1]
[1]Department of Astronomy and Steward Observatory, University of Arizona, Tucson, AZ 85721, USA
Version October 31, 2025

ABSTRACT

We derive a Markov Chain Monte Carlo sampler based on following ray paths in a medium where the refractive index $n(\mathbf{x})$ is a function of the desired likelihood $\mathcal{L}(\mathbf{x})$. The sampling method propagates rays at constant speed through parameter space, leading to orders of magnitude higher resilience to heating for stochastic gradients as compared to Hamiltonian Monte Carlo (HMC), as well as the ability to cross any likelihood barrier, including holes in parameter space. Using the ray tracing method, we sample the posterior distributions of neural network outputs for a variety of different architectures, up to the 1.5 billion-parameter GPT-2 (Generative Pre-trained Transformer 2) architecture, all on a single consumer-level GPU. We also show that prior samplers including traditional HMC, microcanonical HMC, Metropolis, Gibbs, and even Monte Carlo integration are special cases within a generalized ray tracing framework, which can sample according to an arbitrary weighting function. Public code and documentation for C, `JAX`, and `PyTorch` are available at https://bitbucket.org/pbehroozi/ray-tracing-sampler/src.

Subject headings: statistics – sampling algorithms

1. INTRODUCTION

Sampling algorithms, including MCMC methods, aim to draw independent samples from a probability distribution $P(\mathbf{x})$, often only using an unnormalized likelihood function $\mathcal{L}(\mathbf{x}) \propto P(\mathbf{x})$. This problem arises in all scientific disciplines, as it is equivalent to finding the distribution of model parameters $\mathbf{x}$ consistent with a set of observations. Early sampling algorithms, such as Metropolis et al. (1953) and Hastings (1970), are hence widely used across fields.

Since these early papers, there have been many advances. Sampling algorithms differ in their transition functions, often denoted as $\pi(\mathbf{x} :\rightarrow \mathbf{y})$ or $T(\mathbf{y}|\mathbf{x})$, defined as the probability density of moving to point $\mathbf{y}$ when starting at point $\mathbf{x}$. Sampling efficiency depends on how well the transition function aligns with the target distribution $P(\mathbf{x})$, so modern samplers have transition functions that adapt to the target distribution. We can broadly classify modern algorithms by whether they use local or global information to adapt the transition function. Local samplers use information near the current location, such as likelihood gradients (e.g., Hamiltonian/Hybrid Monte Carlo, Duane et al. 1987; Neal 2012; the Metropolis-Adjusted Langevin Algorithm, Besag 1994; Roberts & Tweedie 1996; and recently, Bouncy Particle and Zig-Zag Sampling, Peters & de With 2012; Bouchard-Côté et al. 2015; Bierkens et al. 2016; Corbella et al. 2022). Global samplers instead learn approximate fitting functions to the global target distribution (e.g., mode jumping, Tjelmeland & Hegstad 2001; nested sampling, Skilling 2004; Feroz & Hobson 2008; Feroz et al. 2019; Speagle 2020; KDE sampling, Rosenblatt 1956; Parzen 1962; Gaussian processes, Benavoli et al. 2021; and normalizing flows, Tabak & Vanden-Eijnden 2010; Dinh et al. 2014; Jimenez Rezende & Mohamed 2015). Hybrid methods that combine global and local information (e.g., flowMC, Wong et al. 2023) also exist.

Yet, the complexity of scientific modeling has grown exponentially in the meantime. It is now common to train neural networks with millions to trillions of parameters in every scientific field (e.g., computer science, physics, biology, chemistry, and geoscience; Ahmad et al. 2022; Choudhary et al. 2022; Chowdhery et al. 2022; Puzyrev et al. 2022; Sriram et al. 2022; Lin et al. 2023; OpenAI 2023). In astronomy, this includes problems like initial condition recovery (e.g., reconstructing the initial matter distribution that led to the current Milky Way galaxy and its local environment, for billions to trillions of particles; see Vogelsberger et al. 2020 for a review) and blended source recovery (e.g., reconstructing intrinsic colors and luminosities for billions of partially overlapping galaxies as imaged by the *Rubin* telescope; see Melchior et al. 2021 for a review). All of these applications would benefit from Bayesian sampling to obtain uncertainties on model outputs, but it has remained challenging with current methods to do so.

At present, with $O(D^{1/4})$ scaling in the number of dimensions D, Hamiltonian Monte Carlo (HMC) techniques remain the leading methods for high-dimensional exploration (Neal 2012). HMC techniques follow the trajectory of a massive particle in a gravitational potential given by $U(\mathbf{x}) = -\ln \mathcal{L}(\mathbf{x})$, typically with a specific kinetic energy of $T = \frac{1}{2}|\mathbf{v}|^2$ (defining $\mathbf{v} \equiv \dot{\mathbf{x}}$), thus leading to an acceleration of $\dot{\mathbf{v}} = -\nabla U$. While HMC and related techniques can work well for neural networks with exact likelihood gradients (Izmailov et al. 2021), using stochastic gradients results in path heating that must be counteracted by an additional cooling term (Chen et al. 2014). Because the correct cooling term can be a complex function of the likelihood, it can be difficult to implement stochastic HMC in practice for Bayesian sampling of neural networks.

In this paper, we derive a simple and physically motivated alternative, which is to integrate the paths of light rays through a space where the refractive index varies as $n(\mathbf{x}) = \mathcal{L}(\mathbf{x})^{1/(D-1)}$, which we show to result in fair sampling. We implement light ray propagation with a constant speed through parame-

ter space, which leads to very high resilience to heating from stochastic gradients. This offers enormous performance advantages compared to traditional HMC, allowing us to perform approximate Bayesian sampling of neural networks on consumer hardware.

With the ray tracing method, parameter space trajectories can cross arbitrary likelihood barriers, similar to microcanonical HMC (Robnik et al. 2022). With simple extensions, the method can also cross arbitrary disallowed regions in parameter space, which is not possible with past HMC-like methods. The algorithm admits a simple Metropolis-like acceptance criterion, so that exact sampling can be achieved even with imperfect integrators.

There are deep connections between Hamiltonian mechanics and Hamiltonian optics (which underpins the ray tracing method)—essentially, Hamiltonian optics has much the same mathematics, but has only the direction of propagation as a free parameter instead of the full velocity. These connections mean that traditional HMC and microcanonical HMC, along with many other past approaches, can be rewritten in terms of ray tracing samplers using appropriate weighting and/or velocities, providing a unified theoretical framework for interpreting local sampling methods and their relative efficiency at sampling high-likelihood peaks vs. exploring low-likelihood tails.

In this paper, we introduce the ray tracing method and connect it to past approaches in Section 2, show example applications to Gaussian distributions and deep neural networks in Section 3, and provide discussion and conclusions in Sections 4 and 5, respectively.

2. METHODS

2.1. *Overview*

Here, we provide intuition for how ray tracing can sample fairly from likelihood distributions, with the formal mathematical derivation provided in Sections 2.2–2.7.

When light passes through the interface between two media, it is refracted according to Snell's law:

$$n \sin(\theta) = constant, \tag{1}$$

where n is the refractive index and θ is the angle with respect to the interface normal vector. This equation implies that light traveling from a lower-index medium to a higher-index medium will reorient to be closer to the interface normal vector. Second, the mathematical form of Snell's law results in light being focused from a larger range of input solid angles in the lower-index medium to a smaller range of output solid angles in the higher-index medium, causing an overall radiance increase within the higher-index medium.

If we create a space in which the refractive index $n(\mathbf{x})$ is an increasing function of the likelihood $\mathcal{L}(\mathbf{x})$, Snell's law provides behavior useful for sampling. For example, if we start from a low-likelihood (low-index) region, we would want our path to reorient closer to the likelihood (refractive index) gradient, to explore higher-likelihood (higher-index) regions. As well, we would want to sample higher-likelihood (higher-index) regions with higher probability. Both of these behaviors occur due to the refracting and focusing effects of Snell's law, as shown for example in Fig. 1.

As we show for arbitrary dimensionality D (Section 2.2), Snell's law conserves the *basic radiance* R of the system, defined as:

$$R \equiv n^{1-D} L = constant, \tag{2}$$

where L is the radiance (power per unit area per unit solid angle), which corresponds to the geometric density of ray paths, i.e., the sampling density. As a result, we can make an appropriate choice for the refractive index to enable fair sampling, which is

$$n(\mathbf{x}) = \mathcal{L}(\mathbf{x})^{\frac{1}{D-1}}. \tag{3}$$

With this choice, the radiance L will automatically be proportional to the likelihood function $\mathcal{L}(\mathbf{x})$, so the ray paths will fairly sample the parameter-space posterior distribution. Eq. 2 also provides a simple test of the detailed balance of the system—similar to the energy conservation Metropolis test for Hamiltonian Monte Carlo systems—so that exact sampling can be guaranteed even for imperfect integrators.

Finally, we briefly note that ergodicity is guaranteed as long as we introduce periodic scattering events that refresh the direction of travel, which results in ray diffusion throughout the parameter space. Provided that parameter space boundaries are modeled as blackbody radiators, light paths can even reach parameter space islands, which is very difficult for other Hamiltonian-like techniques.

In the next few sections, we derive the conservation of basic radiance and étendue for arbitrary dimensionality (Sections 2.2 and 2.3), show how this leads to a fair sampling algorithm in Section 2.4, prove ergodicity in 2.5, discuss continuous momentum refreshment in 2.6, generalize the sampler to arbitrary sample weights in Section 2.7, provide pseudocode in Section 2.8, connect the sampler to prior work in Section 2.9, discuss adapting the sampling distribution in Section 2.10, and provide specific recommendations for neural networks in Section 2.11.

2.2. *Conservation of basic radiance in D dimensions*

In Hamiltonian optics, the canonical approach for dealing with light transport is to consider the evolution of a *ray bundle*—that is, a group of light rays with differential cross section (dA) and differential solid angle ($d\Omega$). We adopt this approach here, as it simplifies the derivation of the Jacobian (i.e., the compression or expansion of space) of the transition function and hence shortens the proof of stationarity. In this section, we show that the basic radiance, $R = n^{1-D}L$, is conserved during refraction of ray bundles.

First, we consider a ray bundle (i.e., a collection of rays in a differential volume of position and direction phase space) crossing the interface between two transparent media with refractive indices n_1 and n_2. This is shown in Fig. 2 for $D > 1$. Within medium i, we define the angle with respect to the interface normal as θ_i, and the opening angle of the bundle along the direction to the interface normal vector as $d\theta_i$. We define $d\psi_1, \ldots, d\psi_{D-2}$ to be the remaining opening angles in hyperspherical coordinates, which have no subscript i as their values do not change across the interface. The differential powers for the incoming/outgoing ray bundles are $d^2P_i = L_i \cos(\theta_i) dA d\Omega_i$, where L_i is the incoming or outgoing radiance, $\cos(\theta_i)dA$ is the differential projected hyper-area that the bundle hits, and $d\Omega_i = \sin^{D-2}(\theta_i)d\theta_i \sin^{D-3}(\psi_1)d\psi_1 \sin^{D-4}(\psi_2)d\psi_2 \ldots d\psi_{D-2}$ is the differential hyper-solid angle.

Because light (or, in our case, probability) cannot be created or destroyed at the interface, we find that $d^2P_1 = d^2P_2$. Canceling like factors of $\sin^{D-3}(\psi_1)d\psi_1, \ldots, d\psi_{D-2}$, we find:

$$L_1 \cos(\theta_1) d\theta_1 \sin^{D-2}(\theta_1) = L_2 \cos(\theta_2) d\theta_2 \sin^{D-2}(\theta_2). \tag{4}$$

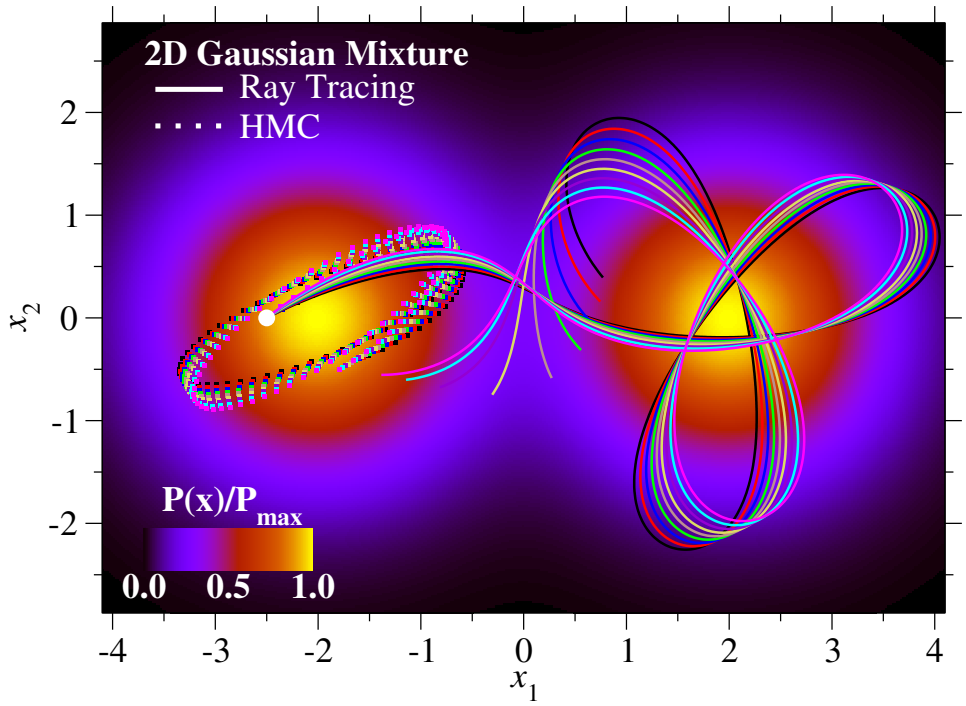

FIG. 1.— Example sampling dynamics for ray tracing and Hamiltonian Monte Carlo (HMC) for a 2D Gaussian mixture, with $\mathcal{L}(\mathbf{x}) = [\exp(-0.5(x_1-2)^2) + \exp(-0.5(x_1+2)^2)]\exp(-0.5x_2^2)$. For this case, as well as any other 2D distribution, fair sampling with ray tracing occurs when the spatially-varying refractive index is the same as the likelihood function (i.e., $n(\mathbf{x}) = \mathcal{L}(\mathbf{x})$). This figure illustrates the bending of light toward higher-probability regions due to Snell's law for a ray bundle originating from the white dot in the left of the figure. Notably, the rays can explore a wide range of likelihood values. In contrast, particle orbits starting from the same initial conditions using HMC are limited by the initial kinetic energy, chosen for this illustration to be the mean kinetic energy. The background color scale corresponds to the underlying probability distribution function, scaled by its maximum value.

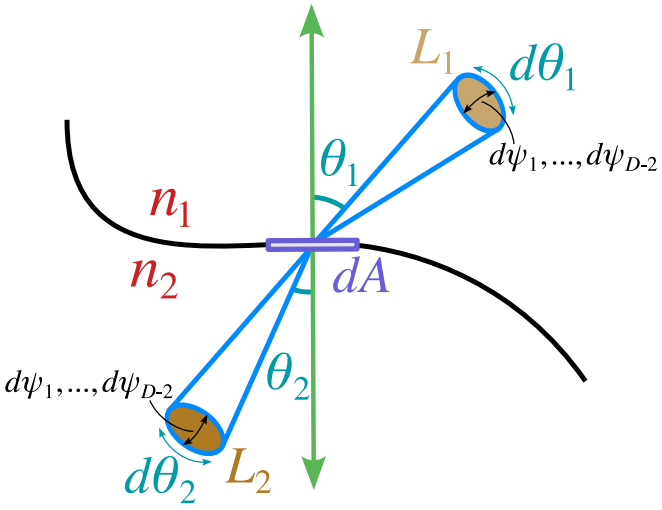

FIG. 2.— Refraction of light for an arbitrary dimensionality D. This figure shows an incoming beam of light with radiance L_1 in a medium with refractive index n_1, crossing a differential area dA at the interface with another medium with refractive index n_2, resulting in an outgoing beam of light with radiance L_2. By Snell's law, $n_1 \sin\theta_1 = n_2 \sin\theta_2$, where θ_1 and θ_2 are the respective angles to the interface normal vector (*in green*). If n_2 is larger than n_1, the outgoing beam of light is compressed not only in the $\hat{\theta}_2$ direction (by Snell's law), but also along the directions perpendicular to $\hat{\theta}_2$, similar to how lines of longitude are compressed as one moves closer to the Earth's poles. In addition, the outgoing beam becomes closer to the interface normal vector, increasing the projected area ($\cos\theta_2 dA$) from which it appears to originate. These two effects combine to result in an outgoing radiance L_2 that is a factor $(n_2/n_1)^{D-1}$ larger than the incoming radiance L_1 (Eq. 6).

By Snell's law, $n_1\sin(\theta_1) = n_2\sin(\theta_2)$, and differentiating with respect to θ_1 or θ_2 yields $n_1\cos(\theta_1)d\theta_1 = n_2\cos(\theta_2)d\theta_2$.

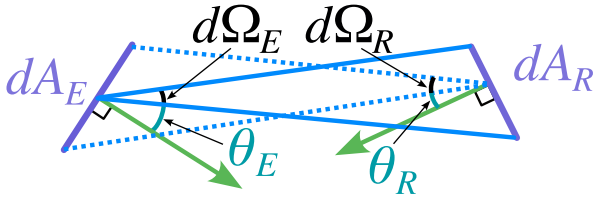

FIG. 3.— Parameters describing rays propagating from a differential emission area dA_E to a differential receiving area dA_R. The angles between the rays and the normal vectors of the emitter and receiver are θ_E and θ_R, respectively. The differential solid angle encompassed by the rays at the emitter is $d\Omega_E$, and that encompassed by the rays at the receiver is $d\Omega_R$. As discussed in the text, the étendue at the emitter ($d^2G_E \equiv n_E^{D-1} dA_E \cos\theta_E d\Omega_E$) is equal to the étendue at the receiver ($d^2G_R \equiv n_R^{D-1} dA_R \cos\theta_R d\Omega_R$).

This gives the following useful substitutions:

$$\frac{\sin(\theta_2)}{\sin(\theta_1)} = \frac{\cos(\theta_2)d\theta_2}{\cos(\theta_1)d\theta_1} = \frac{n_1}{n_2}. \tag{5}$$

As a result, Eq. 4 reduces to

$$\frac{L_1}{L_2} = \left(\frac{n_1}{n_2}\right)^{D-1}, \tag{6}$$

and we have proved the conservation of basic radiance for $D > 1$. For a one-dimensional space, there is no change in angle when moving from n_1 to n_2, so $L_1 = L_2$, and Eq. 6 also applies for $D = 1$.

In the derivation above, we neglected reflection. This is necessary to consider for abrupt transitions of the refractive index, but in this paper, we are concerned with smoothly-varying refractive indices. As one takes the limit of an increasing number k of layers that linearly interpolate between an initial refractive index n_i and final refractive index n_f, it is straightforward to

show that the Fresnel reflection coefficients all asymptote to 0—i.e., there are no reflections, as with graded-index lenses in three dimensions. Similarly, the radiance at the final layer will be:

$$L_f = L_i \prod_{j=1}^{k} \left(\frac{n_i + (j-1)(n_f - n_i)/k}{n_i + j(n_f - n_i)/k} \right)^{D-1}, \tag{7}$$

which reduces to Eq. 6 regardless of the number of layers. As a result, Eq. 6 holds for all smoothly-varying refractive indices in arbitrary dimensions D—i.e., the quantity Ln^{1-D} is invariant.

2.3. *Conservation of étendue in D dimensions*

In Hamiltonian mechanics, Liouville's theorem guarantees invariance of phase-space volume, and hence a unit Jacobian for transitions. In Hamiltonian optics, however, the Jacobian is *not* unity, and instead there is another conserved quantity called *étendue*, which is analogous to the phase-space volume of a ray bundle.

The derivation of Eq. 6 suggests that the following quantity will be conserved, which we will define as the étendue for D dimensions:

$$d^2G \equiv n^{D-1} dA \cos\theta d\Omega, \tag{8}$$

where d^2G refers to the scaled phase-space volume of a ray bundle (as in Fig. 2), encompassing rays within a solid angle $d\Omega$ that are crossing a surface of area dA at an angle θ to the surface normal, embedded in a medium with refractive index n. Indeed, recognizing that basic radiance is connected to étendue by $R = \frac{d^2P}{d^2G}$, basic radiance will be conserved if étendue is conserved along the ray path.

The standard conservation proof (in three dimensions) for the étendue d^2G generalizes very simply. Consider the rays that start on a differential area dA_E of an emitter and fall on a differential area dA_R of a receiver (Fig. 3) in a medium of uniform refractive index n. At the location of the emitter, the étendue will be $d^2G_E = n^{D-1} dA_E \cos\theta_E d\Omega_E$, where θ_E is the angle between the emitted rays and the surface normal, and $d\Omega_E$ is the differential solid angle of the rays that land on dA_R. Since the refractive index is uniform, light will travel in straight lines, and so $d\Omega_E$ will correspond to the projected area of the receiver divided by the area of a hypersphere with radius equal to the distance r between the source and the receiver:

$$d^2G_E = n^{D-1} dA_E \cos\theta_E dA_R \cos\theta_R H_D^{-1} r^{1-D}, \tag{9}$$

where θ_R is the angle between the surface normal of dA_R and the incoming rays, and $H_D r^{D-1}$ is the surface area of the hypersphere. Likewise, the étendue at the receiver will be $d^2G_R = n^{D-1} dA_R \cos\theta_R d\Omega_R$, where $d\Omega_R$ is the differential solid angle of the incoming rays originating from dA_E. As before, we relate $d\Omega_R$ to the projected area of the emitter as $d\Omega_R = dA_E \cos\theta_E H_D^{-1} r^{1-D}$, so

$$d^2G_R = n^{D-1} dA_R \cos\theta_R dA_E \cos\theta_E H_D^{-1} r^{1-D}, \tag{10}$$

and so $d^2G_E = d^2G_R$ by inspection. Intuitively, as a ray bundle propagates, it will spread to a larger projected area ($dA_R \cos\theta_R$), but the apparent solid angle subtended by the emitting source ($d\Omega_R$) will become smaller and smaller, so that the product d^2G remains constant.

