Automatic Differentiation of Programs with Discrete Randomness

Anonymous Author(s)
Affiliation
Address
email

Abstract

Automatic differentiation (AD), a technique for constructing new programs which compute the derivative of an original program, has become ubiquitous throughout scientific computing and deep learning due to the improved performance afforded by gradient-based optimization. However, AD systems have been restricted to the subset of programs that have a continuous dependence on parameters. Programs that have discrete stochastic behaviors governed by distribution parameters, such as flipping a coin with probability p of being heads, pose a challenge to these systems because the connection between the result (heads vs tails) and the parameters (p) is fundamentally discrete. In this manuscript we develop a new reparameterization methodology that allows for generating programs whose expectation is the derivative of the expectation of the original program. We showcase how this method gives an unbiased and low variance estimator which is as automated as traditional AD mechanisms. We demonstrate unbiased forward-mode AD of discrete-time Markov chains, agent-based models such as Conway's Game of Life, and unbiased reverse-mode AD of a particle filter.

1 Introduction

Automatic differentiation (AD) is a technique for taking a mathematical program $X(p)$ and generating a new program $\widetilde{X}(p) = \frac{\mathrm{d}X}{\mathrm{d}p}$ for computing the derivative [1, 2]. AD is widely used throughout machine learning and scientific computing due to the increased performance of gradient-based optimization techniques compared to derivative-free methods [3]. However, if $X(p)$ returns the flip of a coin with probability p of receiving a 1 and probability $1 - p$ of receiving a 0, it is clear that $\frac{\mathrm{d}X}{\mathrm{d}p}$ is not defined in the classical sense. But when attempting to calibrate the parameter p to data, one may wish to fit the model using the statistical quantities, e.g. find p such that the average of $X(p)$ is close to the average sum of N real-world coin flips. Given this use case, can one automatically construct a program $\widetilde{X}(p)$ that computes the derivative of the statistical quantities, i.e. $\mathbb{E}[\widetilde{X}(p)] = \frac{\mathrm{d}\mathbb{E}[X(p)]}{\mathrm{d}p}$?

A naïve solution to this problem would be to use finite differences, i.e.:

$$\frac{\mathrm{d}\mathbb{E}[X(p)]}{\mathrm{d}p} \approx \frac{\mathbb{E}[X(p+\varepsilon)] - \mathbb{E}[X(p)]}{\varepsilon}. \tag{1.1}$$

Finite differences' major issue in the context of stochastic programs is that this calculation does not correlate the calculation of $\mathbb{E}[X(p+\varepsilon)]$ with the calculation $\mathbb{E}[X(p)]$. This leads to a large variance in the finite-difference estimator [4] that goes to ∞ as $\varepsilon \to 0$. This variance issue could be solved if one could run the perturbed program with the same set of random numbers, i.e. sample from $X(p+\varepsilon) - X(p)$ as depicted in Figure 1.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

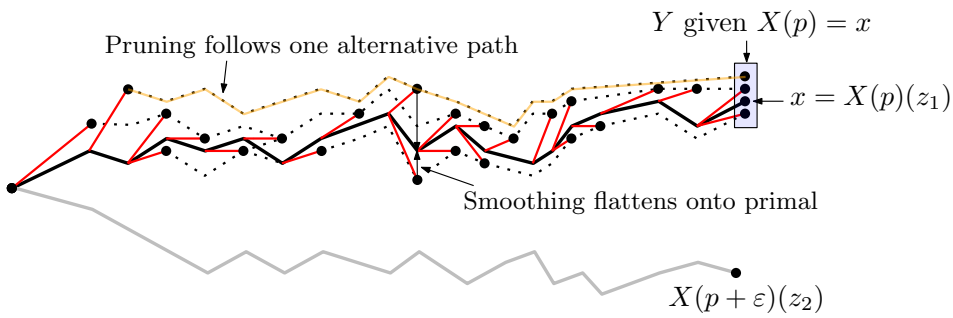


Figure 1: Qualitative sketch of our pruning and smoothing methods and comparison to finite differences, adopting the notation of Section 2. The primal computation (solid black line) samples $X(p)$ with random number sequence z_1. Black-box finite differences samples an independent copy of the program $X(p+\varepsilon)$ with random number sequence z_2 (bottom) for some *finite* choice of ε. In contrast, the component Y of the stochastic derivative of the program considers the effect of the minimal possible perturbations (solid red lines diverging from primal path) to the program which could stem from a truly *infinitesimal* change ε in the input p.

We demonstrate how to automatically construct a stochastic program $\widetilde{X}(p)$ whose expectation satisfies $\mathbb{E}[\widetilde{X}(p)] = \frac{d\mathbb{E}[X(p)]}{dp}$. We derive this through a technique which we term the *stochastic derivative*, propagating the proportional probability of differing event outcomes due to infinitesimal changes in p. Our technique has several benefits:

- **Composability.** We derive an extension language-level AD to stochastic programs composed of any of a wide range of "elementary" stochastic programs, including discrete distributions such as Bernoulli, Geometric, and Poisson distributions, which may be freely mixed with continuous distributions such as Gaussian and exponential distributions.
- **Unbiasedness and low-variance.** The discrete structure of the original program is preserved while staying coupled, avoiding continuous relaxations which exhibit a variance-bias trade-off and cannot be applied to highly discrete constructs.
- **No tunable parameters.** A single run of the primal computation is transformed into an unbiased estimate of the derivative in a stateless fashion with no internal parameters to tune.
- **Speed.** Our technique's computational cost is proportional to the number of steps in the original program and is independent of how stochastic programs are composed together.

We show the utility of our technique by demonstrating AD of stochastic simulations like inhomogenous random walks, AD of agent-based models such as the Game of Life, and the construction of an end-to-end differentiable particle sampler. Additionally, we provide an open source implementation of the method, `StochasticAD.jl`, for readers to explore the technique on their own applications.

1.1 Related work

Gradient estimators for stochastic functions can be divided into three different classes [5]: score-function [6, 7, 8], measure-valued [9, 10], and pathwise [11, 12] gradient estimators. Our method operates as a gradient estimator, but beyond that, the main feature is composability with arbitrary user-written functions, enabling an automated mechanism. When it comes to discrete randomness, the score-function method is a popular general-purpose choice because it can be composed by stochastic computation graphs [13] and provides an unbiased estimate of the gradient. However, the score-function method cannot be used to, e.g., differentiate a resampling step, and has other well-known limitations [5]. Importantly, the score-function method is not coupled to the primal evaluation and suffers from high variance, making gradient computation with discrete variables challenging. A number of techniques (REBAR [14], RELAX [15]) have been introduced that rely on control variates for variance reduction [16]. Measure-valued derivations have no notion of intrinsic coupling, though coupling can be achieved by using common random numbers for certain distributions [17]. A drawback of this coupling method is that its implementation may require nontrivial changes to the primal evaluation code, which makes it less attractive for automation. As an alternative direction, several recent works such as Gumbel-Softmax [18] have considered continuous relaxations of the

program so that the reparameterization trick can then be applied. However, such methods suffer a variance-bias tradeoff and furthermore are inapplicable to highly discrete programs which cannot be continuously relaxed. Our method extends the pathwise gradient estimator from continuous to discrete programs while preserving all discrete structure in the original computatoin. Smoothed perturbation analysis [19, 20] develops a mathematically equivalent object to the smoothed stochastic derivative based on conditional expectations, which is a special case of our formalism. But such ideas have not previously been applied to construct a general-purpose AD algorithm via rigorous composition rules and computational structures.

While existing reverse-mode AD frameworks do not support unbiased differentiation of discrete random programs, Storchastic [21] is a specialized framework for AD of stochastic computation graphs, where the user can specify which estimator to use at each node, as well as any tunable hyperparameters. Storchastic implements an exhaustive set of prior gradient estimation methods at each sampling step. However, previous reviews of these techniques revealed a performance scaling issue: the run time of the derivative estimate is exponential in the length of the largest chain of stochastic nodes, a symptom of the way many prior gradient estimators compose [22]. In Section 3 we demonstrate forward-mode and reverse-mode stochastic AD with identical computational complexity to their deterministic counterparts, alleviating these performance issues.

2 Composable derivatives of stochastic programs

In this section, we develop the notion of a *stochastic derivative* for programs containing discrete randomness. We shall motivate this object as a natural generalization of the standard reparameterization trick, and present the key ideas underpinning the formalism. In offering intuition, we make frequent use of big-O notation for infinitesimal asymptotics, where a quantity is $\mathcal{O}(\varepsilon)$ if bounded above in magnitude by $C\varepsilon$ for some real C and all sufficiently small ε. Colloquially, we describe quantities that are $\mathcal{O}(\varepsilon)$ as "infinitesimal".