Extending this to media with varying refractive indices is straightforward. Considering a ray bundle passing through the boundary between two different media (Fig. 2), the incoming étendue is $d^2G_1 = n_1^{D-1} dA \cos\theta_1 d\Omega_1$, and the outgoing étendue is $d^2G_2 = n_2^{D-1} dA \cos\theta_2 d\Omega_2$. We hence have:

$$\begin{aligned} \frac{d^2G_1}{d^2G_2} &= \left(\frac{n_1}{n_2}\right)^{D-1} \frac{\cos(\theta_1) dA d\Omega_1}{\cos(\theta_2) dA d\Omega_2} \\ &= \left(\frac{n_1}{n_2}\right)^{D-1} \frac{d^2P_1}{d^2P_2} \frac{L_2}{L_1} \\ &= 1, \end{aligned} \tag{11}$$

where $d^2P_1 = d^2P_2$ are the incoming and outgoing differential powers as defined above. Hence, étendue is conserved across media boundaries. To extend this to media with continuously varying refractive indices, we can again approximate light paths using a discretized medium—i.e., wherein the refractive index is constant over a small volume. Regardless of how finely we discretize, the étendue will not change, and hence in the limit of infinitesimally small discretization, étendue is conserved.

2.4. *The ray tracing sampler*

The conservation of generalized étendue implies that the phase-space volume of a ray bundle is compressed by a factor of n^{D-1} depending on the refractive index. We can use this to our advantage by linking the refractive index to the target likelihood distribution $\mathcal{L}(\mathbf{x})$, so that ray paths cluster in regions of higher likelihood. The most natural way to do so is:

$$n(\mathbf{x}) = \mathcal{L}(\mathbf{x})^{\frac{1}{D-1}}, \tag{12}$$

so that the ray bundle phase-space density (i.e., the radiance L) is exactly proportional to $\mathcal{L}(\mathbf{x})$. That is, the Jacobian of a ray tracing transition between two points under Eq. 12 exactly cancels out the difference in probability between the two points.

If this seems all too easy, we can leverage the results of the previous sections to show detailed balance quantitatively. Consider a differential volume $ds dA_{\mathbf{x}}$ located at a position $\mathbf{x}$, where ds is the differential distance along the ray path and $dA_{\mathbf{x}}$ is the differential cross-sectional area that is perpendicular to the direction of travel. Now suppose that we emit light rays from this volume in uniform directions with an intensity proportional to the likelihood $\mathcal{L}(\mathbf{x})$. Formally, the radiant intensity density (power per volume per solid angle) emitted by this volume will be:

$$\frac{d^3P_{\mathbf{x}}}{dA_{\mathbf{x}} ds d\Omega_{\mathbf{x}}} = C\mathcal{L}(\mathbf{x}), \tag{13}$$

where C is the constant of proportionality, which by definition does not depend on $\mathbf{x}$ or the solid angle $\Omega_{\mathbf{x}}$.

We next consider the radiant intensity density of ray bundles reaching a volume $ds dA_{\mathbf{y}}$ after traveling a total path length s (with $dA_{\mathbf{y}}$ again the cross-sectional area perpendicular to the direction of travel, so again $\cos\theta = 1$). This formulation explicitly preserves the extent along the direction of travel, so ds is invariant.[1] As the étendue d^2G is preserved, we find:

$$d^2G = n(\mathbf{x})^{D-1} dA_{\mathbf{x}} d\Omega_{\mathbf{x}} = n(\mathbf{y})^{D-1} dA_{\mathbf{y}} d\Omega_{\mathbf{y}}. \tag{14}$$

[1] To avoid potential confusion: as we are propagating the paths of ray bundles instead of photons, there is no explicit speed of light. If we were tracking individual photons, they would travel slower in regions with higher refractive indices, but then the distances between photons would decrease correspondingly, canceling the effect of the photon speed on the luminosity.

Hence,

$$\frac{d^3P_{\mathbf{x}}}{dA_{\mathbf{y}}dsd\Omega_{\mathbf{y}}} = C\frac{n(\mathbf{y})^{D-1}}{n(\mathbf{x})^{D-1}}\mathcal{L}(\mathbf{x}) = C\mathcal{L}(\mathbf{y}). \tag{15}$$

That is, the differential power reaching the volume around $\mathbf{y}$ is *non-uniformly* distributed in solid angle and cross-section in exactly the right way to balance the difference in likelihoods between the two points. Because light rays propagate along reversible paths, the differential power emitted uniformly at $\mathbf{y}$ that reaches the differential volume $dsdA_{\mathbf{x}}$ at $\mathbf{x}$ will be non-uniformly distributed in the exact opposite way, ensuring detailed balance.

All practical ray tracing algorithms will of course introduce path errors, such that the product of the phase-space density and the likelihood will not be perfectly conserved. Similar to HMC samplers, however, we can introduce an acceptance criterion based on the conservation of basic radiance. As typically implemented, ray tracing algorithms update the angle of propagation of the ray path relative to the likelihood gradient by an amount $\Delta\theta$ at each timestep t_i. According to Eqs. 4 and 5, the boost in radiance will then be:

$$\frac{L_{i+1}}{L_i} = \left(\frac{\sin(\theta)}{\sin(\theta+\Delta\theta)}\right)^{D-1}, \tag{16}$$

so the net radiance boost $L(\mathbf{y})/L(\mathbf{x})$ between the starting and ending positions can be calculated as the product of the individual boosts. This can then be compared to the actual change in likelihood, resulting in the following acceptance probability:

$$P(\mathbf{x}\to\mathbf{y}) = \min\left(1, \frac{\mathcal{L}(\mathbf{y})L(\mathbf{x})}{\mathcal{L}(\mathbf{x})L(\mathbf{y})}\right). \tag{17}$$

We note for completeness that ray tracing sampling is only meaningful in $D > 1$ dimensions, as there are no degrees of freedom for the direction of propagation (and hence sampling density) in $D = 1$, so Eq. 12 becomes undefined. In such cases, we recommend simply integrating the likelihood function over the single free parameter or using an alternate approach in Section 2.9.

2.5. *Ergodicity*

Ergodicity requires that the direction of travel be periodically refreshed, for example, after a randomly chosen path length s. This allows for full exploration of the direction space $\mathbf{v}$ at any given location $\mathbf{x}$. Now, suppose that the reachable positions $\mathbf{x}$ are a bounded set B. By definition, we can choose $\mathbf{x}$ to be a point at most an arbitrarily small distance ϵ away from the boundary of B. Ray tracing involves a one-to-one mapping of initial directions to final directions because it is a reversible transformation (even for discrete integration, as in Eq. 23 in Section 2.8). Hence, an initial isotropic distribution for $\mathbf{v}$ will be transformed into a non-isotropic distribution of outgoing $\mathbf{v}$; however, all directions will be represented, including directions perpendicular to the boundary of B. Hence, if the step size Δs is equal to $\epsilon + \delta$ for some appropriately small δ, then a single step of the ray tracing algorithm will reach a point outside the boundary B. Provided that this point has a nonzero likelihood, then it will be reachable by the algorithm, and so the boundary for B can only exist if the likelihood is identically zero at all points along the boundary—that is, if there are parameter space islands.

Parameter space islands are a special challenge for most Hamiltonian methods, since they cannot be reached by integrating the equations of motion. However, for ray tracing methods, we can treat parameter space boundaries as black-body radiators that absorb ray bundles from arbitrary directions and then re-radiate them in a new direction according to black-body emissivity (i.e., where the emission probability is proportional to the product of n^{D-1} and the cosine of the angle to the surface normal, also known as Lambertian emissivity). Treating the non-allowed regions of parameter space as a medium with a uniform refractive index, this allows ray bundles to propagate between parameter space boundaries and be re-radiated back into allowed regions. Noting that computing the full surface normal of a parameter space boundary may be prohibitive if the number of dimensions is large, Appendix A provides a simple algorithm for black-body emission that only requires knowing the location of the parameter space boundary along two randomly chosen orthogonal directions.

2.6. *Partial momentum refreshment, i.e., scattering*

Choosing a momentum refreshment rate involves a trade-off between exploring the parameter space along a fixed ray path (infrequent refreshes) vs. different ray paths (more frequent refreshes). For problems where the typical sets (i.e., the regions with typical values of $\ln\mathcal{L}$; see also Betancourt 2017) are long, thin tubes (e.g., neural networks), it can be advantageous to have long timescales between momentum refreshes. However, when momentum refreshes do occur, they can be disruptive to exploration rates and require short timesteps to avoid leaving the typical set. Performing many partial momentum refreshes, analogous to an optical scattering process, provides an appealing alternative because step sizes can be kept high even as the momentum becomes uncorrelated over large time scales.

We can treat scattering/partial momentum refresh as a random rotation of the ray direction, which ensures that the Jacobian is unchanged. However, some care is necessary for *when* the scattering is performed. It is best to perform scattering at locations that would in principle be on the typical set, i.e., in between completed integration steps. Performing scattering inside the integration step (e.g., drift–scatter–kick–scatter–drift), while allowed, may nonetheless lead to increased rejections, because the scattering may occur at locations outside of the typical set, and hence correspond physically to heating or cooling of the trajectories. As discussed in Neal (2012), if the direction of the momentum vector is not refreshed between Metropolis tests, it should be reversed upon Metropolis failure to maintain reversibility.

2.7. *Generalized ray tracing with sample weights*

The ray tracing method discussed so far has involved following paths at constant speed. We can generalize the method by allowing the speed to change along the path. Higher speeds result in a lower sampling density per unit integration time, so the implicit sample weight is related to the velocity as $W_i = v^{-1}$. Alternately, we could retain a constant path speed and simply have an explicit weight W applied to each sample. Neither the sample weighting nor the speed affects the conservation of basic radiance, which is instead a statement about the geometric density of ray bundles. Instead, variable sample weighting and speed allow the sampler to be tuned so that it spends more time in lower- or higher-likelihood regions, depending on user needs. As discussed in Section 2.9, we can

also use weighting to describe a large variety of past sampling methods within the context of ray tracing.

For an arbitrary explicit weight W and path speed v, the sample density will scale as the product of the radiance L and the weight W, divided by the speed v. Letting $L(\mathbf{x})W(\mathbf{x})/v(\mathbf{x}) = \mathcal{L}(\mathbf{x})$ to preserve fair sampling, this is equivalent to rescaling $\mathcal{L}(\mathbf{x})$ by $W(\mathbf{x})/v(\mathbf{x})$ and so we find that the refractive index is:

$$n(\mathbf{x}) = \left(\frac{\mathcal{L}(\mathbf{x})v(\mathbf{x})}{W(\mathbf{x})}\right)^{\frac{1}{D-1}}. \tag{18}$$

The probability conservation test simply generalizes from Eq. 2:

$$P(\mathbf{x} \to \mathbf{y}) = \min\left(1, \frac{n(\mathbf{x})^{1-D}L(\mathbf{x})}{n(\mathbf{y})^{1-D}L(\mathbf{y})}\right), \tag{19}$$

which corresponds to the inverse of the excess basic radiance at $\mathbf{y}$ compared to the starting point $\mathbf{x}$, and so exact sampling can be achieved for arbitrary weighting/velocity schemes.

2.8. *Algorithm pseudocode*

The standard form of Snell's law used for integrating ray paths is given in textbooks such as Born & Wolf (1980):

$$\frac{d}{ds}\left(n(\mathbf{x})\frac{d\mathbf{x}}{ds}\right) = \nabla_{\mathbf{x}} n(\mathbf{x}), \tag{20}$$

where s is the path length and the gradient has a subscript $\mathbf{x}$ to emphasize that it is taken with respect to the parameter space instead of the path. We can rewrite to solve for the change in the direction of travel, $\frac{d\mathbf{x}}{ds}$, as:

$$\frac{d}{ds}\left(\frac{d\mathbf{x}}{ds}\right) = \nabla_{\mathbf{x}} \ln(n(\mathbf{x})) - \left(\frac{d\mathbf{x}}{ds} \cdot \nabla_{\mathbf{x}} \ln(n(\mathbf{x}))\right)\frac{d\mathbf{x}}{ds}. \tag{21}$$

This formulation makes it clear that the dynamics depend only on the gradient of $\ln(n)$ (and hence only on the gradient of the log-likelihood), so that both the refractive index and the likelihood can be renormalized by a constant factor without affecting the dynamics. As well, in Eq. 21, the rate of change of direction depends only on the component of the log refractive index gradient that is perpendicular to the direction of travel. That is, the magnitude of the right-hand side is $\sin\theta|\nabla_{\mathbf{x}} \ln(n(\mathbf{x}))|$, with θ being the angle between the direction of travel and the refractive index gradient.

When integrating Eq. 21, some care is necessary to maintain the unit length of $\frac{d\mathbf{x}}{ds}$. Indeed, because $\frac{d\mathbf{x}}{ds}$ has unit length, we can interpret $\frac{d}{ds}\left(\frac{d\mathbf{x}}{ds}\right)$ as the rate of change in θ:

$$\frac{d\theta}{ds} = -\sin\theta|\nabla_{\mathbf{x}} \ln(n(\mathbf{x}))|. \tag{22}$$

This equation specifies how the refractive index influences the angle of propagation, and hence how the luminosity changes according to Eq. 16. For readers who are less familiar with Eq. 20, Eq. 22 can also be derived simply from the differential form of Snell's law ($\frac{dn}{ds}\sin\theta = -n\cos\theta\frac{d\theta}{ds}$), with the identity $\frac{dn}{ds} = \cos\theta|\nabla_{\mathbf{x}} n|$. The main subtlety of Eq. 22 is that the definition of θ changes during ray propagation. Hence, the correct approach is to treat Eq. 22 as an update to $\frac{d\mathbf{x}}{ds}$ (e.g., as a "kick" step of a numerical integration method), and then to use the resulting $\frac{d\mathbf{x}}{ds}$ to propagate the ray (e.g., as a "drift" step). To maintain reversibility, we can integrate Eq. 22 over the appropriate step length Δs:

$$\tan\left(\frac{\theta_f}{2}\right) = \tan\left(\frac{\theta_i}{2}\right)\exp(-\Delta s|\nabla_{\mathbf{x}} \ln(n(\mathbf{x}))|), \tag{23}$$

where θ_f is the final angle and θ_i is the initial angle. This equation is explicitly symmetric under a reversal of the propagation direction ($\theta \to \theta + \pi$) and the labels $\theta_f \leftrightarrow \theta_i$, as is simply shown from the identity $\tan((\theta+\pi)/2) = -\cot(\theta/2)$.

Algorithm 1 Generalized ray tracing sampler

procedure GRTSample($\mathbf{x}_0, D, N, \Delta s, f, \mathcal{L}(\mathbf{x})$, optional: $W(\mathbf{x}) := 1$)
- $\mathbf{v} \leftarrow$ random Gaussian in D dimensions
- $\Delta \ln L \leftarrow 0$
- $\Delta s \leftarrow \sqrt{D}\Delta s$ *Recommended—rescales step size with dimensionality*
- **for** $k \leftarrow 0, \ldots, N$ **do**
  - $\mathbf{v}' \leftarrow$ random Gaussian in D dimensions
  - $\mathbf{v} \leftarrow \exp(-|f|)\mathbf{v} + \sqrt{1-\exp(-2|f|)}\mathbf{v}'$ *Partial momentum refresh*
  - **if** $k < N$ **then**
    - $\mathbf{x}_{k+0.5} \leftarrow \mathbf{x}_k + 0.5\Delta s \cdot \mathbf{v}/|\mathbf{v}|$ *Drift step*
    - $\nabla \ln n \leftarrow \frac{\nabla \ln(\mathcal{L}(\mathbf{x})/W(\mathbf{x}))}{D-1}|_{\mathbf{x}=\mathbf{x}_{k+0.5}}$ *Log refract. index gradient*
    - $(\mathbf{v}, \Delta \ln L) \leftarrow$ UpdateV($\mathbf{v}, \nabla \ln n, D, \Delta \ln L, \Delta s$) *Kick step*
    - $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_{k+0.5} + 0.5\Delta s \cdot \mathbf{v}/|\mathbf{v}|$ *Drift step*
- $r \leftarrow$ uniform random number in $(0, 1)$
- **if** $\ln(r) > \ln\left(\frac{\mathcal{L}(\mathbf{x}_N)W(\mathbf{x}_0)}{\mathcal{L}(\mathbf{x}_0)W(\mathbf{x}_N)}\right) - \Delta \ln L$ **then** *Metropolis test*
  - $\mathbf{x}_N \leftarrow \mathbf{x}_0$
- **return**$(\mathbf{x}_N, W(\mathbf{x}_N))$ *Return new sample and sample weight*

procedure UpdateV($\mathbf{v}, \nabla \ln n, D, \Delta \ln L, \Delta s$)
- **if** $|\nabla \ln n| = 0$ **or** $|\mathbf{v}| = 0$ **then**
  - **return**$(\mathbf{v}, \Delta \ln L)$ *Cases with no change in direction*
- $\hat{\mathbf{v}} \leftarrow \mathbf{v}/|\mathbf{v}|$
- $\hat{\mathbf{n}} \leftarrow \nabla \ln n/|\nabla \ln n|$ *Direction of refractive index gradient*
- **if** $|\hat{\mathbf{v}} \cdot \hat{\mathbf{n}}| = 1$ **then**
  - $\Delta \ln L \leftarrow \Delta \ln L + \Delta s \cdot |\nabla \ln n| \cdot (\hat{\mathbf{v}} \cdot \hat{\mathbf{n}})$
  - **return**$(\mathbf{v}, \Delta \ln L)$ *Velocity parallel to index gradient*
- $\theta_i \leftarrow 2\arctan\left(\frac{|\hat{\mathbf{v}}-\hat{\mathbf{n}}|}{|\hat{\mathbf{v}}+\hat{\mathbf{n}}|}\right)$ *Initial angle between ray path and index grad.*
- $\theta_f \leftarrow 2\arctan\left(\frac{|\hat{\mathbf{v}}-\hat{\mathbf{n}}|}{|\hat{\mathbf{v}}+\hat{\mathbf{n}}|\exp(|\nabla \ln n|\Delta s)}\right)$ *Final angle*
- $f_v \leftarrow \sin(\theta_f)/\sin(\theta_i)$
- $f_n \leftarrow \cos(\theta_f) - f_v \cdot \cos(\theta_i)$
- $\mathbf{v} \leftarrow f_v\mathbf{v} + f_n|\mathbf{v}|\hat{\mathbf{n}}$ *Direction update, maintains velocity normalization*
- $\Delta \ln L \leftarrow \Delta \ln L + (1-D)\ln(f_v)$ *Luminosity change*
- **return**$(\mathbf{v}, \Delta \ln L)$

The general ray tracing algorithm with arbitrary weighting function $W(\mathbf{x})$ is shown as Algorithm 1 in pseudocode. Notes include:

1. The algorithm is explicitly independent of velocity normalization and is explicitly time-reversible. Although we show a traditional drift-kick-drift leapfrog integrator here, any symmetric integrator may be used with the UpdateV function for kick steps. Ideally, momentum refreshes should be done in between integration steps to avoid path heating per Section 2.6.

2. At each integration step, the old velocity vector is rescaled by a factor $f_v = \sin(\theta_f)/\sin(\theta_i)$ and added to the new velocity vector, corresponding to a phase-space compression factor of f_v^{1-D}, i.e., $\sin^{D-1}(\theta_i)/\sin^{D-1}(\theta_f)$, exactly matching the radiance boost in Eq. 16.

3. The angle between the velocity and the refractive index gradient could be computed more transparently as $\theta = \arccos(\hat{\mathbf{v}} \cdot \hat{\mathbf{n}})$, but the version shown is more numerically

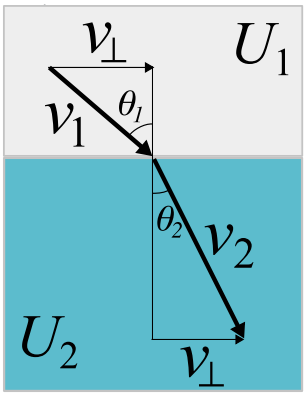

FIG. 4.— Path dynamics of Hamiltonian Monte Carlo. A particle moving from potential U_1 to U_2 experiences a force normal to the potential boundary, causing it to accelerate. The velocity component perpendicular to the force, $\mathbf{v}_\perp$, is unaffected. Hence, we have the geometrical identity $|\mathbf{v}_1| \sin\theta_1 = |\mathbf{v}_\perp| = |\mathbf{v}_2| \sin\theta_2$, and the path is identical to that of a light ray traveling in a medium with refractive index $n = |\mathbf{v}|$. Since the magnitude of $\mathbf{v}$ is only a function of the potential energy for a given choice of total energy E, this results in a well-defined solution for n for all particle paths at fixed total energy.

stable if the usual function (`atan2`) is used with the numerator and denominator of the fraction.

4. The sampler could be equivalently recast in an unweighted form (i.e., returning x_N with uniform weight) for arbitrary $W(\mathbf{x})$ by symmetric updates to the speed instead of moving a constant distance Δs (e.g., Appendix B), or by setting the Metropolis acceptance probability to the ratio of weights of the starting and final positions.

5. The Metropolis criterion could also be equivalently recast as a sample weight or an inverse velocity, corresponding to $\exp(\Delta \ln \mathcal{L} - \Delta \ln L)$, where the delta is taken across all prior steps; this would then allow a 100% acceptance rate.

6. Sample weighting could be equivalently recast as sample *masking* or *discretization*, wherein the number of copies of a particular point $\mathbf{x}$ in the chain is drawn from a Poisson distribution with the Metropolis expectation value. This also allows a "perfect" acceptance rate, except that some points in the chain are skipped. For HMC, this can be a poor choice, since once a trajectory diverges from constant energy, it usually does not recover. However, for ray tracing, even catastrophic errors are recoverable, as the light path will be redirected towards higher-likelihood regions as in Fig. 1.

2.9. *Integrating prior methods into the framework of generalized ray tracing*

The framework of generalized ray tracing is convenient for interpreting existing samplers, since it allows solving for the path curvature for an arbitrary sample weighting (equivalently, the speed of propagation along the path) and vice versa. In particular, this lets us simply compute sample weighting for samplers that are primarily defined based on their paths (e.g., HMC) for straightforward comparison with other methods.

2.9.1. *Common gradient-based approaches: Hamiltonian and Langevin Monte Carlo Methods*

We can reproduce traditional Hamiltonian dynamics straightforwardly using a ray tracing sampler. The analogy to Fig. 2 is a particle encountering a sudden jump in the potential energy $U = -\ln \mathcal{L}(\mathbf{x})$. The change in specific kinetic energy, $T_2 - T_1$, will be the opposite of the potential change, $U_2 - U_1$. We can then solve for the change in direction due to the potential change by breaking the velocity into components. By symmetry, the velocity $\mathbf{v}_\perp$ perpendicular to the potential gradient ∇U will remain unchanged. However, the velocity parallel to ∇U will be increased or decreased such that the direction of propagation becomes closer to the direction of the acceleration $-\nabla U$. Hence, the geometry (Fig. 4) requires:

$$|\mathbf{v}_1| \sin\theta_1 = |\mathbf{v}_\perp| = |\mathbf{v}_2| \sin\theta_2. \tag{24}$$

This is immediately suggestive of the form of Snell's law, and so the general solution for n is[2]

$$n(\mathbf{x}) \propto |\mathbf{v}(\mathbf{x})| \tag{25}$$

by inspection, with the specific solution

$$n(\mathbf{x}) \propto |\mathbf{v}(\mathbf{x})| \propto \sqrt{E + \ln \mathcal{L}(\mathbf{x})}, \tag{26}$$

for the usual specific kinetic energy ($T = \frac{1}{2}v^2$), with E being the total specific energy. This formulation as a ray tracing approach gives paths and weights (inverse velocities) that are identical to standard Hamiltonian Monte Carlo sampling. For example, mechanical paths can only reach points that satisfy $E + \ln L(\mathbf{x}) > 0$. This is also true of the ray paths: at positions $\mathbf{x}$ where $\ln \mathcal{L}(\mathbf{x}) = -E$, the gradient of the log refractive index is infinite, as $n = 0$, corresponding to an impassible barrier. Langevin Monte Carlo methods then follow, as they can be rewritten as single-step Hamiltonian Monte Carlo methods (Girolami & Calderhead 2011).