2.1 Infinitesimally perturbing a stochastic program

Consider a stochastic program: a program which when provided with input p gives a random variable $X(p)$ as output. To obtain a single sample of $X(p)$, which we call the *primal evaluation*, one should imagine a number z being randomly chosen from a sample space Z to produce a then deterministic output $X(p)(z)$. For example, a Bernoulli random variable with probability p can be represented by picking a uniform random number $z \in [0, 1]$ and defining $X(p)(z) = \mathbf{1}_{[1-p,1]}(z)$ where $\mathbf{1}_\Omega(z)$ is the indicator function for $z \in \Omega$. Note that parameterizations of many univariate distributions can be constructed via the inversion method [23].

We define the *stochastic differential* $\mathrm{d}X(\varepsilon)$ as the change in the stochastic program $X(p)$ w.r.t. a change ε in its input p,

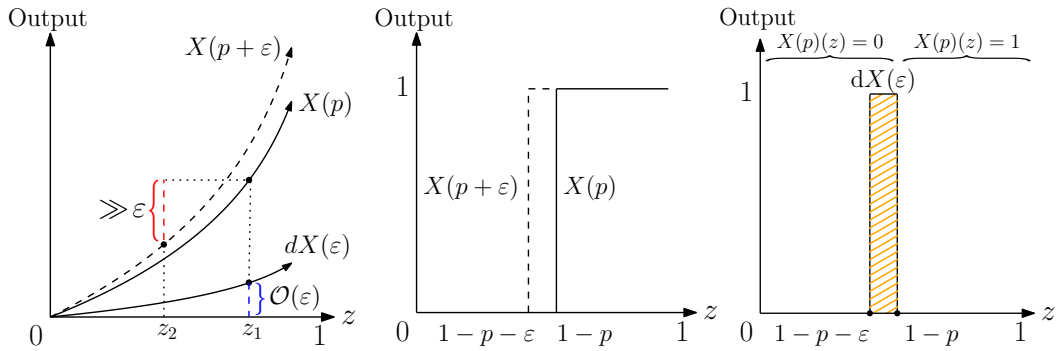
$$\mathrm{d}X(\varepsilon) = X(p + \varepsilon) - X(p), \tag{2.1}$$

Note that in the stochastic setting, $\mathrm{d}X(\varepsilon)$ is itself a random variable over the same sample space as $X(p)$. Crucially, $X(p + \varepsilon)$ and $X(p)$ share the same sample space (i.e. belong to a joint distribution), distinguishing Eq. (2.1) from black-box finite differences.

For the full user-provided stochastic program, we are interested in computing $\partial_p \mathbb{E}[X(p)]$, which is related to $\mathbb{E}[\mathrm{d}X(\varepsilon)]$ by

$$\partial_p \mathbb{E}[X(p)] = \lim_{\varepsilon \to 0} \frac{\mathbb{E}[\mathrm{d}X(\varepsilon)]}{\varepsilon}. \tag{2.2}$$

We thus expect $\mathrm{d}X(\varepsilon)$ to have "infinitesimal" $\mathcal{O}(\varepsilon)$ expectation as ε approaches 0. Let us consider the form of $\mathrm{d}X(\varepsilon)$ for two illustrative examples: a program returning a sample from the continuous exponential distribution $\mathrm{Exp}(p)$ with scale p and a program returning a sample from the discrete Bernoulli distribution $\mathrm{Ber}(p)$ with probability p, both parameterized via the inversion method over the sample space $[0, 1]$ (Fig. 2). For the exponential distribution, $X(p)(z) = -p \log(1 - z)$ is differentiable in p. Thus, the range of $\mathrm{d}X(\varepsilon)$ is $\mathcal{O}(\varepsilon)$ for any fixed z, as illustrated in Fig. 2a. In contrast, consider the Bernoulli distribution. As shown in Fig. 2b, the output at a random $z \in Z$ almost surely does not change when the input probability p is perturbed by ε. Specifically, $\mathrm{d}X(\varepsilon)$ is 0 with probability $1 - \varepsilon \approx 1$, and assumes the value 1 with probability ε (Fig. 2c). This is the fundamental challenge presented by discrete randomness: a $\mathcal{O}(\varepsilon)$ change in the input turns into



(a) $X(p) \sim \mathrm{Exp}(p)$. (b) $X(p) \sim \mathrm{Ber}(p)$. (c) $X(p) \sim \mathrm{Ber}(p)$.

Figure 2: Illustrations of the stochastic differential $\mathrm{d}X(\varepsilon)$, where a small finite ε is chosen to illustrate the behavior as ε approaches 0, and we take $\varepsilon > 0$ for simplicity. **Left:** The $\mathcal{O}(\varepsilon)$ range of $\mathrm{d}X(\varepsilon)$ for continuous $X(p)$. The blue dashed line indicates a sample from $\mathrm{d}X(\varepsilon)$, while the red dashed line indicates a sample obtained by black-box finite differences. **Middle:** The original program $X(p)$ and the perturbed program $X(p+\varepsilon)$ for discrete (Bernoulli) $X(p)$. **Right:** Only those samples z below the hatched area cause a non-zero change in output. The hatched area corresponds to $\mathbb{E}[\mathrm{d}X(\varepsilon)]$, the expected size of a change for randomly drawn z.

a *finite perturbation* in the output that is non-vanishing as $\varepsilon \to 0$. The perturbation occurs with "infinitesimal" $\mathcal{O}(\varepsilon)$ probability, contributing to the $\mathcal{O}(\varepsilon)$ expectation of $\mathrm{d}X(\varepsilon)$.

2.2 Coupling to the primal

To produce low-variance estimates of the derivative of $X(p)$, it is important that the primal and derivative computations are coupled by a shared source of randomness. To see this, let us understand what happens when they are entirely uncoupled. A black-box finite difference approach with step size ε would independently sample z_1 and z_2 from the sample space (Fig. 2a), computing the derivative estimate $1/\varepsilon \cdot (X(p+\varepsilon)(z_2) - X(p)(z_1))$. However, since the samples in the numerator are independent, the variance of these samples is of order $1/\varepsilon^2 \to \infty$, so we are forced to pick a finite step size and balance a variance-bias tradeoff.

In the case of continuous randomness, the reparameterization trick surmounts this issue by differentiating $X(p)$ point-wise at each fixed random sample z_1 (Fig. 2a) to form $X'(p)$. Mathematically, the reparameterization trick relies on the following interchange of limits and expectation,

$$\partial_p \mathbb{E}[X(p)] = \lim_{\varepsilon \to 0} \frac{\mathbb{E}[\mathrm{d}X(\varepsilon)]}{\varepsilon} \stackrel{?}{=} \mathbb{E}\left[\lim_{\varepsilon \to 0} \frac{\mathrm{d}X(\varepsilon)}{\varepsilon}\right] = \mathbb{E}\left[X'(p)\right]. \tag{2.3}$$

But, considering a simple Bernoulli variable as in Fig. 2c, we see how this approach is ill-suited for the discrete case! As ε approaches 0, the stochastic differential $\mathrm{d}X(\varepsilon)$ is non-zero with infinitesimal $\mathcal{O}(\varepsilon)$ probability and thus $\frac{\mathrm{d}X}{\varepsilon}$ is almost surely 0 as ε approaches 0. This means $X'(p)$ is everywhere 0, while the true derivative of $\mathbb{E}[X(p)]$ is $\partial_p p = 1$. The finite perturbation is neglected.

Thus, the reparameterization trick needs to be modified to handle discrete randomness. Our insight is that $\mathrm{d}X(\varepsilon)$ may in general be decomposed into an infinitesimal component (i.e. of magnitude $\mathcal{O}(\varepsilon)$) and a component consisting of finite perturbations that occur with infinitesimal probability. We can sample from these parts of the probability space *separately*, curing the degeneracy of finite changes occurring with infinitesimal probability. As in stratified sampling, the expectation is then obtained as a weighted average of the separately sampled parts, cf. [4].