The parameter-space sampling density of HMC is L/v, which for $n \propto |\mathbf{v}|$ becomes v^{D-2}. This results in a sampling bias at fixed energy, since some regions will be sampled preferentially compared to the likelihood function:

$$\frac{L(\mathbf{x})/v(\mathbf{x})}{\mathcal{L}(\mathbf{x})} = \frac{(E + \ln \mathcal{L}(\mathbf{x}))^{D/2-1}}{\mathcal{L}(\mathbf{x})}. \tag{27}$$

This results in some interesting behavior in low-dimensional spaces. Indeed, for $D = 2$, HMC sampling at a single value of E will yield a uniform(!) density regardless of $\mathcal{L}(\mathbf{x})$, for all regions with $\ln \mathcal{L}(\mathbf{x}) > -E$. This is readily confirmed numerically by refreshing only the velocity direction but not the kinetic energy for an HMC sampler.

For high-dimensional spaces, the behavior is more regular. We expect for a Gaussian-distributed velocity that $|v| \sim \sqrt{D}$, so typical kinetic energies (i.e., $E + \ln \mathcal{L}(\mathbf{x})$) will be $\sim D/2$. Hence, we can reparameterize the bias equation in terms of a change in the likelihood $\Delta \ln \mathcal{L}$ away from the starting likelihood, giving:

$$\frac{L(\mathbf{x})/v(\mathbf{x})}{\mathcal{L}(\mathbf{x})} \sim \left(\frac{D}{2}\right)^{D/2-1} \cdot \frac{(1 + \frac{2}{D}\Delta \ln \mathcal{L}(\mathbf{x}))^{D/2-1}}{\mathcal{L}(\mathbf{x})}. \tag{28}$$

In the limit of small $(\Delta \ln \mathcal{L})/D$, the rightmost fraction asymptotes to $\exp(\Delta \ln \mathcal{L})/\mathcal{L}(\mathbf{x}) \propto 1$, resulting in sampling density proportional to the likelihood function, similar to the behavior of ray tracing. This limit occurs after burn-in for large D,

[2] It is tempting to expect $v \propto n^{-1}$, as for physical light rays. However, the physics of particle vs. light transport are different—for example, the speed $v_\perp$ is not conserved for light rays propagating between two media. The better analogy is with the Minkowski momentum of light, which indeed scales with n as $p = h/\lambda = nh/\lambda_{\rm vac}$.

where the central limit theorem suggests that the typical set should have a log-likelihood width of $\Delta \ln \mathcal{L} \propto \sqrt{D}$. However, during burn-in, $|\Delta \ln \mathcal{L}|$ can be large, with HMC undersampling more relative to the expectation for fair sampling (i.e., compared to ray tracing) unless the momentum is periodically refreshed. This suggests that a very simple non-reversible method for approximating ray tracing with an existing HMC sampler is to resample the kinetic energy (but not the direction) at every integration step (following a Metropolis check), and to resample the velocity direction only after much longer trajectories. Ray tracing, as implemented in Eq. 23, is effectively this energy resampling process taken to the continuous (and reversible) limit.

Alternate formulations of HMC that better explore multimodal regions (e.g., parallel tempering; see Earl & Deem 2005 for a review) do so by reducing the velocity differences (and hence sample density differences) between high– and low–probability regions, corresponding to reducing the refractive index gradient (see also Section 2.9.5).

We can also simply derive microcanonical Hamiltonian sampling, which corresponds to setting $W(\mathbf{x}) = \mathcal{L}(\mathbf{x})^{1/D}$ as in Robnik et al. (2022).[3] This results in a refractive index of $n(\mathbf{x}) = \mathcal{L}(\mathbf{x})^{1/D}$. At large D, we hence expect the path dynamics of microcanonical HMC to be similar to ray tracing, and—based on the analysis above—we expect the sampling density of both methods to be similar to HMC after burn-in. The main difference of the ray tracing formulation is hence the complete independence of the path speed (or weight) on the likelihood, enabling strong noise resilience for stochastic gradients.

2.9.2. *Goodman-Weare sampling*

The stretch step proposed in Goodman & Weare (2010) (and familiar as a common step in Foreman-Mackey et al. 2013) proposes a new position $\mathbf{x}$ along a line connecting the current position $\mathbf{x}_c$ and a target walker position $\mathbf{x}_t$ from a complementary set of walkers. This is equivalent to setting $n = \infty$ outside of a small cone with infinitesimally small solid angle $d\Omega$ originating from the target walker's position $\mathbf{x}_t$ and including the current walker's position $\mathbf{x}_c$. Inside the cone, the sample weighting is given by the stretch step distribution size, namely $W(z) = \frac{1}{\sqrt{z}}$ for $z \in [\frac{1}{a}, a]$ and 0 otherwise, with a the usual step size parameter and $z(\mathbf{x}) = |\mathbf{x} - \mathbf{x}_t|/|\mathbf{x}_c - \mathbf{x}_t|$. The narrowing shape of the cone means that the net volume at a given z would scale as $z^{D-1}dz$, correspondingly limiting the fraction of rays that would penetrate to low z, and matching the acceptance criterion of Goodman & Weare (2010). We could hence more simply recast a series of stretch steps with the same $\mathbf{x}_t$ as 1D Gibbs sampling, with $\mathcal{L}(z) = z^{D-1}\mathcal{L}(\mathbf{x}(z))$, and use the approach for Gibbs sampling in Section 2.9.3.

2.9.3. *Metropolis Sampling, Gibbs Sampling, and Monte Carlo Integration*

Gibbs methods (Geman & Geman 1984), operate by sampling from the conditional distribution, often along single-parameter axes. In the ray tracing framework, this corresponds to setting $W(\mathbf{x}) = \mathcal{L}(\mathbf{x})$ and $n = 1$ everywhere. As a result, rays travel in straight lines throughout parameter space, and the weights ensure sampling from the one-dimensional conditional distribution. As expected, both energy and phase-space volume are always conserved with such ray paths (as $\nabla \ln n = 0$), and so the probability conservation test in Eq. 19 is automatically satisfied.

A Metropolis-Hastings (Metropolis et al. 1953; Hastings 1970) step corresponds to performing a single ray tracing step using the Euler method for integration, with size Δs drawn from a desired distribution (typically Gaussian), with $n = 1$ and $W(\mathbf{x}) = \mathcal{L}(\mathbf{x})$, and with the choice of unweighted samples based on the Metropolis criterion instead of a sample weight or a speed.

Last, Monte Carlo integrals can be thought of as Gibbs sampling with masked samples—that is, weights are set to 0 everywhere except for ray endpoints (i.e., where the direction of travel is refreshed), where they are set to $\mathcal{L}(\mathbf{x})$.

2.9.4. *Refractive Sampling*

While searching for prior work during paper preparation, we became aware of a not elsewhere published thesis chapter by Levi Boyles, called "Refractive Sampling," in which the authors used ray transport equations and derived similar acceptance criteria in 2014. In their case, they adopted $\nabla \ln n = \frac{r}{\Delta s} \frac{\nabla \ln \mathcal{L}}{|\nabla \ln \mathcal{L}|}$, so that the direction of the refractive index gradient was tied to the direction of $\nabla \ln \mathcal{L}$, but not the magnitude, as they did not derive the conservation of basic radiance. Instead, the authors evaluated using an effective refractive index ratio r that was constant. This corresponds to a strong overweighting of low-likelihood regions; unsurprisingly, the authors found that standard HMC performed substantially better for unimodal distributions, but that refractive sampling had the potential to better access multimodal distributions.

2.9.5. *Tempering*

Since generalized ray tracing separates the path dynamics from sample weights/velocities, it is straightforward to create fair samplers with path dynamics corresponding to a different effective temperature τ—or even allowing the temperature to vary along the path. For standard ray tracing dynamics, this is a simple substitution:

$$n = \mathcal{L}(\mathbf{x})^{\frac{1}{\tau \cdot (D-1)}} \tag{29}$$

$$W = v^{-1} = \mathcal{L}(\mathbf{x})^{\frac{\tau-1}{\tau}}. \tag{30}$$

As expected, increasing the temperature corresponds to increasing the effective speed (or decreasing the effective weight) in unlikely regions, reflecting the fact that the paths will overpopulate regions with low likelihood. We note that there is no issue with making τ a function of position or likelihood, to better explore long-tailed distributions or to better travel between modes of the distribution. For example, setting $\tau \propto \ln \mathcal{L}$ is equivalent to Gibbs sampling, as follows from the identity $\ln(x^{1/\ln(x)}) = 1$. In another example, if τ is a function of $\ln(\mathcal{L})$, then the spatial gradient of $\ln(n)$ used in Eq. 23 becomes:

$$\nabla_{\mathbf{x}} \ln(n(\mathbf{x})) = \frac{\nabla_{\mathbf{x}} \ln(\mathcal{L}(\mathbf{x}))}{\tau \cdot (D-1)} \left[1 - \ln(\mathcal{L}(\mathbf{x})) \frac{d \ln(\tau)}{d \ln(\mathcal{L}(\mathbf{x}))} \right], \tag{31}$$

which has the useful property that the direction of $\nabla_{\mathbf{x}} \ln(n)$ is parallel (or anti-parallel) to the $\tau = 1$ case.

The temperature τ could also be a function of time, provided that it is symmetric along the trajectory to maintain reversibility. In this case, the effective weighting should be the product

[3] Robnik et al. (2022) noted the connection between geodesics and sampling density, calling it an "interesting open question" as to whether it could be used to inform sampler design. This paper is one answer.

of weight ratios at constant temperature, as the weights are not conserved during temperature changes. A disadvantage to this approach is that, by definition, the stationary distribution will change with time. If the timescale for burn-in is longer than the timescale for temperature change, changing the temperature in this manner will lead to many rejected paths.

We note in passing that, despite the method above, HMC paths do not admit varying temperatures unless the starting and ending locations have the same likelihood. This is because, for HMC, it is otherwise impossible to guarantee energy preservation at multiple temperatures simultaneously, leading to non-reversible paths (Neal 2012).

2.10. *Adaptive Sampling and Separability*

Generically, all samplers benefit from adapting the transition function to the target distribution. Existing approaches including adaptive covariance matrices as in Haario et al. (2001) and normalizing flow–enhanced approaches like Wong et al. (2023) are hence expected to improve the performance of ray tracing.

Adaptive covariance matrix approaches attempt to perform an affine transformation of the parameter space to remove parameter-space degeneracies. For a D-dimensional independent and identically-distributed (i.i.d.) Gaussian distribution, typical ray paths will correspond to roughly circular orbits in the limit of large D. Under an affine transformation, the distribution will become ellipsoidal. Typical ray paths will still have an isotropic initial velocity distribution, resulting in helical paths along the ellipsoidal surface. The greater the aspect ratio of the ellipsoid, the tighter the winding of the helix, and the more timesteps that will be necessary to find an independent sample. Hence, in large-dimensional problems where measuring the full covariance matrix is intractable, the largest improvement will come from rescaling the space to lessen the dynamic range of the eigenvalues—i.e., targeting the largest and smallest principal components of the space.

With the above said, we do not attempt rescaling of the space for the neural network applications in this paper. Because neural networks have a high degree of symmetry (e.g., adjacent nodes can be swapped without affecting the results of the network), renormalizing the space does not necessarily lead to improved sampling performance. Instead, the filamentary nature of the likelihood space for neural networks means that there are highly local changes in the Hessian matrix that make a global rescaling less helpful.

We also note the related topic of separability. Many samplers, including Hamiltonian Monte Carlo, Metropolis, and Gibbs sampling are *separable*, in the sense that if the likelihood function can be written as the product over two independent subspaces ($\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$ with $\mathcal{L}(\mathbf{x}) = \mathcal{L}(\mathbf{x}_1)\mathcal{L}(\mathbf{x}_2)$), then the dynamics of exploration over the subspace $\mathbf{x}_1$ are not affected by the dynamics of the exploration in the subspace $\mathbf{x}_2$. This is not generically true for ray tracing, since ray tracing dynamics have the constraint of constant velocity, which links the dynamics of any two subspaces together. Hence, if the gradients are expected to be of significantly different magnitude for two different subspaces (e.g., different layers of a neural network), it may be advantageous to run ray tracing dynamics on each subspace independently—i.e., to have independent velocity vectors for each subspace and to update them in parallel according to Algorithm 1.

2.11. *Practical Sampling of Neural Networks*

High-dimensional problems often have large computational costs, necessitating some trade-offs to maximize sampling accuracy given the computing time available.

2.11.1. *Lack of a well-defined likelihood function*

This issue arises most often for regression tasks in which the network is fit to minimize a loss function (e.g., mean squared error; MSE), rather than to predict the distribution of the training data. As the underlying label uncertainty is often assumed to be zero, the mean squared error cannot be interpreted as a χ^2 or similar value in such cases. However, one can still usefully ask about the posterior distribution of outputs for neural networks that perform within some tolerance $\Delta f_{\rm loss}$ of the best fit. We can then construct a likelihood function that will lead to typical losses within $\Delta f_{\rm loss}$ of the best solution as:

$$\ln \mathcal{L}(\mathbf{x}) \approx -f_{\rm loss}(\mathbf{x}) \cdot (D_{\rm eff}/(2\Delta f_{\rm loss})), \tag{32}$$

where $D_{\rm eff}$ is the effective number of parameters, $f_{\rm loss}$ is the loss function, and $\Delta f_{\rm loss}$ is the desired tolerance. This equation is exact for a Gaussian distribution, where the the expectation value of the log likelihood should be $\sim D/2$ less than the best-fitting log likelihood value (equivalently, χ^2 should be higher by $\sim D$). Heuristically, the central limit theorem suggests that it should also be reasonable for other distributions in the limit of large numbers of dimensions, provided that at least a small fraction of dimensions are uncorrelated with each other.

We note that $D_{\rm eff}$ will in practice only equal the total number of parameters if all the parameters are constrained by the data set. Neural networks can have extensive symmetries and unused parameters, so the true effective parameter count can be significantly smaller (e.g., 3% of the neural network parameter count in Section 3.2), and particularly so if the number of data points in the training set is much less than the parameter count (e.g., 0.06% of the parameter count in Section 3.3). Hence, some trial and error can be necessary to choose $D_{\rm eff}$ so as to achieve the desired loss tolerance.

2.11.2. *Lack of Convergence in Weight Space*

MCMC samplers are expected to converge asymptotically to the correct distribution for any function of the underlying parameter space, which for neural networks is also known as *weight space*. However, for most machine learning applications, the distribution in weight space is not directly used, and instead the distribution in *function space*—usually, the output of the neural network—is what is important. Typically, posteriors in weight space for neural networks are narrow tubes or sheets, and so convergence in weight space is difficult (Izmailov et al. 2021); but due to the many symmetries of neural networks, the effective convergence in function space is much more rapid.

We also note that it is common to apply a Gaussian prior in weight space of the form $\mathcal{L} \propto \exp(-\mathbf{x}^2/\alpha^2)$ (e.g., Izmailov et al. 2021). An advantage of this is that the posterior distribution in weight space will be bounded and finite, which can improve convergence in weight space. A disadvantage is that the choice of α is usually arbitrary, and if α is chosen to be too small, it can alter the meaning of the effective number of parameters $D_{\rm eff}$. Hence, if using a weight space prior, it is typically necessary to try multiple values of α to make sure that the choice of α does not affect the distribution of the posterior in function space. If not using a weight space prior (the approach we adopt here), the advantage is that a range of effective α is probed automatically, with the disadvantage that the weight space distribution may never converge.

2.11.3. *Stochastic Likelihood Sampling*

Training neural networks is greatly accelerated by splitting the training data into multiple batches and using the loss gradients of individual batches to perform successive fitting steps instead of calculating the loss gradient of the entire training sample at each step. Depending on batch sizes, this can speed up the rate of convergence by orders of magnitude, especially at early times when the network is far from a good solution.

Stochastic sampling of the likelihood gradient has been explored for traditional HMC in Chen et al. (2014), which found that a friction term is necessary to counterbalance the effective heating of using an imperfect gradient estimator. Chen et al. (2014) argued that such an approach was critical, since for arbitrarily long orbits, the amount of energy injected by stochastic sampling could be arbitrarily large, leading to arbitrarily bad mismatches between the sampling and the target distribution. On the other hand, it is not always easy to estimate the correct noise caused by stochastic sampling, including the noise covariance, so it is difficult to accurately cancel the impact of stochastic sampling for HMC.

With ray tracing, however, the energy is explicitly preserved, even in the ray tracing formulation of Hamiltonian Monte Carlo (Section 2.9). The impact of stochasticity on the dynamics is instead limited to perturbations in the direction of travel. To first order, this is in fact desired behavior(!), given that, in Section 2.5, we had to explicitly add continuous partial momentum refreshment to achieve ergodicity.

Of course, what we lose with stochastic sampling is perfect knowledge of the likelihood function. The results are errors in calculating the Metropolis acceptance criterion in Eq. 19, which result in 1) increased rejections, and 2) broadening of the typical set. To reduce the problem of increased rejections, we can interpret the stochastic likelihood as the true likelihood convolved with an aleatoric uncertainty. What matters then is that the error in path integration remains subdominant to the uncertainty in calculating the likelihood. We can approximate this with a stochastic acceptance criterion as:

$$P(\mathbf{x} \to \mathbf{y}) = \min\left(1, \left(\frac{n(\mathbf{x})^{1-D} L(\mathbf{x})}{n(\mathbf{y})^{1-D} L(\mathbf{y})}\right)^{\frac{1}{\sqrt{1+\sigma_{\rm sto}^2}}}\right), \tag{33}$$

where $\sigma_{\rm sto}^2$ is the variance in $\ln(\mathcal{L}_{\rm batch}/\mathcal{L}_{\rm true})$. This criterion reflects the fact that we have imperfect information about whether the path is erroneously integrated or whether the likelihood was erroneously calculated. Notably, because it is an acceptance criterion and does not otherwise affect the dynamics of the system, it will remove paths that deviate badly from the expected trajectories, but it will not lead to runaway growth of the loss function or to log sampling rates that deviate by $\gg \sigma_{\rm sto}$ from the true values. In principle, step sizes should be lowered until acceptance rates per Eq. 33 stop improving significantly.

With the above said, ray tracing performs so well with suppressing heating from stochastic gradients that it is easy to have $\sigma_{\rm sto} \gg \sigma_{\ln(\mathcal{L}_{\rm true})}$ while still achieving low error on the resulting true likelihood distribution (as in Section 3.1.2). In such cases, Eq. 33 loses meaning as a test for acceptance of individual MCMC steps, since it will simply randomly discard valid samples from the chain. A simpler alternative in such cases is to set a likelihood threshold (e.g., $\pm 2\sigma_{\rm sto}$) for individual steps to reject badly deviating paths, and then to verify that decreasing the step size does not change any moments of the distribution.

2.12. *Sampling Recipe*

We have found the following steps helpful for sampling the neural networks described in Sections 3.2–3.4:

1. Perform burn-in of a single walker at a low temperature (or use an optimizer like Adam) to find a location near the typical set. For HMC burn in, frequent momentum refreshes are needed to remove excess kinetic energy.

2. If necessary, choose the likelihood scaling in Eq. 32 according to the desired loss tolerance $\Delta f_{\rm loss}$.

3. Adjust the step size, starting with a reasonable guess (e.g., $\Delta s \sim 0.03 D^{1/2}$), and running until the loss distribution stabilizes. Try reducing the step size until the averaged loss (or inverse log-likelihood) per epoch stops decreasing. It may be necessary to adjust the likelihood scaling to maintain the desired $\Delta f_{\rm loss}$.

4. Adjust the refresh rate. Lower refresh rates often lead to faster exploration (less diffusion), at the cost of more path heating due to stochastic gradients—more of an issue for HMC than for ray tracing. However, lowering the refresh rate too far can lead to too much time spent exploring the same trajectory, so eventually autocorrelation times can increase. Ray tracing generally has a wide window for similar results with different refresh rates.

5. Perform full sampling run: initialize the desired number of walkers with random initial conditions, perform burn-in on each walker, and then perform approximate MCMC sampling with fixed hyperparameters.

6. Validate with a convergence test in the function space (e.g., the values of the network predictions on the validation set), as well as verifying the loss on the test set.

3. APPLICATIONS

In this paper, we focus on the applications to parameter spaces with large numbers of dimensions (> 1000). This is because global and hybrid global/local samplers like nested sampling and normalizing flows are often better choices for lower-dimensional problems. For such high-dimensional problems, the only practical algorithms are gradient-based, including Hamiltonian Monte Carlo (HMC) and ray tracing. We first examine sampling dynamics for a 10,000-dimensional Gaussian distribution in Section 3.1, and then apply the ray tracing method to neural networks with 1400 to 1.5 billion parameters in Sections 3.2 to 3.4. Additional comparisons with HMC for standard distributions appear in Appendix D. Herein, all autocorrelation times are computed using the Sokal (1996) estimator.

3.1. *Ray Tracing on Gaussian Distributions*

3.1.1. *Perfect Gradient with Imperfect Integrators*

In this section, we consider ray tracing dynamics on a 10,000–dimensional independent and identically distributed (i.i.d.) Gaussian distribution ($\ln \mathcal{L} = -0.5|\mathbf{x}|^2$), with a perfect gradient calculator. We consider both the Metropolis-corrected and uncorrected cases, given the existence of both uncorrected and corrected microcanonical HMC methods (i.e., Robnik et al. 2022, 2025).

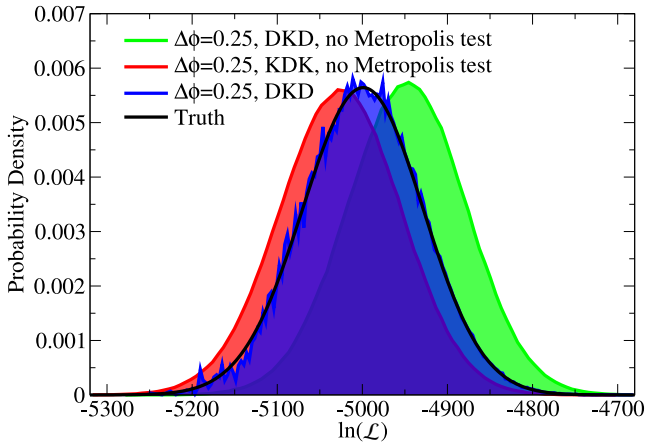

FIG. 5.— This figure shows ray tracing samples for a 10,000–dimensional Gaussian likelihood ($\ln \mathcal{L} = -0.5|\mathbf{x}|^2$), using a moderate step size equivalent to a change in propagation direction of $\langle \Delta\phi \rangle = 0.25$ at each step. The non-Metropolis cases both converge to stable distributions that are biased, so a Metropolis test can be helpful to achieve larger step sizes when exact likelihoods are used.

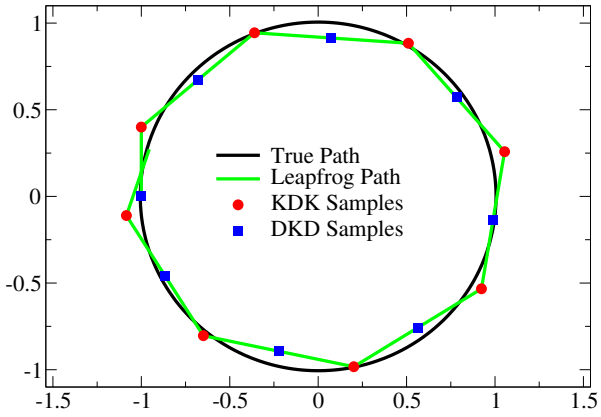

FIG. 6.— Leapfrog path ($\Delta\phi = 0.8$) vs. true path for the potential $U = x^2$. Kick-drift-kick integrators sample the *red points*, which are typically outside the true path, whereas drift-kick-drift integrators sample the *blue points*, which are typically inside the true path. At in-between locations, the path radii are correct, but the velocities are not tangent to the true path. Randomly choosing between KDK and DKD steps is one way to have correct expectation values for both the positions and velocities, with the tradeoff of larger path variance.

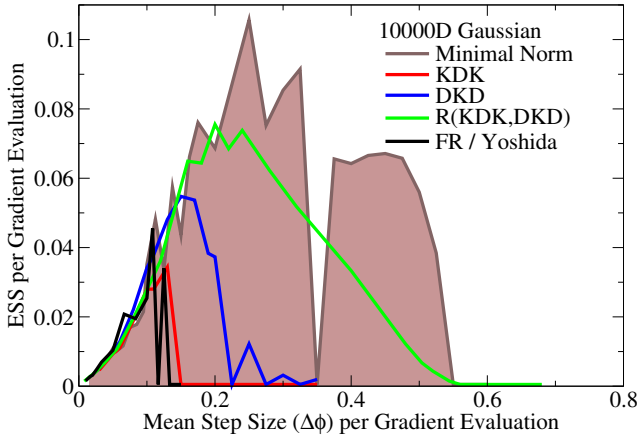

FIG. 7.— Effective sample sizes (ESS) per gradient evaluation for kick-drift-kick (KDK), drift-kick-drift (DKD), Omelyan et al. (2003) Minimal Norm second-order (Minimal Norm), and Forest & Ruth (1990)/Yoshida (1990) integrators, all for the same 10,000–dimensional Gaussian likelihood. All show a characteristic sharp dropoff with increasing step size (shown in units of the average angular change $\Delta\phi$). We find that randomly choosing between KDK and DKD steps smooths the dropoff substantially, and the Omelyan et al. (2003) integrator improves performance even further.

When perfect gradients are available, we find that it may be more efficient to use a Metropolis-corrected integrator, since the step sizes can be much larger. We show, for example, the likelihood distributions for kick-drift-kick (KDK) and drift-kick-drift (DKD) leapfrog integrators in Fig. 5. Without a Metropolis test, the KDK integrator converges to a lower-likelihood distribution, and the DKD integrator converges to a higher-likelihood distribution as compared with the true distribution, which is recovered accurately when a Metropolis test is used. Although KDK and DKD integrators will trace identical paths, they will sample different locations along that path, as shown in Fig. 6, with KDK sampling at larger radii than DKD. Intuitively, KDK will sample at the vertices of the polygonal path (which tend to be at lower likelihoods), while DKD will sample at the midpoints of the edges of the polygonal path (which will tend to be at higher likelihoods), as shown in Fig. 6.

For each integration method and step size we test, we start 500 independent chains from a Gaussian distribution with 4× the standard deviation as the true likelihood function. We then discard the first 1500 steps as burn-in, the earliest 1000 of which have no Metropolis test applied. The latter choice is because the ray tracing technique functions effectively as a gradient descent algorithm when it is far away from the typical set. We refresh the path direction and apply a Metropolis test every $N = \sqrt{D} \cdot \pi/(2\Delta s)$ steps, corresponding to traveling an angle of $\pi/2$ within the typical set, which is optimal for Gaussian distributions. To evaluate performance, we use the metric of effective sample size (estimated using the ratio of the chain length to the autocorrelation time) per gradient computed. To be conservative, we take the autocorrelation time to be the maximum of 1) the average autocorrelation time of the individual parameter space dimensions, 2) the average autocorrelation function of the absolute values of the individual parameter space dimensions, and 3) the autocorrelation time of $\ln(\mathcal{L})$. This ensures proper mixing both in the phases and in the energies of the resulting distribution.

We show the resulting performance for several step sizes and symplectic integrators, including KDK, DKD, the minimal norm second-order integrator of Omelyan et al. (2003), and the fourth-order integrator of Forest & Ruth (1990) and Yoshida (1990) in Fig. 7. To provide a better comparison across different integration orders, we report the mean step size per gradient evaluation ($\Delta\phi \equiv \pi/(2NG)$), where N is the path step count as above and G is the number of gradients per full integration step), instead of the total step size across all integration stages.

At low step sizes, acceptance rates are high due to low cumulative path errors, and so all integrators show a universal rise due to a lower number of gradients needed for a given path length. The KDK, DKD, and Forest-Ruth / Yoshida integrators all show a steep drop-off at low step sizes, because these methods often fail to reach the typical set by the time burn-in is over. For DKD and KDK steps, the cause is overshoot or undershoot (Fig. 6), and for the Forest-Ruth / Yoshida integrator, the cause seems to be the large positive step sizes, as the method contains a negative step in the middle. The simplest method to improve the performance of DKD/KDK steps is to randomly choose between the two along the path, which offers better effective sample sizes over a wider range of steps. However, the Omelyan et al. (2003) method offers the best performance over the widest range of step sizes, as it minimizes both velocity and position errors relative to other

methods.

We expect the situation to be different at much higher dimensionality, where the errors of fourth-order methods improve more rapidly than the errors of second-order methods as the step size shrinks. Following Neal (2012), we can estimate that both the step size and the effective sample size per gradient will scale as $D^{-1/(2k)}$, with k the order of the integration method, for i.i.d. Gaussian distributions. Hence, the factor of ~ 3 advantage in effective sample size of the Omelyan et al. (2003) second-order over the Forest-Ruth / Yoshida fourth-order integrators is expected to disappear around a few tens of millions of dimensions. Empirically, we find this to be the case at ∼30 million dimensions. We note that Omelyan et al. (2003) includes an 11-stage fourth-order algorithm that may have an even higher performance than the Forest-Ruth / Yoshida integrator and hence a lower crossover dimensionality.

As discussed in Appendix D, we find that ray tracing and HMC have roughly equivalent performance when tuned appropriately, across a range of standard tests including Gaussian, Rosenbrock, and Cauchy distributions. The main performance benefit of ray tracing discussed in this paper comes from its resilience to heating in stochastic gradients, discussed in the following section.

3.1.2. *Stochastic Gradients*

When training neural networks, it is often too costly to have a perfect likelihood function, and so the tradition is to sample from a stochastic gradient of the log-likelihood—i.e., a distribution that has the expectation value of the true log-likelihood gradient with some scatter. For a Gaussian distribution, we can approximate this in a few different ways, the simplest being:

$$\mathcal{L}_{\rm sto}(\mathbf{x}) \equiv \exp\left(-\frac{1}{2}[\mathbf{x} - \mathbf{R}(\sigma_{\rm sto})]^2\right), \tag{34}$$

where $\mathbf{R}(\sigma_{\rm sto})$ is a random Gaussian distribution with variance $\sigma_{\rm sto}^2$. As required, $\langle \nabla \ln \mathcal{L}_{\rm sto}(\mathbf{x})\rangle = -\mathbf{x}$. However, the expected magnitude $\langle |\nabla \ln \mathcal{L}_{\rm sto}(\mathbf{x})|\rangle$ scales proportionally to $\sqrt{1+\sigma_{\rm sto}^2}$ for points in the typical set, so the distribution of $\mathcal{L}_{\rm sto}$ is very different from the distribution of the true likelihood function even for small values of $\sigma_{\rm sto}$.

As with the previous section, we choose $D = 10000$. We use a smaller step size ($\Delta\phi = 0.03$), and do not use a Metropolis test, because the expected uncertainty in evaluating the likelihood is typically far higher than the error from evaluating the path, as is shown later in this section. (In other words, had we used a Metropolis test, we would be randomly discarding samples, as opposed to removing any biases). As well, we use the R(KDK,DKD) integrator, finding that this seems optimal for both ray tracing and Hamiltonian Monte Carlo (HMC).

Results for the true likelihood distribution as a function of $\sigma_{\rm sto}$ are shown in Fig. 8. The simplicity of the figure belies how shocking it is. Only at $\sigma_{\rm sto} = 10$ does the likelihood distribution become visibly different from the true distribution for ray tracing. At this large value of $\sigma_{\rm sto}$, only 1% of the variance in the gradient is correlated with the actual location of the typical set, and typical values of the stochastic likelihood are a factor $\exp(-500000)$ smaller than the values of the true likelihood. In comparison, HMC shows worse performance with perturbations that have 100× less variance ($\sigma_{\rm sto} = 1$).

The performance benefit is equally stark in Fig. 9, which shows the error in the median of the log-likelihood distribution,

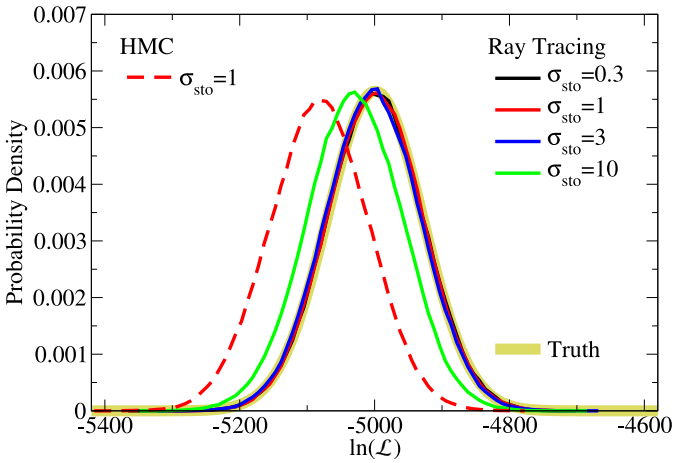

FIG. 8.— True likelihood distribution for the ray tracing algorithm using stochastic gradients (Eq. 34) with a 10,000-dimensional Gaussian distribution and a step size of $\Delta\phi = 0.03$. Minimal deviation from the true distribution (thick sand-colored line) is observed until $\sigma_{\rm sto} = 10$, corresponding to perturbing the center of the Gaussian by an amount 10 times larger than its standard deviation(!) By comparison, Hamiltonian Monte Carlo shows worse performance already by $\sigma_{\rm sto} = 1$, i.e., with 100 times smaller perturbation variance.

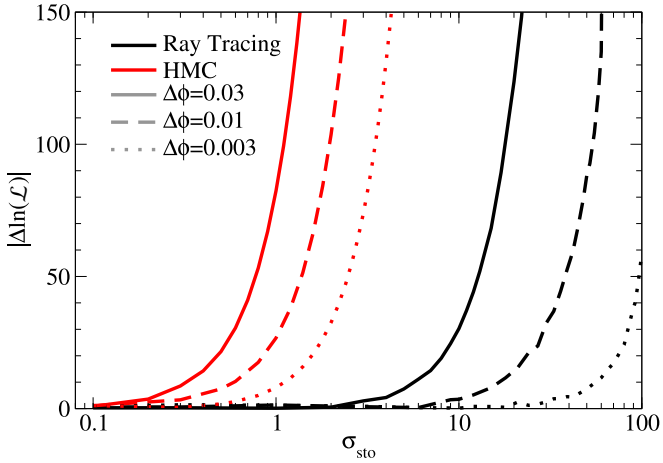

FIG. 9.— Comparison between the resilience of ray tracing and traditional Hamiltonian Monte Carlo (HMC) to noise in stochastic gradients, again for a 10,000-dimensional Gaussian distribution. The vertical axis shows the deviation in the median log likelihood ($\ln \mathcal{L}$) as a function of the gradient noise (parameterized by $\sigma_{\rm sto}$ in Eq. 34). At a moderate step size of $\Delta\phi = 0.03$, ray tracing can tolerate 16× larger $\sigma_{\rm sto}$ than HMC (∼ 250× more variance) for an equivalent accuracy in the likelihood posterior distribution. As the step size decreases, the advantage of ray tracing increases: the stochasticity resilience of HMC scales as only $O(\Delta\phi^{-1/2})$, whereas the resilience of ray tracing scales as $O(\Delta\phi^{-1})$.

compared to the true value, as a function of $\sigma_{\rm sto}$. Both HMC and ray tracing show errors that grow linearly in the variance of the stochastic gradients:

$$|\Delta \ln \mathcal{L}| = (\sigma_{\rm sto}/\sigma_c)^2, \tag{35}$$

where σ_c is a characteristic resilience size for stochastic gradient perturbations. For HMC in this application, $\sigma_c^2 = 0.012$, whereas for ray tracing, $\sigma_c^2 = 3.2$, more than 250× larger. This conclusion becomes even stronger as the step size is reduced. For HMC, the characteristic perturbation resilience size scales as $\sigma_c \propto 1/\sqrt{\Delta\phi}$, whereas for ray tracing, it scales as $\sigma_c \propto 1/\Delta\phi$ (Fig. 9). Given that the expected variance $\sigma_{\rm sto}^2$

will scale as $O(B^{-1})$, with B the mini-batch size, the total compute time will be independent of batch size B for HMC (i.e., $O(1)$): decreasing the mini-batch size by a factor of 100 requires decreasing the step size by a factor of 100, for no overall computational gain. However, for ray tracing, the total compute time will scale as $O(\sqrt{B})$: decreasing the mini-batch size by a factor of 100 would only require a factor 10 smaller step size, yielding the same effective sampling size with 10x less computational cost. Hence, ray tracing benefits from smaller mini-batch sizes (analogous to improved performance for stochastic gradient descent methods), whereas HMC does not.

We can understand the relative noise resilience intuitively. For both ray tracing and HMC, taking smaller steps (i.e., more steps for a given trajectory length) will reduce the effective gradient noise per unit distance traveled. However, as mentioned in Section 2.11.3, using stochastic gradients will heat the velocity component for HMC at every step (Chen et al. 2014), boosting the trajectory to higher energies (i.e., lower likelihoods). Hence, for HMC, larger step counts will partially cancel the improvement from lower average gradient noise per unit distance traveled. In contrast, the kinetic energy of the ray tracing path is fixed, so a longer trajectory simply means better averaging over the stochastic directions of the gradients, with no memory for previous heating, and no need for a diffusion term tailored to the network and loss function (as in Chen et al. 2014).

We note that robustness to stochastic gradients can be achieved for some other sampling methods in Section 2.9 by using weights with generalized ray tracing and keeping $|\mathbf{v}|$ constant. This leads to accurate paths, with inaccurate (stochastic) weights. Nonetheless, the correct weights can be recomputed for a subsample of the full chain in a postprocessing step. Given that the autocorrelation time is often large for parameter exploration of neural networks, the weights may only need to be calculated at large intervals, leading to an overall performance gain. In addition, weights may be initially computed at low accuracy, so that more expensive high-accuracy calculations can be saved for the locations that are expected to contribute most to the overall weight of the path. As a result, we expect that both traditional and microcanonical HMC sampling could (with appropriate implementation choices) be made to perform almost identically to ray tracing.

3.2. *Comparing Stochastic Ray Tracing and Stochastic HMC for a Multi-Layer Perceptron with 1433 Parameters*

Many scientific applications of neural networks involve using them as modest-dimensional fitting functions, for example to accelerate a slow code or to interpolate values within a large dataset. In this section, we build a small example neural network to accelerate the output of the UniverseMachine code (Behroozi et al. 2019), an empirical model that generates observed galaxy properties for gravity-only simulations. The network we use is simple enough for its posteriors to be explored both with HMC and with ray tracing, allowing us to compare the two approaches.

To fit the UniverseMachine outputs, we use a simple multi-layer perceptron (MLP), a type of neural network where the layers are fully connected (i.e., each neuron receives inputs from all the outputs of the previous layer). For this example application, the number of training data samples ($\sim$ 800k) is much larger than the number of network parameters ($D = 1433$). We also chose a mini-batch size (20,000) that was large with respect to the network parameter count, so that the expected stochasticity was relatively small, enabling HMC sampling to be used. Full details of the network inputs, outputs, architecture, hyperparameters, and hardware used are in Appendix E.1.

We found that it was straightforward to perform approximate MCMC sampling with both ray tracing and HMC on either a laptop computer or a consumer GPU following the recipe in Section 2.12. For both samplers, we chose the loss tolerance to be $\Delta f_{\rm loss} = 0.0042$, corresponding to an increase in RMSE of $\sim 10\%$ and a loss scaling of 5,000 (i.e., $-\ln \mathcal{L}(\mathbf{x}) = 5000 f_{\rm loss}(\mathbf{x})$). Per Eq. 32, this implies an effective number of dimensions of $D_{\rm eff} \sim 42$ for the network. To place step sizes for HMC and ray tracing on equivalent footing, we calculate an effective timestep size, $\Delta t \equiv \Delta s/\sqrt{D}$, for ray tracing, corresponding to the distance traveled divided by the mean HMC exploration speed.

We find, as expected from past work (Izmailov et al. 2021), that sampler mixing is generally excellent in function space and is poor in weight space for both methods. Although both samplers converged to equivalent function space distributions, the parameter norm $|\mathbf{x}|$ increased without bound for both HMC and ray tracing. We also found that the likelihood function in weight space was not well-approximated by a Gaussian distribution. Besides the fact that the parameter norm continued to increase indefinitely, a Gaussian weight-space distribution would have led to helical orbits (Section 2.10); instead, trajectories showed much more complex likelihood geometries (Fig. 10, left panel). For our main analysis, we used the same hyperparameters for both samplers (step size $\Delta t = 2 \times 10^{-4}$ and momentum refresh rate $f = 5\Delta t$), which resulted in somewhat longer autocorrelation times for HMC, as shown in Fig. 10.

We found that ray tracing and HMC gave similar loss distributions (Fig. 11, top-left panel). Marginal distributions appear similar between HMC and ray tracing for model predictions on individual halos in the validation set (Fig. 11). We found that both HMC and ray tracing showed steep dependence of acceptance rates on step sizes, as expected from Fig. 7—while acceptance rates were $\sim 98 - 99\%$ with the step sizes above, increasing them by a factor of 2 invariably resulted in one or more walkers getting stuck for a significant fraction of its path length, leading to a clear spike in the posterior distributions for the validation set.

For comparison, the Adam optimizer (Kingma & Ba 2014) had lower average losses and tighter posterior prediction distributions (Fig. 11)—as expected, since we chose the loss tolerance to be nonzero. At the same time, we note that Adam typically explored in a tight region around the initial position ($\langle|\mathbf{x}_{\rm final} - \mathbf{x}_{\rm initial}|/|\langle\mathbf{x}_{\rm initial}|\rangle \sim 0.3$), whereas the samplers were able to explore locations that were very different from the initial positions ($\langle|\mathbf{x}_{\rm final} - \mathbf{x}_{\rm initial}|/|\langle\mathbf{x}_{\rm initial}|\rangle \sim 3.8$ for HMC and ~ 3.6 for ray tracing). This may be one reason why the posterior uncertainties of HMC and ray tracing are broader than that expected by inflating the Adam posteriors according to $\Delta f_{\rm loss}$. For example, in Fig. 11, the posteriors for HMC and ray tracing show a 50% broader range than the Adam posterior for the first halo in the validation set, whereas the 10% increase in MSE would imply a broader range of 5% on average. Section 4.2 contains a broader discussion on the relative merits of deep ensembles vs. Bayesian neural networks, and when the choice of each is appropriate.

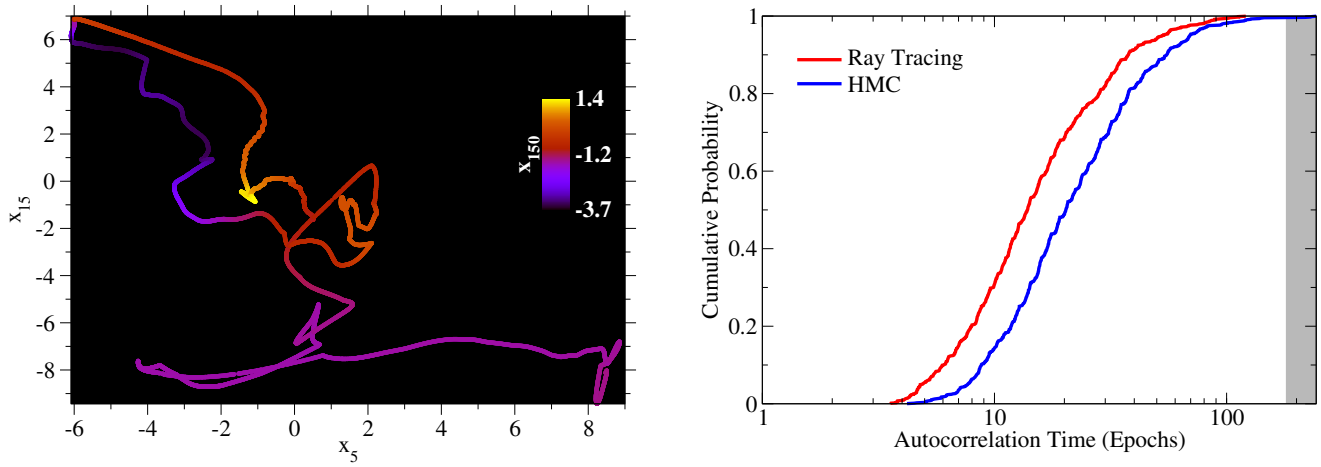

FIG. 10.— **Left**: A 3D projection of a sample trajectory in weight space from ray tracing in a 1433-dimensional neural network (a multi-layer perceptron) in Section 3.2; dimensions 5, 15, and 150 are shown. The path resembles a tangled piece of spaghetti, but the path never crosses itself, due to the many dimensions available for exploration. Frequent twists and turns are evident, indicating that the likelihood space is not well-approximated by any simple shape, as assumed by, e.g., variational inference. For this sample trajectory, a small step size of $\Delta t = 2 \times 10^{-5}$ was used (10× smaller than our main analysis) with no momentum refresh to show the geometry of the target distribution as accurately as possible. **Right**: Stochastic HMC had longer autocorrelation times than ray tracing for the same multi-layer perceptron application. This figure shows the cumulative distribution of autocorrelation times for validation set outputs (expressed in units of training epochs) for both HMC and ray tracing. The grey shaded region shows where the chain length is insufficient to guarantee convergence. Ray tracing achieved 100% convergence, and HMC achieved 99.6% convergence across validation outputs. In both cases, the first 100 epochs were discarded as burn-in.