Definition 2.1. Suppose $X(p)$ is a stochastic program with real input, whose output belongs to a Euclidean space E. We say that the triple of random variables (δ, w, Y), with $\delta \in E$, weight $w \in \mathbb{R}$ and alternative outcome $Y \in E$ is a *stochastic derivative* of X at the input p if for all smooth bounded real functions f,

$$\frac{\partial}{\partial p} \mathbb{E}\left[f(X(p))\right] = \mathbb{E}\left[f'(X(p)) \cdot \delta + w \cdot (f(Y) - f(X(p)))\right]. \tag{2.4}$$

Theorem 2.2 (Simplified)**.** *Given a sufficiently regular stochastic program $X(p)$ with derivative $X'(p)$, there exists a right stochastic derivative $(X'(p), w_R, Y_R)$ of p with $w_R \geq 0$. Likewise there exists a triple $(X'(p), w_L, Y_L)$ with $w_L \leq 0$ giving a left stochastic derivative.*

The components w and Y of the stochastic derivative form a correction to the output $\delta = X'(p)$ of the reparameterization trick, necessary to handle potential finite perturbations to the output. Theorem 2.2 shows existence of the stochastic derivative for a wide range of stochastic programs, subject to technical assumptions given in our formal treatment (Appendix B). Its proof is by construction, where the weight w can be understood as the derivative w.r.t. ε of the probability that $\mathrm{d}X(\varepsilon)$ jumps by a non-infinitesimal amount, as characterized by the following event,

$$A(\varepsilon) = \{|\mathrm{d}X(\delta) - \delta \cdot X'(p)| > |\delta| \text{ for some } \delta \text{ between } 0 \text{ and } \varepsilon\}, \tag{2.5}$$

so that w is given as follows (separating the cases of right and left-handed limits):

$$w_{R/L} = \lim_{\varepsilon \to 0^{+/-}} \frac{\mathbb{P}\left(A(\varepsilon) \mid X(p)\right)}{\varepsilon}. \tag{2.6}$$

The component $Y_{R/L}$ then has distribution given as the limit as $\varepsilon \to 0^{+/-}$ of the conditional distribution of $X(p+\varepsilon) = X(p) + \mathrm{d}X(\varepsilon)$ given the event $A(\varepsilon)$ and the outcome of $X(p)$. In words, Y represents the distribution of possible alternate values of $X(p)$, conditioned on the primal output and on the fact that some finite perturbation occurs.

Example 2.3 (Stochastic derivative of Bernoulli variable)**.** Suppose $X(p) \sim \mathrm{Ber}(p)$, parameterized via the inversion method, and take $\varepsilon > 0$. As shown in Fig. 2c, $\mathrm{d}X$ is parameterized as,

$$\mathrm{d}X(\varepsilon)(z) = \begin{cases} 1 & \text{if } 1 - p - \varepsilon < z < 1 - p, \\ 0 & \text{otherwise.} \end{cases}$$

If $X(p) = 1$, then $z > 1 - p$ and $\mathrm{d}X(\varepsilon)$ is deterministically 0. On the other hand, if $X(p)$ is 0, then $z < 1 - p$, so with probability $\varepsilon/(1-p)$ the stochastic differential $\mathrm{d}X(\varepsilon)$ assumes a value of 1, i.e. the Bernoulli variable flips from 0 to 1. Thus, letting $w_R = 1/(1-p), Y_R = 1$ conditionally on $X(p) = 0$, and $w_R = 0$ conditionally on $X(p) = 1$, the triple $(0, w_R, Y_R)$ is a right stochastic derivative of the program $X(p) \sim \mathrm{Ber}(p)$.

We give a wide range of examples of left and right stochastic derivatives in Appendix A. It is of crucial importance that w and Y are random variables depending conditionally on the output of $X(p)$. Intuitively, this coupling allows us to consider only the smallest possible perturbations to the program in the derivative computation (Fig. 1), and thereby achieve variance reduction without resorting to continuous relaxations. For example, for a binomial variable $X(p) \sim \mathrm{Bin}(n, p)$ parameterized via the inversion method (a natural parameterizaton for maximizing coupling), it holds that $Y \in \{X(p) - 1, X(p) + 1\}$. We show in Example A.1 that the variance of our gradient estimator for such a binomial variable is $\mathcal{O}(n)$, whereas the score function estimator has variance $\mathcal{O}(n^3)$, justifying this intuition.