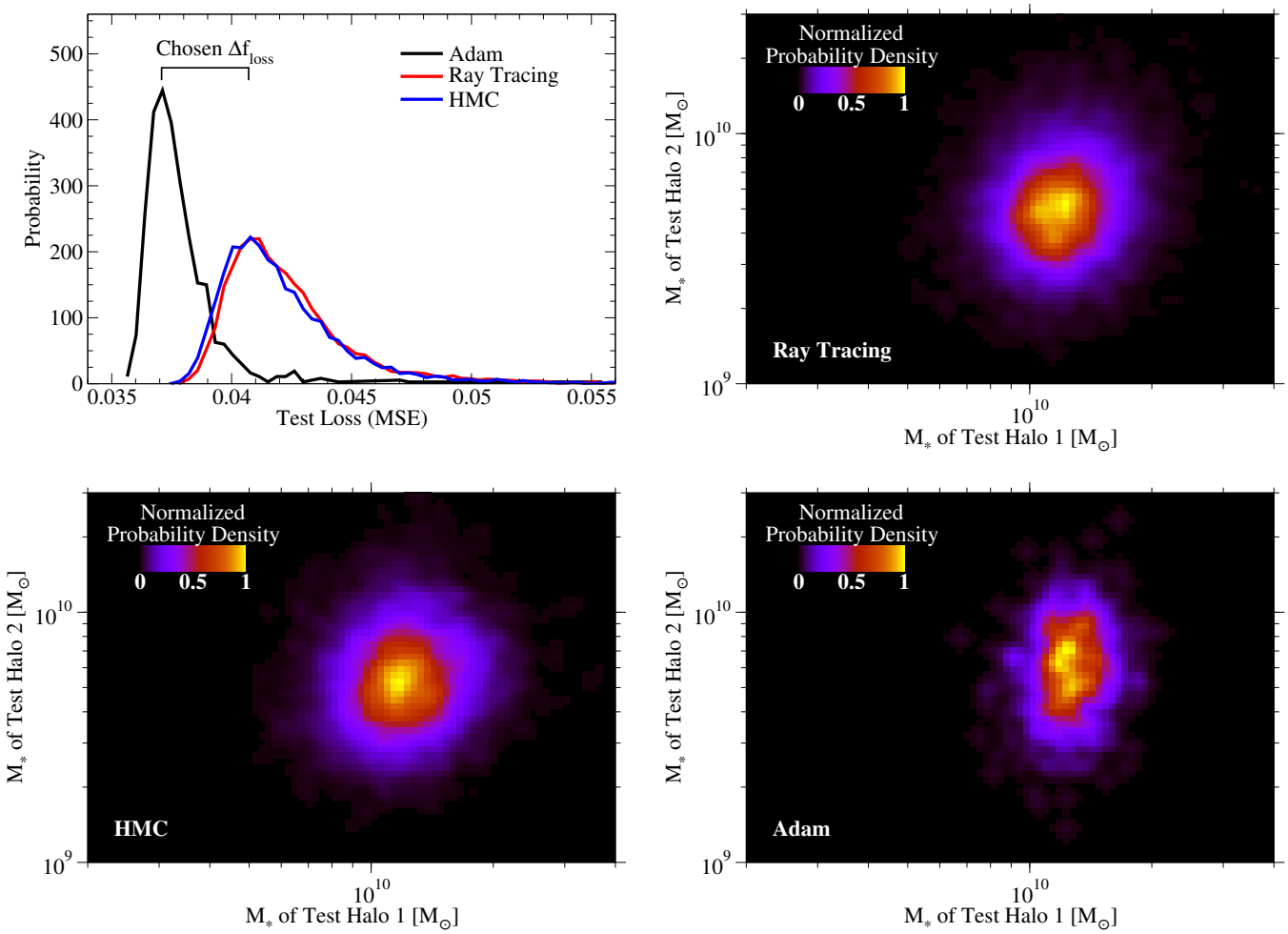

FIG. 11.— **Top left**: distributions of the loss function (mean-squared error; MSE) on the validation set for the multi-layer perceptron application in Section 3.2. Adam has the lowest loss, as it searches for loss minima. Ray tracing and HMC both sample from the posterior distribution of models within a given tolerance of the minimum loss (here chosen to be $\Delta f_{\rm loss} = 0.0042$, or $\sim 10\%$ of typical MSE), and so have higher loss values. **Top right**, **bottom panels**: distribution of model predictions for two halos in the validation set, for ray tracing, HMC, and Adam. HMC and ray tracing were run with identical parameters (i.e., mean step sizes and momentum refresh rates).

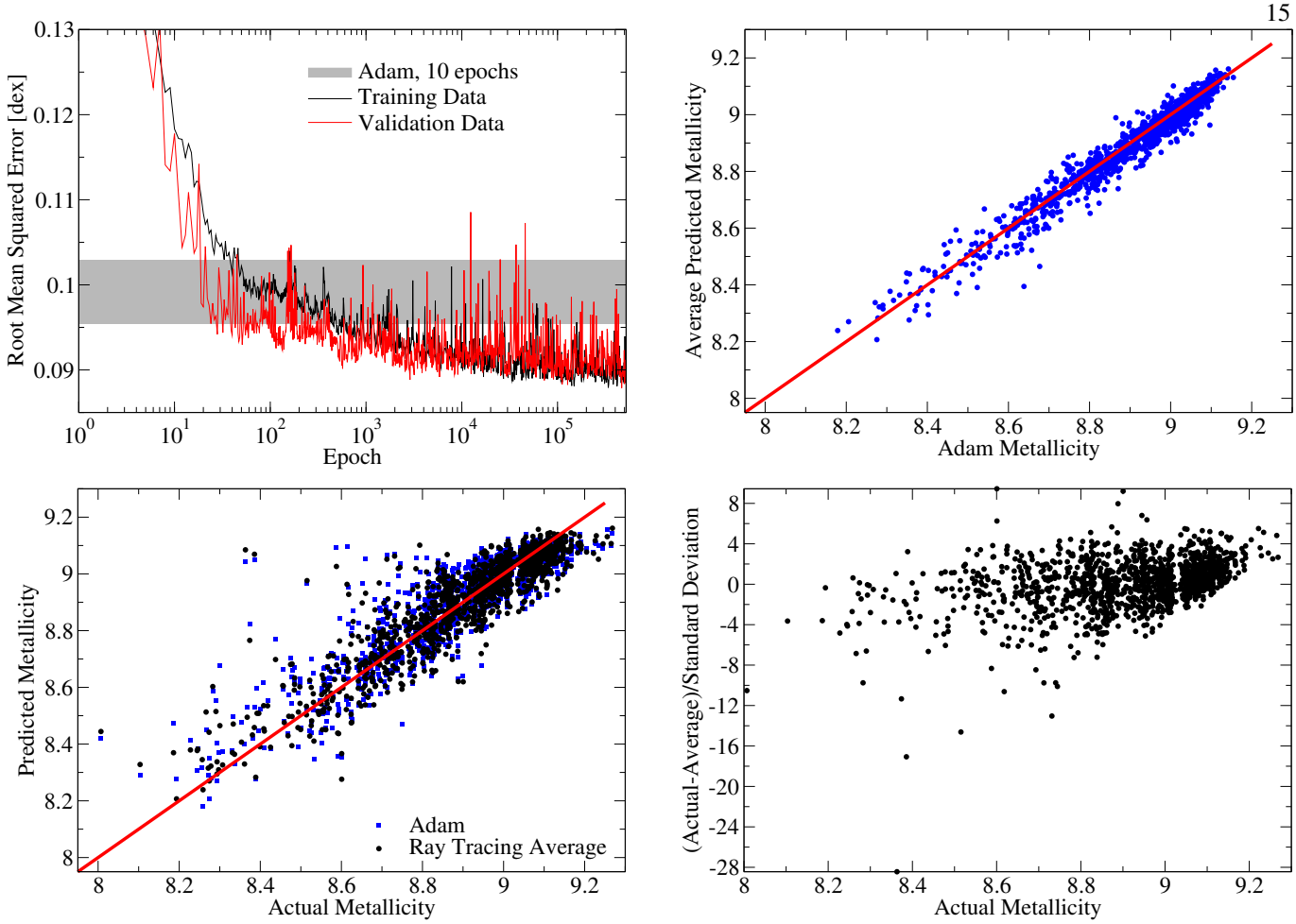

FIG. 12.— **Top left**: Root mean square error (RMSE) for training vs. validation data, using ray tracing on a ResNet-34 network with 22 million free parameters and a training data input size of ~12,000 (Section 3.3). Even after sampling for >500,000 epochs, training and validation loss are similar, suggesting resilience to overfitting. The grey shaded region shows the 68% range of Adam predictions after 10 epochs of training (to match the Wu & Boada 2019 methodology). **Top right**: Adam returns similar overall metallicity estimates as MCMC sampling on the validation data. **Bottom left**: Average predicted metallicity for ray tracing and predicted metallicity for Adam vs. true metallicity for the galaxies in the validation data set. Several outliers are notable, as also found in Wu & Boada (2019); as expected from the top-right panel, Adam and ray tracing result in similar outliers. **Bottom right**: Typical offsets between true metallicities and average predicted metallicities are well outside the sample standard deviation. This is expected (and good) behavior: it indicates that the uncertainty in the best prediction is much less than the variance of the data around the best fitting manifold, so that increasing the training sample size is less likely to lead to improved predictions (see text). Typical posterior distribution widths are ~0.02 dex for network predictions, compared to an RMSE of 0.09 dex for the distribution of actual metallicity errors. All gas-phase oxygen metallicities are reported in standard log units ($12 + \log_{10}(O/H)$).

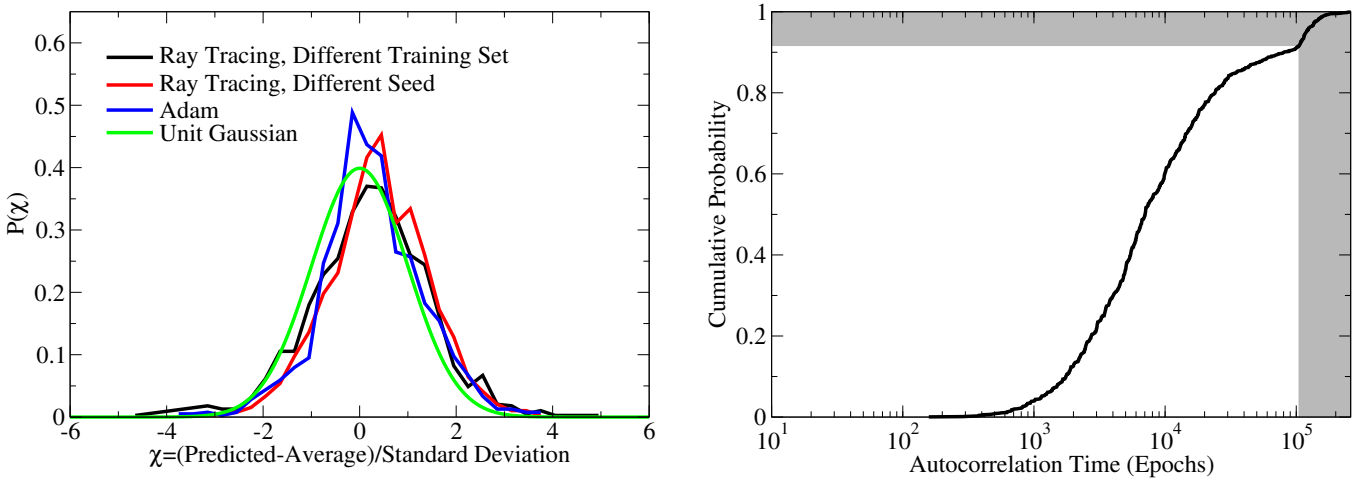

FIG. 13.— **Left**: The distribution of standardized differences (i.e., difference in predictions divided by the posterior standard deviation) of galaxies in the validation set for 1) a model generated using 5000 epochs of ray tracing on an identically-sized but non-overlapping training set drawn from the same data distribution, 2) a model generated using 5000 epochs of ray tracing using the same training set but a different random starting position, and 3) the original ResNet-34 weights after 10 epochs of Adam. All are essentially similar to a unit Gaussian distribution, consistent with the interpretation that the Bayesian network posterior is the uncertainty in the best prediction to make given the input data. **Right**: Autocorrelation time distribution for output metallicities for galaxies in the validation set. The gray shaded region indicates where autocorrelation times are long relative to sampling time, and so represent unconverged results. Roughly 90% of the validation galaxies have converged metallicity predictions, with about 10% of the validation galaxies showing unconverged results (similar to the ResNet-20 results in Izmailov et al. 2021 for images in CIFAR-10).

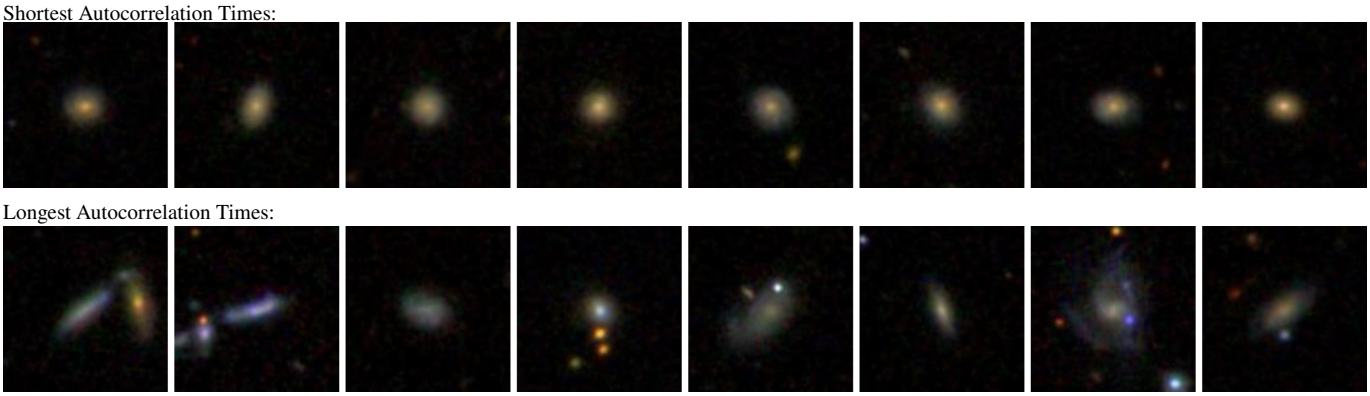

FIG. 14.— Examples of galaxies with the shortest short (top row) autocorrelation times (<100 epochs) and the longest (bottom row) autocorrelation times (>10^5 epochs). Galaxies with long autocorrelation times are typically outliers in some sense: they often have unusual colors, mergers, or contamination with stars, leading to unconstrained metallicities. Colors shown are from SDSS *gri*-band images.

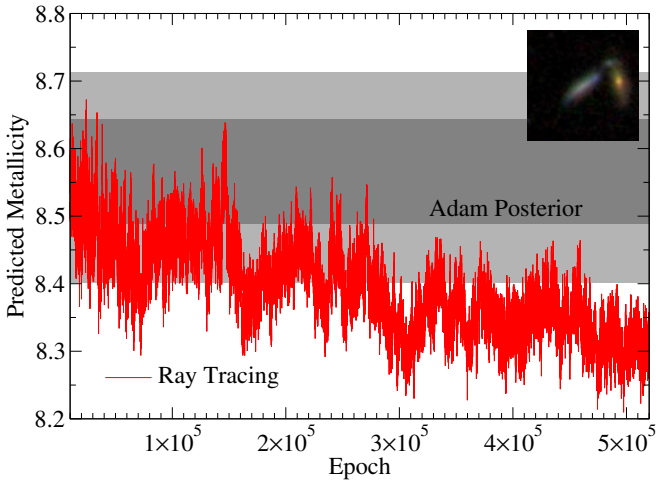

FIG. 15.— The sequence of predicted metallicities for a galaxy in the validation set (shown in the center of the inset) with an exceptionally long autocorrelation time. In practice, the metallicity of the galaxy is unconstrained given the training set and the model architecture, and so the samples do not converge to any fixed distribution over the fraction of the posterior set that has been sampled. Of note, the distribution returned by Adam (here shown as the 68% [*dark grey*] and 95% [*light grey*] confidence intervals for 1,000 independent runs) does not capture the true posterior because Adam—like any optimizer—must be run for a limited number of epochs to avoid overfitting.

3.3. *Stochastic Ray Tracing for a Convolutional Neural Network with ~22 Million Parameters*

A second common scientific application is re-using an existing neural network and transferring it to a new problem after fine-tuning. Here, we repeat a simplified version of the method in Wu & Boada (2019), who used a convolutional neural network (ResNet-34; He et al. 2016) to predict spectroscopic gas-phase metallicities (i.e., the fractional occurrence of heavy elements relative to hydrogen) for galaxies from input 3-color *gri*-band photometric images from the Sloan Digital Sky Survey Data Release 14 (Abolfathi et al. 2018).

Wu & Boada (2019) found that they could estimate metallicities from galaxy images with a relative uncertainty of ~22% (~0.085 dex) compared to the true values. A natural scientific question is whether the success of their method (and the biases inherent in it) are inherited from the underlying ResNet-34 pre-training, or whether it came from the overall architecture of the network.

Because ResNet-34 networks have ~22 million parameters, and Wu & Boada (2019) used only ~10^5 training images, prolonged training would result in overfitting. However, with an appropriate MCMC sampler, overfitting is not a problem. Networks that contain more information (e.g., networks that overfit to a given dataset) have lower entropy and thus occupy a much smaller volume in parameter space (see also the discussion in Section 4). Since learning the underlying manifold (e.g., features in an image that correlate with metallicity) requires dramatically less information than memorizing input–output pairs, the expected likelihood of an overfitted network is vanishingly small. The caveat to this intuition is that, in many problems, the appropriate likelihood function may not be known *a priori* (e.g., if $D_{\rm eff}$ is not known), and so has to be found by trial and error—e.g., by comparing the training loss to the validation loss, as is standard practice.

Full details of the network architecture, hyperparameters, and hardware are presented in Appendix E.2. We found that it was straightforward to perform approximate MCMC sampling with ray tracing on a single consumer GPU following the recipe in Section 2.12, but due to the small mini-batch size (32 images) used, the gradient variance was too high to attempt to sample with HMC. As expected, training with Adam indefinitely led to overfitting; given that there were many more network parameters than training set images, we expect that the "best" achievable training loss would be a perfect memorization of the training set metallicities ($f_{\rm loss} = 0$). Following the methodology in Wu & Boada (2019), we hence truncated Adam fits after 10 epochs.

Using ray tracing, we could achieve a typical loss similar to or better than the best 10-epoch Adam fits using a loss scaling of 4×10^4, corresponding to $\Delta f_{\rm loss} = 0.18$ (measured empirically during sampling) and implying an effective number of parameters of $D_{\rm eff}$ ~14,000, similar to the number of images in our training set (~12,000). Results for training vs. validation loss are shown in the top-left panel of Fig. 12. Aside from the fact that we are performing MCMC on a 22-million parameter neural network, such a plot would be unremarkable. As expected from the intuition that overfitted networks should occupy vanishingly small regions of parameter space, we were able to sample for >500,000 epochs without overfitting, with similar validation and training loss throughout.

Typical scatter between the Adam results and the average of ray tracing sampling was small (5%, or ~0.02 dex; Fig. 12, top-right panel) relative to overall network error (22%, or ~0.09 dex for both Adam and ray tracing; Fig. 12, bottom-left panel). This suggests that starting Adam optimization from

the ResNet-34 pretrained seed (as in Wu & Boada 2019) did not significantly lower the errors, but it also did not bias the results.

In general, the standard deviations in predicted metallicities across the MCMC posterior are small, of order 5% (0.02 dex). As with the comparison to Adam, this is much smaller than the RMSE on the training and the validation sets (~ 0.09 dex; see Fig. 12, bottom-right panel), which is desired behavior. This is straightforward to understand. To minimize the RMSE, the network attempts to estimate $\langle Z(I)\rangle$, i.e., the average metallicity Z for a given image I. Even though there is scatter in Z at fixed I (e.g., from spectroscopic measurement error), the uncertainty in the average metallicity $\langle Z(I)\rangle$ will go to zero in the limit of infinite training data. More generally, we would like the uncertainty in predictions to be dominated by metallicity scatter at fixed I (irreducible/aleatoric uncertainty in Z for a fixed image I) as opposed to the uncertainty in the average metallicity $\langle Z(I)\rangle$ (epistemic uncertainty, reducible with more training data). Hence, small posterior uncertainties compared to overall RMSE is desired behavior that indicates that we have sufficient training data for most galaxies.

To confirm that our interpretation is correct (i.e., that $\langle Z(I)\rangle$ is well-constrained by the training dataset), we compared predictions from our main run to ray tracing with 1) a different starting seed and 2) a different identically-sized but non-overlapping training set, both after 5000 epochs of training. We also compared the predictions to a deep ensemble of 1,000 Adam runs trained for 10 epochs each starting from the original ResNet-34 weights. The distribution in the predictions from the main run compared to all three of these alternate predictions is essentially a unit Gaussian (Fig. 13, left panel), with (small) deviations due to correlated errors in the shape of the best-fitting manifold for $\langle Z(I)\rangle$.

We show the distribution of autocorrelation times for the metallicities of galaxies in the validation set in Fig. 13, right panel. We find that ~91% of the galaxies in our validation set have autocorrelation times short enough to be well-sampled, and 9% do not. This percentage matches past efforts to sample from ResNet-type networks. For example, Izmailov et al. (2021) also found 90% convergence using HMC for Bayesian neural network posteriors on validation set images from CIFAR-10.[4] In comparison, Izmailov et al. (2021) used a smaller network (ResNet-20, with 270,000 parameters instead of >22 million in our ResNet-34 network), smoother activation functions (SiLU, instead of ReLU in our network), and more processing power (512 GPUs instead of our single GPU).

It may be tempting to ask why convergence is not achieved for all galaxies in the validation set. However, the correct question is really the opposite: with 22 million network parameters and only ~12,000 training set images, it is surprising that convergence is obtained for *any* of the galaxies in the validation set, given how unconstrained most network parameters are. Some resolution to both questions can be found in Fig. 14, which shows galaxies in the validation set with short autocorrelation times (top row) and long autocorrelation times (bottom row). Visual examination suggests that the galaxies with short autocorrelation times are relatively homogeneous, common galaxies, so that many examples of systems on the same manifold exist in the training set, and it is plausible that the network samples give reasonable uncertainties for their metallicities. In contrast, the galaxies with long autocorrelation times have unusual colors, mergers, and extensive contamination with stars, leading to fewer corresponding reference points in the training set from which a metallicity could be inferred. These aspects could be remedied by targeted augmentation of the training set and image precleaning, respectively; however, these aspects are beyond the scope of this paper to address.

The fact remains that, when a given input does not have many comparable examples in the training set, and when the model is underfitted, the model output is likely to be unconstrained. As an example, we show the galaxy with the longest autocorrelation time in Fig. 15, for which the network output has not converged even after 500,000 epochs. For this galaxy, the distribution of predicted metallicities from our deep ensemble of Adam-trained networks does not encompass the range from ray tracing, as Adam runs must be terminated early to avoid overfitting. Indeed, the epistemic uncertainty in predicted metallicities for this galaxy (>0.2 dex) substantially exceeds both the typical value for other galaxies (0.02 dex) and the typical RMSE in metallicity predictions (0.09 dex).

3.4. *Stochastic Ray Tracing on a Large Language Model with 1.5 Billion Parameters*

The frontier of large neural networks has been in transformer-based networks (Vaswani et al. 2017) including large language models (LLMs), which now reach above 1 trillion parameters (e.g., Du et al. 2021; OpenAI 2023). Large language models are seeing increasing usage in astronomy and other scientific fields (e.g., Fouesneau et al. 2024; Ting & O'Briain 2025; Zheng et al. 2025). As access to the GPU clusters required to train such models is not widespread, we demonstrate instead that the ray tracing algorithm scales well to the 1.5 billion-parameter GPT-2 LLM (Radford et al. 2019), which is one of the largest models that can be trained on consumer hardware. Text prediction networks, like GPT-2 and later variants, operate by predicting the next token (i.e., text segment) in a passage given the passage's previous $N_{\rm context}$ tokens (1024, for GPT-2). The exact meaning of "token" depends on the LLM, but for GPT-2, tokens are largely complete words and morphemes (i.e., fragments of words that carry meaning, such as "con" or "ing").

In this section, we perform approximate Bayesian sampling on GPT-2. As usual, full details of the hardware and network architecture are in Appendix E.3. Although the GPT-2 weights and network geometry are public, the training data are not, so we use a 100-billion token subset of the `fineweb-edu` training set (Penedo et al. 2024, as implemented in `llm.c`[5]). We begin from a public pretrained network with the GPT-2 architecture[6] on the `fineweb-edu` dataset. For the loss, we use the standard cross-entropy metric, defined as:

$$f_{\rm loss}(\mathbf{x}) = \langle -\ln(p_{\mathbf{x}}(y|C))\rangle, \tag{36}$$

where the average is taken over all output tokens y in the training set, with $p_{\mathbf{x}}(y|C)$ the probability that a given model with parameter values $\mathbf{x}$ will return the correct token y given the preceding context tokens C. We adopted a likelihood of $\mathcal{L}(\mathbf{x}) = 4\times10^9 \cdot f_{\rm loss}(\mathbf{x})$, corresponding to $\Delta f_{\rm loss} = 0.1875$ (assuming $D_{\rm eff} = 1.5\times10^9$). Intuitively, this lets us explore models that have an accuracy within about $\exp(\Delta f_{\rm loss}) \sim 20\%$ of that of the seed model (which we label $\mathbf{x}_0$). We adopt the burn-in time to be the point at which our target loss ($f_{\rm loss}(\mathbf{x}_0)+$

[4] https://www.cs.toronto.edu/~kriz/cifar.html

[5] https://github.com/karpathy/llm.c

[6] https://huggingface.co/karpathy/gpt2_1558M_final3_hf

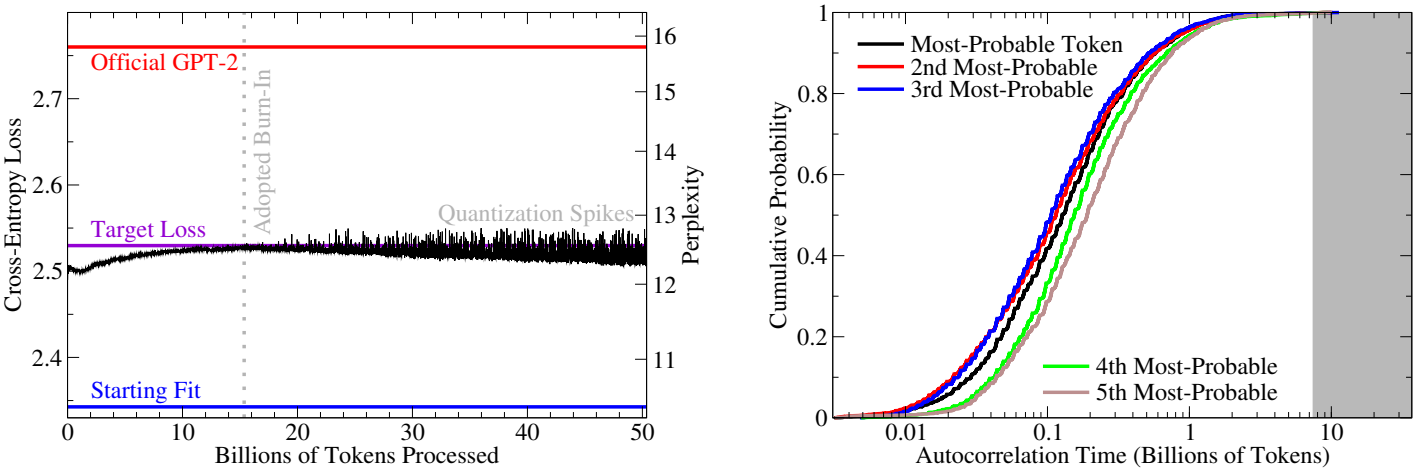

FIG. 16.— **Left**: Validation loss (cross-entropy) as a function of total tokens processed. For reference, perplexity (i.e., the exponential of cross-entropy loss) is also shown. We chose a target loss (~2.53; purple line) in between the official GPT-2 weights applied to our training set (red line) and the initial trained seed (blue line), corresponding to $\mathcal{L}(\mathbf{x}) = 4 \times 10^9 f_{\rm loss}(\mathbf{x})$, and used a step size of 5 (corresponding to $5 \times 1024 = 5120$ tokens). After long training times, we see quantization spikes—a sharp excursion to high validation loss, followed by an immediate return to the target loss—caused by using BF16 (16-bit) floating point values for parameters. **Right**: Cumulative distributions for autocorrelation times of model predictions for the validation set. For each of the first 5,120 tokens in the validation set, we recorded the probabilities of the top five predicted next tokens, and computed the autocorrelation times of these probabilities as shown above. Median autocorrelation times for GPT-2 token predictions with ray tracing are around 10^8 tokens.

Validation text example:

The Research Debate
It is sometimes claimed that research does not support the efficacy of bilingual education. Its **harsh**est critics, however (e.g., Rossell & Baker, 1996), do not claim that bilingual education does not work; instead, they claim there is little evidence that it is superior to all-English programs. Nevertheless, the **evidence** used against bilingual education is not convincing.
One major problem is in labeling. Several **critics**, for example, have claimed that English immersion programs in El Paso and McAllen, Texas, were shown to be superior to bilingual education. In each case, however, programs labeled immersion were really **bilingual** education, with a substantial part of the day taught in the primary language.

Color Key: Blue: high entropy; Purple: long autocorrelation time; **Bold Red: long autocorrelation time and high entropy**.

FIG. 17.— Bayesian exploration of parameters for GPT-2 allows coloring validation text by both entropy and autocorrelation time. Blue: **high entropy** (> 4) but short autocorrelation (colored text is unpredictable given training set data); Purple: **long autocorrelation** (> 500M tokens) and low entropy (colored text is predictable, but the network struggles to learn the correct probability distribution); **Bold Red**: **long autocorrelation** and **high entropy** (colored text is unpredictable, and the network struggles to learn the correct probability distribution). Existing optimization methods can tell which words are high-entropy, but only Bayesian methods (such as ray tracing) have access to autocorrelation times, which indicates where information flows inefficiently into and out of the network. There is relatively little overlap between words with long autocorrelation times and words with high entropy. The validation text shown is from the `fineweb-edu` dataset, using text originally from Krashen (2000).

$\Delta f_{\rm loss} \sim 2.53$) is reached, which occurs after roughly 3 million steps (~15 billion tokens).

As shown in the left panel of Fig. 16, we can sample from the posterior distribution of GPT-2, maintaining roughly constant validation loss, confirming that even the small mini-batch size used ($5 \times N_{\rm context}$) does not lead to runaway trajectory heating. We do find that the validation cross-entropy loss shows brief spikes to values as high as 20 for a single step, followed by an immediate return to the typical loss (~2.53). Further tests showed that these spikes occur even when the step size is reduced (e.g., by a factor 5, which would be expected to result in ~ 25× less heating). We hence attribute these spikes to the quantization of the network, which uses a 16-bit floating point format (BF16) instead of the 32-bit format used for the other networks in this paper. We adopt the masking approach in Section 2.8, effectively skipping the 4% of samples where the validation loss exceeds 2.55 (corresponding to our target loss plus twice the expected uncertainty in the validation loss).

Along with the quantization spikes, we also note that the loss shows a very gradual decline below the target value, such that the final model in the chain has a 2% better chance of predicting tokens in the validation set than the first model where the target loss is reached. Qualitatively, this suggests that the initial seed $\mathbf{x}_0$ was not at the global minimum, and so the network as a whole is slowly migrating towards a different likelihood basin around a better minimum. We note, nonetheless, that we are interested here in an *approximate* sampling of the posterior distribution—and that networks that perform within 2% of each other are substantially interchangeable in practice.

The right panel of Fig. 16 shows the autocorrelation times for a subsample of the network predictions on the validation set. Specifically, for each token in the first mini-batch of the validation set (5120 tokens total), we sample the predicted probabilities for the five most-probable next tokens. This gives a 5×5120-dimensional projection of the network's full output. We report the autocorrelation times for each dimension in units

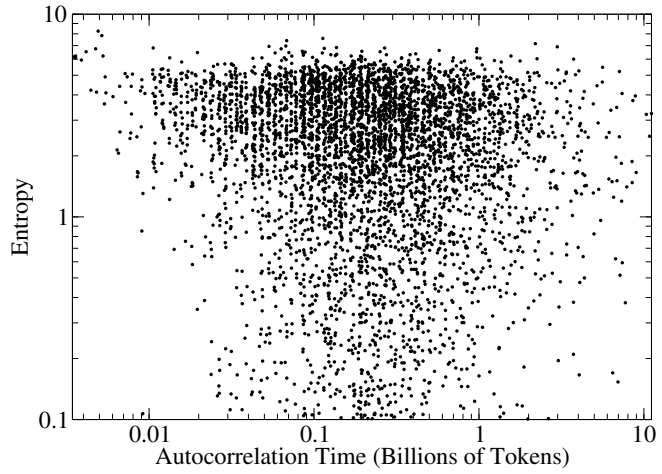

FIG. 18.— Autocorrelation time and entropy are uncorrelated across our validation dataset. This figure shows entropy vs. autocorrelation time for the most-probable next token, in units of billions of tokens; there are roughly 2×10^5 steps per billion tokens. More efficient network architectures than GPT-2 would result in more efficient information flow into the network for a given amount of tokens processed, which would manifest as lower autocorrelation times.

Prompt: "*A patient recently returned from an overseas trip and is suffering from fever, headache, nausea, joint pain, fatigue, coughing, and abdominal pain. The patient should be diagnosed with* "

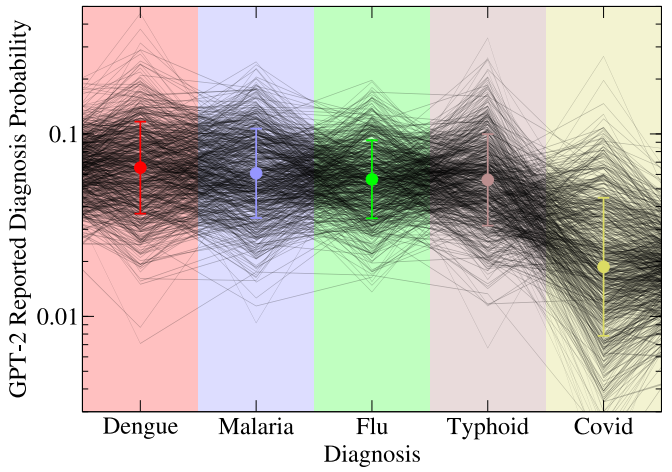

FIG. 19.— Bayesian sampling allows quantifying the uncertainty associated with a given input data vector, in this case, a query about a patient's diagnosis given a list of symptoms (shown at the top of the figure). This figure shows the distribution of answers across different diagnoses (i.e., the GPT-2 Reported Diagnosis Probability, or $P(\text{diagnosis}|\text{model,symptoms})$). Data points and error bars show the averages and standard deviations across all sampled models, and grey lines show individual model realizations. The symptoms in the prompt were drawn from the Mayo Clinic page for typical symptoms of malaria. Different neural networks gave a wide variety of answers, with malaria and four other illnesses with overlapping symptoms among the most common responses. Although 23% of networks gave malaria as the most preferred diagnosis (i.e., more likely to be a response than any other illness), most of the sampled networks gave another preferred diagnosis. Hence, trusting the likelihoods from a single realization (even one trained on the large number of medical journal papers in the `fineweb-edu` dataset), may mislead practitioners away from the correct underlying distribution of diagnoses.

of billions of tokens, i.e., how many tokens are required to be processed through the ray tracing algorithm before the dimension reaches an independent sample. Fig. 16 shows that the median autocorrelation time for predictions of GPT-2 is around 10^8 tokens. For comparison, this is about the same number of words as in 1,000 books, or about the expected number of books read by a single adult in their lifetimes.[7] However, there is a long tail to very high token counts ($\gg10^9$), which suggests that 1) some concepts are rarely mentioned in the training set, and/or 2) the network is too inflexible to adapt easily to new interpretations and requires repeated exposures to the same information (see also Section 4.3). The algorithm reaches (with the caveat above that we are performing approximate sampling) greater than 99.8% convergence across the dimensions probed. This is substantially higher than for the ResNet-34 network in the previous section because our training set size is much larger than the network parameter count.

One key finding of this analysis is that entropy and autocorrelation times are uncorrelated. This is shown in the example validation set text in Fig. 17 as well as more quantitatively in Fig. 18. Here, we define entropy in the usual sense as $H(C,\mathbf{x}) = -\langle\ln(P(T|C,\mathbf{x}))\rangle$—i.e., the negative average log-probability for the next token T at a given model location $\mathbf{x}$ and for given context tokens C. Large entropy values mean that there could be many possibilities for the next token T, whereas low entropy values mean that there are relatively few possibilities. For a human text-predictor, high entropy means either 1) the human has not been exposed to a topic, and so has no basis to know what may come next, 2) there are many possible directions to take the phrase, and/or 3) there are many possible word choices for the same content. For a machine predictor, the first explanation does not apply (due to exceptionally large training sets), and so entropy is essentially synonymous with ambiguity, which depends on the specific content and language (English, in this case). Autocorrelation times, in contrast, are a measure of information flow rates into and out of the network, which depend on network structure, parameter count, and the distribution of content topics in the training set. As a result, it is plausible that the two concepts measure orthogonal quantities.

Returning to more applied topics, large language models are routinely used to answer queries across scientific fields. An open question is how uncertain the responses from large language models are, which is relevant to how much trust should be placed in their responses. This is especially critical for fields such as medical diagnostics (see Zhou et al. 2025, for a review).

Bayesian sampling provides a way to answer this question, so we perform a simple experiment. We sample the approximate posterior distribution of GPT-2 networks, and then ask each sample network to complete a sentence asking for a diagnosis for patient symptoms. For the patient symptom list, we use symptom examples from the Mayo Clinic page for typical symptoms of malaria.[8] As each sample network outputs a range of potential diagnoses, we draw 10^4 samples from each network, and classified the diagnoses by disease type. This enables us to extract an uncertainty on the reported network likelihoods. As an example, a single network might place the chance at 10% of the time that the patient has malaria—but dif-

[7] https://www.pewresearch.org/internet/2016/09/01/book-reading-2016-appendix-a/

[8] https://www.mayoclinic.org/diseases-conditions/malaria/symptoms-causes/syc-20351184

ferent networks give a *range* of responses for what this chance should be, from 3% to 20%. By analogy, this experiment would be similar to asking multiple doctors for their opinions on what a patient's underlying disease most likely is.

As shown in Fig. 19, we found that different networks (like different doctors) preferred different diseases, and gave a wide range of estimated confidences (standard deviations of 0.2–0.4 dex). This is a toy example, in that the networks were trained on medical text in the `fineweb` data set, as opposed to actual patient symptom—diagnosis pairs. However, it shows that, at the scale of GPT-2, individual network prediction likelihoods can be widely variable.

We repeat, as above, that this example shows only approximate sampling of GPT-2 on a single consumer-level GPU, which is much less powerful than the hardware used to train GPT-2 (or, in fact, any more modern LLM). This suggests the hypothesis that, in general, the hardware used to train a given LLM will also be sufficient to perform approximate Bayesian sampling of its posterior distribution, which will be tested by future research.

4. DISCUSSION

4.1. *Pedagogy on the Meaning of Bayesian Sampling*

We feel that a key impact of ray tracing will be enabling widespread approximate Bayesian sampling for neural networks, having demonstrated that sampling with ray tracing is possible for problems with more than a billion parameters. Since not all readers will have previously done Bayesian sampling of neural networks, some pedagogy may be helpful.

At a basic level, the meaning of Bayesian networks depends on the network outputs. In many past scientific papers, neural networks have been used to predict a single "best" output prediction given the input data. In these cases, Bayesian sampling will result in the distribution of possible "best" predictions reflecting the underlying dataset. In the limit of infinite training data, the predicted distribution will converge to an infinitesimally thin manifold, since there is only a single description of the training data that minimizes the loss. This is similar to the case of linear regression—in the limit of infinite data, there is no uncertainty in the location of the line that best fits the data. Hence, the sampled predictions for single-output networks should not be interpreted as the distribution of potential outcomes, but instead as the uncertainty in the *average value* of the potential outcomes for similar inputs (see also Section 3.3).

When, the distribution of potential outcomes is desired instead, this requires building and training a network that generates a probability distribution for its output. Such networks are also known as neural posterior estimators, with common examples including normalizing flows, diffusion models, and large language models. In these cases, Bayesian sampling will give the uncertainty in the posterior distributions, as shown in, e.g., Fig. 19. This is most helpful in cases where wrong-but-confident outputs from a single model would cause problems, as Bayesian sampling would then reveal the true diversity of posterior estimates consistent with the training data.

4.2. *Sampling Temperature: Deep Ensembles vs. Samplers*

The appropriate temperature for sampling and uncertainty quantification has led to some disagreement in the literature over the best approach. Deep ensembles (e.g. Lakshminarayanan et al. 2016) are effectively zero-temperature sampling, while Wenzel et al. (2020) argue that "cold posteriors" (temperatures $\tau < 1$) are preferred and Izmailov et al. (2021) argue for normal Bayesian posteriors ($\tau = 1$). We advocate that $\tau = 1$ is the most defensible answer, but note that the correct likelihood function changes depending on the desired outcome.

It's helpful to consider a concrete example. For a typical supervised learning task to predict a single output value with mean-squared error loss, the log-likelihood might be:

$$\ln(\mathcal{L}(\mathbf{x})) = -\sum_i \frac{(\mathbf{f}_\mathbf{x}(\mathbf{y}_i) - \mathbf{v}_i)^2}{2\sigma_i^2} + G(\mathbf{x}), \tag{37}$$

where $\mathbf{y}_i$ are the input training data, $(\mathbf{f}_\mathbf{x}(\mathbf{y}_i))$ are the network outputs, $\mathbf{v}_i$ are the training labels, σ_i are the errors in the training labels, and $G(\mathbf{x})$ is the (unknown) generalization loss—i.e., the loss that would be present if the training set included all possible samples with the correct weights.

For many supervised learning tasks, there is no label uncertainty ($\sigma_i = 0$), and the log-likelihood is infinitely negative. Hence, sampling in a proper Bayesian sense ($\tau = 1$) is equivalent to sampling a rescaled, finite log-likelihood (e.g., $\ln(\mathcal{L}_s(\mathbf{x})) = -\sum_i (\mathbf{f}_\mathbf{x}(\mathbf{y}_i) - \mathbf{v}_i)^2$) at zero temperature. Deep ensembles effectively use this approach, under the heuristic that the global minimum (if $G(\mathbf{x})$ were known) is likely to be contained within the range of local minima reached during training with different seeds; conformal prediction (e.g., Angelopoulos & Bates 2021) can then regularize this heuristic estimate.

However, in many scientific fields, there *is* label uncertainty (e.g., due to measurement uncertainty) that has to be taken into account. In addition, the marginal coverage guarantees of conformal prediction are often not sufficient, as scientists often need *conditional coverage*: that is, we wish to know the uncertainty for a given measurement (for example, the uncertainty in cosmological parameters for the single Universe that we can observe), as opposed to the uncertainty across all potential measurements (e.g., universes in a simulated test set). Similar needs occur in other fields, e.g., medicine and self-driving cars, where inaccurate conditional coverage can be life-threatening—that is, we wish to avoid wrong-but-confident outputs. Bayesian neural networks in the traditional sense, with $\tau = 1$, are currently the only method that provides conditional coverage while simultaneously admitting the label uncertainty necessary in many scientific fields.

4.3. *Information Flow and Improving Network Architectures*

In the limit of an arbitrarily large number of dimensions, MCMC samplers effectively explore a constant-loss surface, or equivalently, a constant-information surface. In the latter framing, during sampling, information is constantly flowing in and out of the neural network, with each new ingestion of information being accompanied by another piece of information flowing out. This gives a heuristic interpretation of why MCMC sampling results in uncertainty quantification, as it is in effect equivalent to bootstrap resampling of the training set (which can also be done with deep ensembles; see Loaiza-Ganem et al. 2025).

Information flow also provides a way to compare different network architectures. Heuristically, "better" architectures can absorb information more easily (and more easily forget/replace it when better information arrives). We could hence define a mean log information flow rate as:

$$\langle \ln \dot{I} \rangle = -\langle E \rangle - \langle \ln \tau \rangle, \tag{38}$$

where $\langle E \rangle$ is the average log-loss (e.g., the usual cross-entropy loss for a large language model), and $\langle \ln \tau \rangle$ is the average log autocorrelation time of, e.g., the network outputs for the validation set in units of bits.[9] This is very simple to calculate—e.g., for the GPT-2 model in Section 3.4, it is about -32.