2.3 Composition of stochastic derivatives

Definition 2.1 defines the notion of stochastic derivative and Theorem 2.2 shows existence. The missing ingredient is *composition*, i.e. a stochastic derivative "chain rule". Because of the chain rule, we only need to hand-derive stochastic derivatives via Theorem 2.2 for each "elementary" stochastic program which makes up the full program: a stochastic derivative of the full program is obtained automatically by composition.

Proposition 2.4 (Chain rule)**.** *Consider independent stochastic programs X_1 and X_2 and their composition $X_2 \circ X_1$. Suppose that X_1 has a stochastic derivative at p given by (δ_1, w_1, Y_1) and X_2 has a stochastic derivative (δ_2, w_2, Y_2) in the direction δ_1 given conditionally on its input $X_1(p)$, as constructed by Theorem 2.2. Then the stacked program $[X_1; X_2 \circ X_1]$ has a stochastic derivative given by (δ, w, Y) where $\delta = [\delta_1; |\delta_1|\delta_2]$ and*

$$Y \quad = \quad \begin{cases} [Y_1; X_2(Y_1)] & \textit{with probability} \quad \dfrac{w_1}{w_1 + |\delta_1| \cdot w_2}, \\ [X_1; Y_2] & \textit{with probability} \quad \dfrac{|\delta_1| \cdot w_2}{w_1 + |\delta_1| \cdot w_2}, \end{cases} \tag{2.7}$$

and $w = w_1 + |\delta_1| \cdot w_2$.

Proposition 2.4 has two important special cases. The first is when X_2 is well-described by the reparameterization trick, i.e. $w_2 = 0$. In this case, Proposition 2.4 shows that the derivative δ_1 of $X_1(p)$ simply propagates through X_2 via the chain rule. In addition, the possible alternate values of $X_1(p)$, as described by Y_1, propagate through X_2 exactly as one would expect: the output has alternate values described by $X_2(Y_1)$, which are given the same weight w_1 (or "infinitesimal probability") as the original perturbations.

The key is the case where X_2 is discrete random. Here, an infinitesimal perturbation to the input of X_2 leads to potential finite perturbation(s) to the output, introducing another event with infinitesimal probability to be considered by the stochastic derivative. This case is what links infinitesimal changes to finite changes, necessitating the additional components w and Y of the stochastic derivative: without discrete randomness, w would always remain 0, recovering the traditional chain rule. Crucially, as $\varepsilon \to 0$ we may neglect the $\mathcal{O}(\varepsilon^2)$ probability of two finite perturbations occurring simultaneously: in Eq. (2.7), either X_1 is perturbed (and this perturbation propagates to X_2), or X_1 is not perturbed and a new perturbation occurs when evaluating X_2.

2.4 Smoothed stochastic derivatives

For discrete random variables, the reparameterization trick is deterministically 0, while stochastic derivatives fully capture potential finite perturbations with infinitesimal probability. There exists a middle ground between these methods: one may take the *conditional expectation* of the stochastic differential $\mathrm{d}X(\varepsilon)$, such that the derivative has the correct expectation given the primal output, but a finite perturbation has been replaced by an infinitesimal one.

Definition 2.5 (Smoothed stochastic derivative)**.** For a stochastic program $X(p)$ with right (left) stochastic derivative (δ, w, Y) at input p, the random variable $\widetilde{\delta}$ is a right (left) smoothed stochastic derivative of $X(p)$ at input p if

$$\mathbb{E}\left[\widetilde{\delta} \;\middle|\; X(p)\right] = \mathbb{E}[\delta + w(Y - X(p))|X(p)]. \tag{2.8}$$

The key advantage of smoothed stochastic derivatives is that they immediately permit reverse-mode AD instead of forward-mode, as they have the same form as the traditional derivative. However, they enjoy more limited composition properties: they propagate exactly through smoothed functions f that are linear over the conditional distribution of Y given $X(p)$ via the standard chain rule, as we prove in Appendix B.5. Due to the coupling of Y and $X(p)$, this is a much weaker requirement than global linearity, which can lead to low-bias estimates: for example, for the program $X(p) = \mathrm{Geo}(p)^3$ smoothed stochastic derivatives give a derivative estimate with $< 0.5\%$ bias at $p = 0.01$, even though the cube function is highly non-linear on the inter-quartile range $[28, 137]$ of $\mathrm{Geo}(p)$. Furthermore, as we showcase in Section 3.4, smoothed stochastic derivatives lead to *unbiased* AD of the resampling step of a particle filter.