Since this measure is quantitative, it can be used on-the-fly to reconfigure network architecture during training. At a basic level, a range of network architectures could be generated using a genetic algorithm, trained in parallel, and the architectures with the best information flow rates could be used as the basis of a new generation, and the process iterated, akin to natural selection of the architectures that are most resource-efficient. This method may be helpful in finding architectures that perform better in data-limited fields, including large language models and image/video generation models.

4.4. *Generalization of Bayesian Networks*

We caution that Bayesian sampling does not always imply improved generalization (see also Izmailov et al. 2021). This is because there are at least two key types of generalization that are relevant: 1) *extrapolation*, that is, generalization to inputs on the same manifold as the data but outside the training set, and 2) *problem transfer*: that is, generalization to inputs on different manifolds. Bayesian sampling plausibly gives reasonable extrapolation, though it will not magically be better than deep ensembles—the reason being that, by definition, there is no information in the training set on how the extrapolation should be done. The only advantage that Bayesian sampling may have here is that it is less prone to overfitting than optimization algorithms.

We also expect that Bayesian sampling will not magically lead to better problem transfer. This is very simple to understand: a network that does well on multiple input manifolds contains more information than a network that does well on only a single input manifold. Hence, even though both networks will perform equally well on the training sample, the volume of parameter space that contains multi-function networks will be exponentially suppressed compared to the volume that contains single-function networks.

It is not always easy to guess whether a given generalization will be extrapolation or problem transfer, since our intuition may lead us astray. For example, while we might intuit that adding noise to an image should fall into the extrapolation category, our brains have been trained since birth to recognize our surroundings in varying levels of light. In contrast, if networks have only been trained on low-noise images in which single pixels are informative, they may not be equipped to deal with the de-localized information present in noisy images. We can also give a more human example of dialects. If you were raised on American English, and a colleague told you that she thought things were sorted and Bob's your uncle, but the head boffin was a few sandwiches short of a picnic basket and then everything started going pear-shaped—that would be on a different manifold from what you were expecting English to be. Different dialect manifolds can therefore contain mutually incomprehensible content for speakers of the "same" language, and so speakers used to one manifold can be helpless to generalize to another.

The straightforward recommendation for generalization is to use a domain adaptation approach (e.g., Universal Domain Adaption, You et al. 2019), which can both test whether a new data set is distinguishable from the training data set, and can remove information from the training data set that distinguishes it from the target data set (and vice versa), so that only common information is used. Example past applications to astronomical problems have included galaxy morphology classification (Ćiprijanović et al. 2023), strong lensing (Agarwal et al. 2024), and estimating cosmological parameters (Roncoli et al. 2023).

[9] In this framing, an output that is static should be treated as having an indefinitely long autocorrelation time.

4.5. *Misalignment and Hallucinations*

Misalignment refers to a difference between training goals and what a model actually learns to do. One consequence is hallucinations, in which a model that is trained to provide helpful responses instead invents false information. Bayesian neural networks have increased chances for misalignment and hallucinations (often considered negative traits) and for creativity (often considered a positive trait), because they represent the variety of approaches necessary to perform well on the training set. At the same time, it is unlikely that different models will be misaligned in exactly the same way, unless overfitting has occurred (e.g., in deep ensembles).

We note simply that we have lived with misalignment among humans for a long time. The modern human solution for misalignment has been to avoid giving full decision-making power to any single person, implicitly expecting that different misalignments between different people will average out. In that sense, when misalignment is a concern, it may be better to average decision-making power across different samples from a Bayesian network. Similarly, we hypothesize that it is less likely that different samples will have the same hallucinations, whereas it is more likely that non-hallucinated information is shared across multiple models (see also Fig. 19). Future Bayesian exploration of large language models will test these hypotheses.

4.6. *Future Directions*

While the general problem of efficient exploration of neural network weight space remains unsolved, the techniques developed here suggest some promising directions. The basic problem is that, neural networks have a filametary structure, such that the most direct path between two points on the typical set is over a large potential barrier—i.e., locations that should never be traveled by any reasonable sampler. Hence, the basic solution to weight space exploration must involve exiting the typical set and re-entering it, ideally in a reversible fashion.

One option is a likelihood-dependent temperature (Section 2.9.5), which would enable exploration of unlikely regions. This then brings the challenge of adaptively tuning $\tau(\mathcal{L})$ to allow reasonable sampling of both the typical set and unlikely regions, as well as the challenge of efficiently estimating $\mathcal{L}$ for large training sets. A simpler alternative may be to augment the model parameter space with τ (i.e., to define $\mathcal{L}(\mathbf{x}, \tau)$), so that the sampler naturally explores the probability distributions for a range of temperatures. This solution has the appealing feature that errors in integration can be re-interpreted as changes in effective τ so as to increase acceptance rates.

A second option is to use a refractive index that is spatially varying but independent of $\mathcal{L}$. In this case, ray paths can be precomputed, and hence a search algorithm can be used to find the desired stopping point, for example $\mathcal{L}_{\rm final} = \mathcal{L}_{\rm initial}$ (Behroozi et al., in prep.). Because search algorithms have much better scaling with dimensionality ($O(\log D)$ or better), such approaches may make Bayesian sampling of neural networks even more efficient than the method considered here.

However, the challenge with this approach can be in choosing a good direction to search: if the neural network parameters are fully constrained, they will be thin tubes in a high-dimensional parameter space, and so there is no guarantee that a given search direction will be useful.

The best approach may be a hybrid of these two: broadening exploration around the typical set via tempering, and then using a search algorithm to bypass large regions of unlikely parameter space in between the broadened regions around the typical set. As above, future research will test this approach.

5. CONCLUSIONS

In this paper, we developed a gradient-based MCMC method that performs ray tracing with a spatially-varying refractive index. Our main conclusions are:

- The method performs well for sampling function spaces of neural networks, including those with more than a billion parameters, even on consumer hardware (Section 3). These results are possible because the method is extraordinarily robust to stochastic gradient errors, showing orders of magnitude lower bias than Hamiltonian Monte Carlo (HMC) at the same step size (Section 3.1.2).

- The method provides fair sampling, even for imperfect integrators, and can pass through arbitrary likelihood and parameter space barriers (Section 2). The latter ability resolves a key challenge with past methods.

- The method generalizes straightforwardly, encompassing many other existing methods (HMC, microcanonical HMC, Gibbs, stretch-step, Metropolis, etc.) with the appropriate choice of weighting (Section 2.9). This allows straightforward comparison of the relative time each method spends exploring high-probability vs. low-probability regions.

ACKNOWLEDGEMENTS

PB was funded by a Packard Fellowship, Grant #2019-69646, as well as an NSF EAGER grant, #2404989. PB thanks Oddisey Knox for asking about a homework problem involving a spatially varying refractive index, and PB thanks his father for encouraging him to pursue an idea that PB initially thought would be no more efficient than existing methods. PB also thanks Andrej Karpathy for open-sourcing the `llm.c` and `nanoGPT` implementations. Finally, PB thanks Adrian Bayer, Haley Bowden, Suchetha Cooray, Carol Cuesta-Lazaro, ChangHoon Hahn, Andrew Hearin, Ben Horowitz, Marc Huertas-Company, Christian Jespersen, Elisabeth Krause, Guilhem Lavaux, Surhud More, Radford Neal, Masahiro Takada, Leander Thiele, Yixin Wang, John Wu, and Haowen Zhang for helpful conversations and comments that improved this paper. This research was supported in part by grant NSF PHY-2309135 to the Kavli Institute for Theoretical Physics (KITP).

LARGE LANGUAGE MODEL USE DISCLOSURE

We used ChatGPT from September 2024 until paper submission to assist with code drafting and debugging; all suggestions were checked by hand for correctness. Similarly, we used ChatGPT to suggest methods to draw random samples from Rosenbrock and Cauchy distributions in Appendix D, as well as to derive the log-likelihood distribution for Cauchy distributions; as above, all derivations were checked by hand for correctness. Writefull and ChatGPT suggested improvements to the text of the paper, which were again checked by hand. Finally, as described in Section 3.4, we sampled from the posterior distributions of GPT-2 like models to predict successive tokens, to estimate autocorrelation times and epistemic uncertainties for GPT-2 like models.

REFERENCES

Abolfathi B., et al., 2018, ApJS, 235, 42
Agarwal S., Ćiprijanović A., Nord B. D., 2024, in 38th conference on Neural Information Processing Systems. (arXiv:2411.03334)
Ahmad W., Simon E., Chithrananda S., Grand G., Ramsundar B., 2022, preprint, (arXiv:2209.01712)
Angelopoulos A. N., Bates S., 2021, arXiv e-prints, p. arXiv:2107.07511
Behroozi P., Wechsler R. H., Hearin A. P., Conroy C., 2019, MNRAS, 488, 3143
Benavoli A., Wyse J., White A., 2021, arXiv e-prints, p. arXiv:2109.13891
Besag J., 1994, J. Roy. Statist. Soc. Ser. B, 56, 591
Betancourt M., 2017, arXiv e-prints, p. arXiv:1701.02434
Bierkens J., Fearnhead P., Roberts G., 2016, arXiv e-prints, p. arXiv:1607.03188
Born M., Wolf E., 1980, Principles of optics: electromagnetic theory of propagation, interference and diffraction of light. Pergamon Press
Bouchard-Côté A., Vollmer S. J., Doucet A., 2015, arXiv e-prints, p. arXiv:1510.02451
Chen T., Fox E. B., Guestrin C., 2014, arXiv e-prints, p. arXiv:1402.4102
Choudhary K., et al., 2022, npj Computational Materials, 8, 59
Chowdhery A., et al., 2022, arXiv e-prints, p. arXiv:2204.02311
Ćiprijanović A., Lewis A., Pedro K., Madireddy S., Nord B., Perdue G. N., Wild S. M., 2023, Mach. Learn. Sci. Tech., 4, 025013
Corbella A., Spencer S. E. F., Roberts G. O., 2022, arXiv e-prints, p. arXiv:2206.11410
Dinh L., Krueger D., Bengio Y., 2014, arXiv e-prints, p. arXiv:1410.8516
Du N., et al., 2021, arXiv e-prints, p. arXiv:2112.06905
Duane S., Kennedy A. D., Pendleton B. J., Roweth D., 1987, Physics Letters B, 195, 216
Earl D. J., Deem M. W., 2005, Phys. Chem. Chem. Phys., 7, 3910
Feroz F., Hobson M. P., 2008, MNRAS, 384, 449
Feroz F., Hobson M. P., Cameron E., Pettitt A. N., 2019, The Open Journal of Astrophysics, 2, 10
Foreman-Mackey D., Hogg D. W., Lang D., Goodman J., 2013, PASP, 125, 306
Forest E., Ruth R. D., 1990, Physica D: Nonlinear Phenomena, 43, 105
Fouesneau M., et al., 2024, arXiv e-prints, p. arXiv:2409.20252
Geman S., Geman D., 1984, IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-6, 721
Girolami M., Calderhead B., 2011, Journal of the Royal Statistical Society: Series B (Statistical Methodology), 73, 123
Goodman J., Weare J., 2010, Communications in Applied Mathematics and Computational Science, 5, 65
Haario H., Saksman E., Tamminen J., 2001, Bernoulli, 7, pp. 223
Hastings W. K., 1970, Biometrika, 57, 97
He K., Zhang X., Ren S., Sun J., 2015, arXiv e-prints, p. arXiv:1502.01852
He K., Zhang X., Ren S., Sun J., 2016, in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR. p. 1 (arXiv:1512.03385), doi:10.1109/CVPR.2016.90
Izmailov P., Vikram S., Hoffman M. D., Wilson A. G., 2021, arXiv e-prints, p. arXiv:2104.14421
Jimenez Rezende D., Mohamed S., 2015, arXiv e-prints, p. arXiv:1505.05770
Kingma D. P., Ba J., 2014, arXiv e-prints, p. arXiv:1412.6980
Klambauer G., Unterthiner T., Mayr A., Hochreiter S., 2017, arXiv e-prints, p. arXiv:1706.02515
Klypin A., Yepes G., Gottlöber S., Prada F., Heß S., 2016, MNRAS, 457, 4340
Krashen S., 2000, Educators for urban minorities, 1, 53
Lakshminarayanan B., Pritzel A., Blundell C., 2016, arXiv e-prints, p. arXiv:1612.01474

Li C., Sun Y., Liu J., 2016, in 2016 Progress in Electromagnetic Research Symposium (PIERS). pp 4188–4188, doi:10.1109/PIERS.2016.7735574
Lin Z., et al., 2023, Science, 379, 1123
Loaiza-Ganem G., Villecroze V., Wang Y., 2025, arXiv e-prints, p. arXiv:2501.17917
McKeon B., Goncharov A. V., 2023, Appl. Opt., 62, 8621
Melchior P., Joseph R., Sanchez J., MacCrann N., Gruen D., 2021, Nature Reviews Physics, 3, 712
Metropolis N., Rosenbluth A. W., Rosenbluth M. N., Teller A. H., Teller E., 1953, J. Chem. Phys., 21, 1087
Neal R. M., 2012, arXiv e-prints, p. arXiv:1206.1901
Ohno H., 2020, J. Opt. Soc. Am. A, 37, 411
Omelyan I., Mryglod I., Folk R., 2003, Computer Physics Communications, 151, 272
OpenAI 2023, arXiv e-prints, p. arXiv:2303.08774
Parzen E., 1962, The Annals of Mathematical Statistics, 33, 1065
Penedo G., Kydlíček H., Ben allal L., Lozhkov A., Mitchell M., Raffel C., Von Werra L., Wolf T., 2024, arXiv e-prints, p. arXiv:2406.17557
Peters E. A. J. F., de With G., 2012, Phys. Rev. E, 85, 026703
Puzyrev V., Salles T., Surma G., Elders C., 2022, Geoscience Letters, 9, 32
Radford A., Wu J., Child R., Luan D., Amodei D., Sutskever I., et al., 2019, OpenAI blog, 1, 9
Roberts G. O., Tweedie R. L., 1996, Bernoulli, 2, 341
Robnik J., De Luca G. B., Silverstein E., Seljak U., 2022, arXiv e-prints, p. arXiv:2212.08549
Robnik J., Cohn-Gordon R., Seljak U., 2025, arXiv e-prints, p. arXiv:2503.01707
Rodríguez-Puebla A., Behroozi P., Primack J., Klypin A., Lee C., Hellinger D., 2016, MNRAS, 462, 893
Roncoli A., Ćiprijanović A., Voetberg M., Villaescusa-Navarro F., Nord B., 2023, in 37th Conference on Neural Information Processing Systems. (arXiv:2311.01588)
Rosenblatt M., 1956, The Annals of Mathematical Statistics, 27, 832
Satoh T. R., 2003, in Erbacher R. F., Chen P. C., Roberts J. C., Groehn M. T., Borner K., eds, Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series Vol. 5009, Visualization and Data Analysis 2003. pp 277–285, doi:10.1117/12.473932
Skilling J., 2004, in Fischer R., Preuss R., Toussaint U. V., eds, American Institute of Physics Conference Series Vol. 735, Bayesian Inference and Maximum Entropy Methods in Science and Engineering: 24th International Workshop on Bayesian Inference and Maximum Entropy Methods in Science and Engineering. pp 395–405, doi:10.1063/1.1835238
Sokal A. D., 1996. https://api.semanticscholar.org/CorpusID:14817657
Speagle J. S., 2020, MNRAS, 493, 3132
Sriram A., Das A., Wood B. M., Goyal S., Zitnick C. L., 2022, arXiv e-prints, p. arXiv:2203.09697
Tabak E. G., Vanden-Eijnden E., 2010, Communications in Mathematical Sciences, 8, 217
Ting Y.-S., O'Briain T., 2025, arXiv e-prints, p. arXiv:2506.06921
Tjelmeland H., Hegstad B. K., 2001, Scandinavian Journal of Statistics, 28, 205
Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser L., Polosukhin I., 2017, arXiv e-prints, p. arXiv:1706.03762
Vogelsberger M., Marinacci F., Torrey P., Puchwein E., 2020, Nature Reviews Physics, 2, 42
Wenzel F., et al., 2020, arXiv e-prints, p. arXiv:2002.02405
Wong K. W. K., Gabrié M., Foreman-Mackey D., 2023, The Journal of Open Source Software, 8, 5021
Wu J. F., Boada S., 2019, MNRAS, 484, 4683
Yoshida H., 1990, Physics Letters A, 150, 262
You K., Long M., Cao Z., Wang J., Jordan M. I., 2019, in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp 2720–2729
Zheng T., Deng Z., Tsang H. T., Wang W., Bai J., Wang Z., Song Y., 2025, arXiv e-prints, p. arXiv:2505.13259
Zhou S., et al., 2025, npj Artificial Intelligence, 1, 9

APPENDIX

A. LAMBERTIAN EMISSIVITY WITHOUT FULL SURFACE NORMAL INFORMATION

Supposing a ray bundle has intersected with a differential area dA of a parameter space boundary (e.g., as in Fig. 2), we wish to choose a new direction $\hat{\mathbf{s}}$ according to a Lambertian emission distribution. As in Section 2.2, this corresponds to uniform radiance L, or equivalently, an output power $P = L\cos(\theta)dAd\Omega$ that is proportional to the cosine of the angle θ between $\hat{\mathbf{s}}$ and the surface normal vector $\hat{\mathbf{n}}$.

If we were to choose $\hat{\mathbf{s}}$ according to a random direction $\hat{\mathbf{v}}_\mathbf{1}$, this would correspond to uniform power in all directions by symmetry, $P = CdAd\Omega$, which would result in the surface not having uniform radiance (i.e., its apparent surface brightness would vary with angle as $\sec(\theta)$). Conveniently, we need only find the location of the boundary in a two-dimensional subspace to produce a random ray with uniform radiance, according to the algorithm in this section.

First, we choose a second random direction $\hat{\mathbf{v}}_\mathbf{2}$, which together with $\hat{\mathbf{v}}_\mathbf{1}$ defines a plane. We can, without loss of generality, rotate $\hat{\mathbf{v}}_\mathbf{2}$ in this plane such that it is perpendicular to $\hat{\mathbf{v}}_\mathbf{1}$. We can then decompose $\hat{\mathbf{v}}_\mathbf{1}$ and $\hat{\mathbf{v}}_\mathbf{2}$ into components parallel to the surface normal and parallel to the boundary:

$$\hat{\mathbf{v}}_\mathbf{1} \equiv c_1\hat{\mathbf{n}} + c_2\hat{\mathbf{b}}_\mathbf{1} \tag{A1}$$

$$\hat{\mathbf{v}}_\mathbf{2} \equiv c_3\hat{\mathbf{n}} + c_4\hat{\mathbf{b}}_\mathbf{2}, \tag{A2}$$

where $\hat{\mathbf{b}}_\mathbf{1}$ and $\hat{\mathbf{b}}_\mathbf{2}$ are parallel to the boundary. We can then project the surface normal vector into the $\hat{\mathbf{v}}_\mathbf{1} - \hat{\mathbf{v}}_\mathbf{2}$ plane and normalize:

$$\hat{\mathbf{n}}_\mathbf{v} \equiv \frac{\hat{\mathbf{n}}\cdot\hat{\mathbf{v}}_\mathbf{1} + \hat{\mathbf{n}}\cdot\hat{\mathbf{v}}_\mathbf{2}}{||\hat{\mathbf{n}}\cdot\hat{\mathbf{v}}_\mathbf{1} + \hat{\mathbf{n}}\cdot\hat{\mathbf{v}}_\mathbf{2}||} = \frac{c_1\hat{\mathbf{v}}_\mathbf{1} + c_3\hat{\mathbf{v}}_\mathbf{2}}{\sqrt{c_1^2 + c_3^2}} \tag{A3}$$

Hence, we find that the dot product with the unprojected surface normal is:

$$\hat{\mathbf{n}}_\mathbf{v}\cdot\hat{\mathbf{n}} = \sqrt{c_1^2 + c_3^2} \geq c_1, \tag{A4}$$

where the right-hand side satisfies the inequality $\sqrt{c_1^2 + c_3^2} \leq 1$ by construction.

The direction $\hat{\mathbf{n}}_\mathbf{v}$ is hence as or more aligned with the surface normal $\hat{\mathbf{n}}$ than the original direction $\hat{\mathbf{v}}_\mathbf{1}$. As well, we can compute $\hat{\mathbf{n}}_\mathbf{v}$ without full knowledge of $\hat{\mathbf{n}}$, because we need only determine the direction perpendicular to the boundary in the restricted $\hat{\mathbf{v}}_\mathbf{1} - \hat{\mathbf{v}}_\mathbf{2}$ plane, which is a 2D problem instead of a D-dimensional problem. Now, let ϕ be the angle between two randomly chosen vectors in $(D+1)$-dimensional space. We then choose $\hat{\mathbf{s}}$ by rotating $\hat{\mathbf{n}}_\mathbf{v}$ in the $\hat{\mathbf{v}}_\mathbf{1} - \hat{\mathbf{v}}_\mathbf{2}$ plane by an angle $\frac{\pi}{2} - \phi$.

We can now verify that the distribution of $\hat{\mathbf{s}}$ satisfies Lambertian emissivity. By construction, the dot product of $\hat{\mathbf{s}}$ with the surface normal vector will be

$$\hat{\mathbf{s}} \cdot \hat{\mathbf{n}} = \sin(\phi)\sqrt{c_1^2 + c_3^2}. \tag{A5}$$

Although this looks complex, we can interpret it intuitively. We can generate $\hat{\mathbf{v}}_\mathbf{1}$ and $\hat{\mathbf{v}}_\mathbf{2}$ via a random rotation of the Euclidean basis vectors in $D-$dimensional space. Hence, c_1 and c_3 are the first two components of a random unit vector; equivalently, $\sqrt{c_1^2 + c_3^2}$ is the projected length of a random unit vector in D dimensions after projection onto the $\hat{\mathbf{x}} - \hat{\mathbf{y}}$ plane. We can interpret $\sin(\phi)$ as the distribution of projected lengths of $(D+1)$-dimensional unit vectors in $D-$dimensional space. Hence, combining the operations, the distribution of $\hat{\mathbf{s}} \cdot \hat{\mathbf{n}}$ is simply the distribution of projected lengths of $(D+1)$-dimensional unit vectors in a 2D plane.

We then note that a further projection from 2D to 1D would result in the familiar distribution for the dot product of two random unit vectors in $D+1$ dimensions:

$$L \equiv \sin(\psi)\hat{\mathbf{s}} \cdot \hat{\mathbf{n}} \tag{A6}$$

$$P(L) = \int_{\sin^{-1}(L)}^{\pi/2} P(\hat{\mathbf{s}} \cdot \hat{\mathbf{n}} = L/\sin\psi)\frac{d\psi}{\sin\psi}$$

$$\propto (1 - L^2)^{(D-2)/2}, \tag{A7}$$

where ψ is the angle from the $\hat{\mathbf{x}}$ axis, so that $P(\psi) \propto 1$, and the $\frac{1}{\sin\psi}$ term in Eq. A7 is equal to $\frac{d\hat{\mathbf{s}}\cdot\hat{\mathbf{n}}}{dL}$.

We can then show that $P(\hat{\mathbf{s}} \cdot \hat{\mathbf{n}}) \propto \hat{\mathbf{s}} \cdot \hat{\mathbf{n}}(1 - (\hat{\mathbf{s}} \cdot \hat{\mathbf{n}})^2)^{(D-3)/2}$ (i.e., Lambertian emission) is the solution that satisfies Eq. A7. We find that

$$\int_{\sin^{-1}(L)}^{\pi/2} \frac{L}{\sin\psi}\left(1 - \frac{L^2}{\sin^2\psi}\right)^{(D-3)/2}\frac{d\psi}{\sin\psi} \tag{A8}$$

$$= \int_L^1 \left(1 - u^2\right)^{(D-3)/2}\frac{du}{\sqrt{1 - \frac{L^2}{u^2}}}, \tag{A9}$$

using $u = L/\sin\psi$. This evaluates to:

$$(1 - L^2)^{(D-3)/2}\sqrt{u^2 - L^2}$$

$$\times \quad {}_2F_1\left(\frac{1}{2}, \frac{3-D}{2}; \frac{3}{2}; \frac{L^2 - u^2}{L^2 - 1}\right)\Bigg|_{u=L}^{u=1} \tag{A10}$$

$$= (1 - L^2)^{(D-2)/2}\,{}_2F_1\left(\frac{1}{2}, \frac{3-D}{2}; \frac{3}{2}; 1\right) \tag{A11}$$

$$\propto (1 - L^2)^{(D-2)/2}, \tag{A12}$$

as desired, noting that ${}_2F_1(a, b; c; 0) = 1$ for arbitrary (a, b, c) in Eq. A10, and that ${}_2F_1\left(\frac{1}{2}, \frac{3-D}{2}; \frac{3}{2}; 1\right)$ in Eq. A11 is a constant of integration that only depends on D.

B. RAY TRACING ALGORITHM USING VELOCITIES INSTEAD OF WEIGHTS

We can transform the ray-tracing formula (which changes the direction of propagation but not the magnitude) back into a velocity-dependent sampler, with the speed $|\mathbf{v}|$ equal to a function $S(\mathbf{x})$ (corresponding to an effective weight of $W = S^{-1}$). Notably, we cannot generally write down a potential $U(\mathbf{x})$ to describe the dynamics of this transformation. For example, with standard ray tracing, the ray speed does not change, regardless of the path.

The change in v given by:

$$\frac{d|\mathbf{v}|}{dt} = |\mathbf{v}|\frac{d|\mathbf{v}|}{ds} = S(\mathbf{x})^2 \cos\phi\,|\nabla_\mathbf{x} \ln(S(\mathbf{x}))|\,, \tag{B1}$$

where ϕ is the angle between the speed gradient and $\mathbf{v}$ (for $D > 1$; otherwise, $\phi = 0$). For $D > 1$, ϕ changes throughout a given timestep, with:

$$\phi(t) = \phi_i + \theta(t) - \theta_i = 2\arctan\left(\tan\left(\frac{\theta_i}{2}\right)\exp\left(-S(\mathbf{x})\Delta t|\nabla_\mathbf{x}\ln(n(\mathbf{x}))|\right)\right) + \phi_i - \theta_i, \tag{B2}$$

so we find using Mathematica that

$$|\mathbf{v}_f| = |\mathbf{v}_i| + S(\mathbf{x}) \cdot \frac{F(\mathbf{x}, S(\mathbf{x}))}{F(\mathbf{x}, n(\mathbf{x}))}\left[\cos(\phi_i - \theta_i)\left(\ln\left(\frac{1 + \cos(\theta_i)}{1 + \cos(\theta_f)}\right) + \Delta t \cdot F(\mathbf{x}, n(\mathbf{x}))\right) + \sin(\phi_i - \theta_i)(\theta_f - \theta_i)\right] \tag{B3}$$

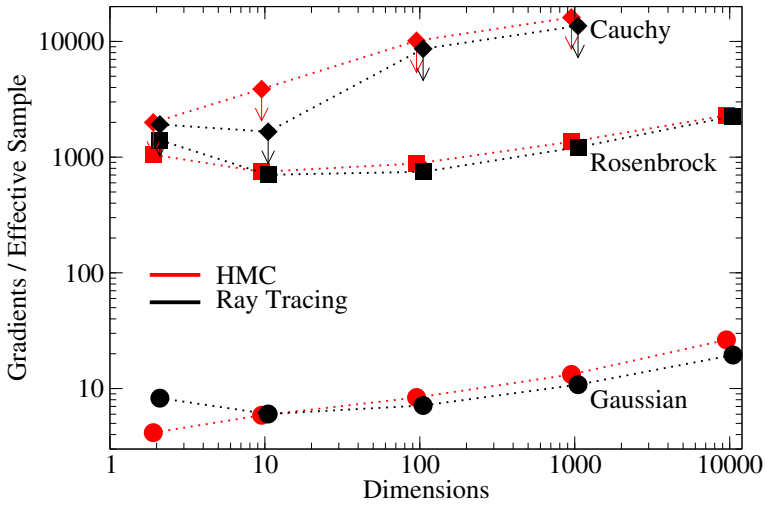

FIG. 20.— Comparison between sampling efficiencies (defined as gradients per effective sample) when perfect gradients are used, for ray tracing and HMC. No strong performance advantages are observed. The minimum dimension shown is 2, since ray tracing is not defined for $D = 1$. Upper limits are shown for the Cauchy distribution, since the sampling efficiency as a function of step size and trajectory length had many local minima, and it is not guaranteed that hand-tuning achieved the global minimum.

with $\theta_i = 0$ if $\nabla_{\mathbf{x}} n(\mathbf{x}) = 0$, and where the operator F performs:

$$F(\mathbf{x}, G(\mathbf{x})) = S(\mathbf{x})|\nabla_{\mathbf{x}} \ln(G(\mathbf{x}))|. \tag{B4}$$

As noted previously in the paper, a velocity-dependent sampler may be more susceptible to path heating when stochastic gradients are used.

C. SYMPLECTIC INTEGRATION

We note that published fully symplectic integration methods for ray-tracing exist (Satoh 2003; Li et al. 2016; Ohno 2020; McKeon & Goncharov 2023). Most of these adopt the convention $\mathbf{p} \equiv n\frac{d\mathbf{x}}{ds}$, with the Hamiltonian $\mathcal{H} = \frac{1}{2}(p^2 - n^2)$. In these formulations, $\frac{d\mathcal{H}}{dp_i} = n\frac{dx_i}{ds}$, so authors often define a "time" as $dt \equiv \frac{1}{n}ds$, which means that the effective sample weighting density would scale as n. An alternate symplectic integration scheme suggested by the master's thesis of Jorg Portegies[10] is to maintain the definition of $\mathbf{p} \equiv n\frac{d\mathbf{x}}{ds}$, but to adopt $\mathcal{H} = |\mathbf{p}| - n$. As a result, $\frac{d\mathcal{H}}{dp_i} = \frac{dx_i}{ds}$, so the sample weighting density is uniform per unit path length.

Fully symplectic integration has the advantage that it carries better guarantees about convergence scaling with timestep size. However, both methods above (as well as symplectic methods for nonseparable Hamiltonians) carry the disadvantage of needing to account for "off-shell" values of the Hamiltonian to maintain reversibility. Specifically, the paths of light rays satisfy $\mathcal{H} = 0$ for both definitions above, but real integrators will deviate, arriving at a new location with $\mathcal{H} = \Delta\mathcal{H} \neq 0$. As a result, the likelihood function $\mathcal{L}(\mathbf{x})$ needs to be lifted to include $\Delta\mathcal{H}$ (e.g., as $\mathcal{L}(\mathbf{x}, \Delta\mathcal{H}) = \mathcal{L}(\mathbf{x})\mathcal{L}(\Delta\mathcal{H})$) to maintain reversibility. Unfortunately, the dynamics of systems with $\mathcal{H} \neq 0$ are different (e.g., $\mathcal{H} < 0$ implies a bound system that cannot escape to infinity), and so the Jacobian becomes messy for such cases. While future investigation of symplectic integrators is certainly interesting, the approach we adopt in this work carries the advantages of 1) always remaining on-shell (so that there are no insurmountable potential barriers), 2) having a simple Jacobian, and 3) efficiently suppressing noise from stochastic gradients.

D. COMPARISON TO HAMILTONIAN MONTE CARLO FOR STANDARD DISTRIBUTIONS

In this section, we compare the hyperparameter tuning necessary for both ray tracing and HMC to achieve optimal sampling efficiency. We use the same autocorrelation time definition as in Section 3, and define the sampling efficiency as $E = 1/(G\tau)$, where G is the number of gradients per trajectory and τ is the autocorrelation time in units of trajectories per independent sample. We hand-tune both step sizes and trajectory lengths for both HMC and ray tracing to achieve optimum effective sample sizes per gradient computed. In particular, we felt that it would be useful for readers to know how to change existing HMC parameters to be able to use ray tracing; the short answer is that ray tracing seems to prefer similar (or slightly larger) step sizes, as well as somewhat longer trajectory lengths than HMC (see Table 1). To make the comparison easier with existing reader codebases, we do not use continuous momentum refreshment here.

In the tests here, we do not find a strong advantage (i.e., more than a factor of a few) for either HMC or ray tracing in any test (summarized in Fig. 20). This stands in contrast to some past results—e.g., Robnik et al. (2022) argued for stronger differences between HMC and microcanonical HMC (which should have performance very similar to ray tracing for $D > 10$). We can interpret this most simply from the theoretical basis in Section 2.9.1. For high-dimensional spaces, HMC and ray tracing (as well

[10] https://pure.tue.nl/ws/portalfiles/portal/46968663/778374-1.pdf

Gaussian Distribution		2D	10D	100D	1000D	10000D
Ray Tracing	Step Size	0.71	0.71	0.52	0.26	0.17
	Path Length	$\pi/2$	$\pi/2$	$\pi/2$	$\pi/2$	$\pi/2$
HMC	Step Size	0.67	0.5	0.36	0.21	0.15
	Path Length	$\pi/2$	$\pi/2$	$\pi/2$	$\pi/2$	$\pi/2$
Rosenbrock Distribution		2D	10D	100D	1000D	10000D
Ray Tracing	Step Size	0.01	0.015	0.013	0.0085	0.0045
	Path Length	3.9	3.7	2.6	2.5	2.5
HMC	Step Size	0.012	0.017	0.01	0.007	0.0045
	Path Length	3.4	3.2	2.5	2.5	2.4
Cauchy Distribution		2D	10D	100D	1000D	
Ray Tracing	Step Size	0.6	0.37	0.3	0.24	
	Path Length	55	65	70	70	
HMC	Step Size	0.5	0.4	0.32	0.23	
	Path Length	55	55	60	70	

TABLE 1
HAND-TUNED PARAMETERS FOR RAY TRACING AND HMC TO PRODUCE THE SAMPLING EFFICIENCIES SHOWN IN FIG. 20.

as microcanonical HMC) all have very similar sample weighting. As well, due to the narrow range of likelihood values in the typical set, there simply is not much freedom to explore regions outside of the typical set, and so none of the samplers enjoys a strong advantage. For lower-dimensional spaces, it would seem that a ray tracing-like method may have an advantage (e.g., in Fig. 1). But in practice, using a large step size with HMC results in a similar effect as within-trajectory resampling for the momentum vector, giving it similar performance as ray tracing. We were unable to contrive an example (even for multimodal distributions) that led to more than a factor of ~ 2 difference between the samplers' optimal tuning when a perfect gradient was used.

As a result, it may be possible that the past results in Robnik et al. (2022) were due more to the automatic tuning methods than to underlying capability. For example, Robnik et al. (2022) found that NUTS-adjusted HMC could not converge at all on a 1000-dimensional Cauchy distribution, but we were able to find hand-tuned HMC parameters that passed our convergence test without issues.

All tests in this section were performed on a Mac M1 Max laptop using the public `JAX` implementation of ray tracing and HMC, which employs DKD leapfrog integration.

D.1. *Independent Gaussian Distributions*

Many samplers do well on Gaussian distributions, so this test functions as an easily interpretable base efficiency for HMC and ray tracing. For the Gaussian tests, we initialize a single walker on the typical set, perform burn-in for 4,000 trajectories, and perform 4,000 more trajectories for the autocorrelation time estimation step. Both HMC and ray tracing have short autocorrelation times on these tests, so this provides 1,000 or more independent samples by the time the chain is complete.

D.2. *Rosenbrock Distributions*

The multidimensional Rosenbrock distribution we use here is defined by:

$$\log \mathcal{L}(\mathbf{x}) = \sum_{i=1}^{D/2} \left[100(x_{2i-1}^2 - x_{2i})^2 + (x_{2i-1} - 1)^2\right]. \tag{D1}$$

This can be recognized as a sheared multivariate Gaussian distribution, from which exact samples can be drawn using:

$$\begin{aligned} x_{2i-1} &= \mathcal{N}(1, 0.5) \\ v &= \mathcal{N}(0, 0.005) \\ x_{2i} &= v + x_{2i-1}^2, \end{aligned} \tag{D2}$$

where $\mathcal{N}(\mu, \sigma^2)$ refers to a normal distribution with mean μ and variance σ^2. The untransformed distribution (Eq. D1) can be challenging to sample because it has a narrow parabolic channel along which probability is distributed, and so many steps are typically needed for gradient-based samplers to traverse its entire length.

We find that false "convergence" can occur with both HMC and ray tracing, particularly for large step sizes that do not allow exploration of narrow probability channels. To measure true convergence, we perform 1000 trials of 1000 independent point samples each (using Eq. D2), and calculate the 68^{th} percentile ranges for the sample mean and sample standard deviation along each dimension. For each dimensionality D that we show here, we require at least 90% of the tests for the mean and standard deviation of each dimension to be within the 1,000-independent-sample ranges. For both ray tracing and HMC, we sample 10,000 trajectories before estimating the autocorrelation times.

D.3. *Cauchy Distributions*

The multidimensional Cauchy distribution we use here is defined by:

$$\ln \mathcal{L}(\mathbf{x}) = -D \ln \pi - \sum_{i=1}^{D} \log(1 + x_i^2) \tag{D3}$$

This can be challenging for samplers due to its wide probability tails, which can require very long trajectories to traverse. The Cauchy distribution has no well-defined average (indeed, its sample mean is itself Cauchy-distributed), and has infinite variance. Hence, the usual approach to test for convergence is to choose a different moment of the distribution that is well defined. Here, we choose the distribution of log-likelihoods for individual dimensions, which are distributed as:

$$P(\ln \mathcal{L}) = \left(\pi \exp(-\ln \mathcal{L}) - \pi^2\right)^{-1/2}, \tag{D4}$$

defined for $\ln \mathcal{L} < -\ln \pi$. This distribution has a well-defined average (~ -2.53102) and standard deviation (~ 1.8138). As for the previous subsection, we compute the 68$^{\rm th}$ percentile ranges for the sample means and sample standard deviations of 1,000 trials of 1,000 independently-drawn points from the Cauchy distribution (which are obtained by taking the ratio of two independent variables both distributed as $\mathcal{N}(0,1)$). As above, we consider a chain converged if at least 90% of the tests for the mean and standard deviation of each dimension fall within the 1,000-independent-sample ranges. For both samplers, we sample 200,000 trajectories before estimating the autocorrelation times.

E. NEURAL NETWORK DETAILS

E.1. *UniverseMachine Stellar Mass Estimates with a 1433-parameter Multi-Layer Perceptron*

To enable comparison between ray tracing and HMC, we use a relatively simple network architecture (i.e., fully-connected network layers). We found that converging geometries (i.e., fewer neurons in each successive layer) and flat geometries (similar neuron count in each successive layer) tended to produce higher fractions of stuck dimensions—i.e., predictions that had long autocorrelation times. We also found that smoother activation functions tended to produce fewer stuck dimensions. We hence use a fan geometry (each successive layer having more neurons) with scaled exponential linear unit (SELU; Klambauer et al. 2017) activation functions, with a final fully-connected linear layer producing the single output prediction. Specifically, after the input layer, the successive hidden layers have 8, 16, 24, and 32 neurons, followed by the final linear layer, leading to a total of 1433 trainable parameters.

For the input data, we use ~ 1 million halos subsampled from the UniverseMachine DR1 catalog (Behroozi et al. 2019, based on the Bolshoi-Planck simulation; Klypin et al. 2016; Rodríguez-Puebla et al. 2016), with an 80%/20% training/validation split. As inputs, we use single-snapshot dark matter halo properties (i.e., the scale factor, the halo mass, the maximum circular velocity, and two random seeds), and we train networks to match the corresponding galaxy stellar mass according to the UniverseMachine. Because the UniverseMachine relies on the entire halo growth history to generate a stellar mass, it is impossible for the networks to exactly fit the results given the input variables, and so typical best root mean squared errors achievable with the Adam optimizer (Kingma & Ba 2014) are ~ 0.35 dex. Inputs and outputs were all normalized to have zero mean and unit variance. For all trials, we scale the initial Kaiming (He et al. 2015) starting positions by a factor of 3 (for a typical norm of $|\mathbf{x}_0| \sim 12$) to explore a greater diversity of parameter space. Initial momenta (and refreshes) are drawn from random unit Gaussians (for a typical momentum norm of $|\mathbf{v}_0| \sim \sqrt{1560} \sim 39.5$).

For both HMC and ray tracing, we scale the loss (mean-squared error) by a factor of 5,000, and use mini-batch sizes of 20,000, so each training epoch has about 400 steps, and the typical $\sigma_{\ln \mathcal{L}}$ is about 10. We tested a range of timesteps to see where the trajectories tended to fail, settling on $\Delta t = 0.0002$ for both ray tracing and HMC. For both HMC and ray tracing, we adopt a stochastic momentum refresh fraction of $f = 5\Delta t$, and run 10 walkers for 1000 epochs each. For HMC, we also do a pre-burn-in with a momentum refresh fraction of $f = 50\Delta t$ for 5 epochs (allowing the excess kinetic energy gained from starting at an initially very unlikely position to decay). We fail trajectories that have ΔE (HMC) or $\Delta \ln L/\mathcal{L}$ (ray tracing) of more than $5\sigma_{\ln \mathcal{L}} \sim 50$, following Section 2.11.3. For comparison with existing practice, we also run Adam fitting (Kingma & Ba 2014) for 50 epochs with a learning rate of 0.02 (notably, the standard learning rate of 0.001 did not result in convergence), for 1000 different starting positions. Sampling and fitting were performed on an NVIDIA RTX 5090 using `PyTorch`. Since the network was not complex, throughput on this GPU was only $\sim 2-3\times$ faster than running on an Apple M1 Max laptop.

E.2. *Metallicity Estimates with ResNet-34*

In our simplified version of the Wu & Boada (2019) study, we use 1/10 of the available SDSS galaxy images for our main analysis, for a total of $N_{\rm train}$ =11,664 images in the training set and 1,296 images in the validation set, as this allowed us to have multiple independent training sets drawn from the same underlying distribution. As in the Wu & Boada (2019) study, we replace the last layer of ResNet-34 with a fully connected linear layer with a single output to perform regression to the renormalized spectroscopic metallicity (i.e., the metallicity shifted and rescaled to have zero mean and unit variance). We find that scaling the MSE loss by a factor of 4×10^4 (corresponding to $D_{\rm eff} \sim 1.4 \times 10^4 \sim N_{\rm train}$, or 0.06% of the full parameter space) allows indefinite sampling without overfitting. We use a timestep size of $\Delta t = 2 \times 10^{-6}$, corresponding to a spatial step of $\Delta s \sim 0.01$, or an angular step of initially $\Delta\phi \sim 10^{-4}$. For the main run, we use Kaiming initialization for a single MCMC walker, running for 5×10^5 epochs with a mini-batch size of 32. This mini-batch size results in a total GPU memory usage of $\sim$ 2GB, i.e., small enough to run on any iPhone released in the past 7 years. To mimic the approach in the Wu & Boada (2019) study, we also use Adam to train the last layer of the ResNet-34 for 2 epochs, and then allow training of all layers simultaneously for 8 epochs; this is repeated 1000 times to estimate the variance of fitting results. We also run a second walker for 5000 epochs with the same training data and a different initial seed, as well as a third walker with an independent (i.e., no overlap) training set of 11,664 images and a different initial seed. Sampling and fitting were performed initially on an NVIDIA RTX 3060 and later on an NVIDIA RTX 5090 using `PyTorch`. The RTX 3060 averaged 1400 epochs/day (5×10^5 steps/day), and the RTX 5090 averaged 5600 epochs/day (2×10^6 steps/day).

E.3. *Exploring GPT-2 Posteriors*

To explore GPT-2 posteriors, we use the `nanoGPT` implementation,[11] which we modified 1) to use the ray tracing algorithm for parameter updates, and 2) to lower GPU memory usage by ∼10% with better garbage collection. We used a mini-batch size of 5 (with 1024 tokens in each sample), this being the largest mini-batch size that could fit on our 32GB RTX 5090. For training data, we used a 100 billion-token subset of the `fineweb-edu` dataset (Penedo et al. 2024), taken from the public `llm.c` implementation, due to the original WebText training set used by OpenAI being private. We initialized the model parameters from public 1.5B-parameter GPT-2 model weights that had been trained using the same `fineweb-edu` dataset.[12]

To perform MCMC sampling, we used shuffled training set data, but to estimate the loss evolution over training epochs for posterior autocorrelation tests, we used a fixed random sample of 400×5120 tokens each from the training and validation sets. This ensured that most computations were spent in parameter updates as opposed to computing the loss function precisely. With the RTX 5090, we could complete ∼120,000 mini-batches per day, covering about 600M input tokens per day (163 days per epoch). We used a timestep size of $\Delta t = 5 \times 10^{-6}$, corresponding to a parameter space step of $\Delta s = 0.2$, or an angular step of $\Delta\phi = 2 \times 10^{-4}$.

This paper was built using the Open Journal of Astrophysics LaTeX template. The OJA is a journal which provides fast and easy peer review for new papers in the `astro-ph` section of the arXiv, making the reviewing process simpler for authors and referees alike. Learn more at http://astro.theoj.org.

[11] https://github.com/karpathy/nanoGPT

[12] https://huggingface.co/karpathy/gpt2_1558M_final3_hf, chosen because the author's later version with a longer training time quickly explodes with further training, per https://huggingface.co/karpathy/gpt2_1558M_final4_hf.