3 Automatic differentiation of stochastic programs

We develop `StochasticAD.jl`, a prototype package for stochastic AD based on our theory of stochastic derivatives. As discussed in Section 2.4, *smoothed* stochastic derivatives obey the usual chain rule, and thus can be used with existing AD infrastructure by supplying custom rules for discrete random constructs. We demonstrate unbiased reverse-mode AD of a particle filter via smoothing in Section 3.4. However, performing automatic differentiation with unsmoothed stochastic derivatives, which are unbiased in all cases, requires new innovation. We develop a novel computational object called a *stochastic triple*, introduced in Section 3.1 and showcased in Section 3.2 and Section 3.3,.

3.1 Educational toy example of stochastic triples

Traditionally, forward-mode AD has been made possible by *dual numbers* [2], which pair the primal evaluation of a deterministic function $f(p)$ with its derivative $\partial_p f(p)$. Dual numbers can be propagated through a program using the chain rule. A useful alternative perspective of dual numbers is that they propagate an "infinitesimal" perturbation ε to the input through the program, where $\partial_p f(p)$ is the coefficient of the "dual" element ε. Stochastic triples generalize dual numbers by including a third component Δs to describe discrete perturbations with infinitesimal *probability* (Fig. 3, left).

In Fig. 3, we consider a toy program $X(p)$ including discrete randomness. We are interested in the sensitivity of $X(p)$ at $p = 0.6$, and hence we provide the stochastic triple printed as `"0.6 +`

```julia
struct StochasticTriple
    value # primal evaluation
    δ # "infinitesimal" component
    Δs # component of discrete change
       # with "infinitesimal"
       # probability
end
```

```julia
using Distributions
function X(p) # p = 0.6 + ϵ
    a = p^2 # a = 0.36 + 1.2ϵ
    b = rand(Binomial(10, p))
    # b = 6 + (1 with probability 10.0ϵ)
    c = 2 * b + 3 * rand(Bernoulli(p))
    # c = 12 + (3 with probability 12.5ϵ)
    return c * rand(Normal(p, p^2))
end
```

```julia
julia> using StochasticAD
julia> st = stochastic_triple(X, 0.8) # sample a single stochastic triple at p = 0.8
10.5 + 23.0ε + (2.63 with probability 12.5ε)
julia> partial_expectation(st) # which produces a single derivative estimate...
55.82
julia> samples = [derivative_estimate(X, 0.5) for i in 1:1000] # take many estimates!
julia> println("derivative of E[X(p)]: $(mean(samples)) ± $(std(samples) / sqrt(1000))")
derivative of E[X(p)]: 23.34 ± 0.56
```

Figure 3: **Left:** Stochastic triple structure (simplified). **Right:** A toy program $X(p)$ to differentiate; comments show intermediate outputs for a single random run. **Bottom:** Differentiating $\mathbb{E}[X(p)]$; printout float precision reduced for clarity. The used stochastic derivatives are listed in Appendix A.

ε" as input. First, the triple is squared and becomes "0.36 + 1.2ε": this is the familiar way that dual numbers propagate, via the chain rule. But what happens when the triple is propagated through the discrete and random Binomial variable? The resultant stochastic triple "6 + (1 with probability 10.0ε)" is an integer with a component of *discrete* change, reflecting an infinitesimal probability of one more success in the Binomial. Formally, we can interpret a printout "x + δε + (Δ with probability wε)" as follows: x is a sample of the random variable $X(p)$ describing the primal evaluation, and δ, w, and x+Δ are samples of the components δ, w, and Y, respectively, of the stochastic derivative of X at p.

Using our chain rule for stochastic derivatives (Proposition 2.4), we write rules for propagating stochastic triples through functions via operator overloading [24], exploiting Julia's multiple dispatch feature [25]. When multiple discrete changes are possible, we pick one probabilistically: we call this strategy *pruning* (see Fig. 1) and show its unbiasedness in Appendix B.4. For example, in line 6 of Fig. 3, right, we probabilistically choose between the perturbation to the Binomial and the perturbation to the Bernoulli, in this case picking the latter. Thus, stochastic triples may efficiently propagate through the full toy function written in Fig. 3. The function `derivative_estimate` creates a stochastic triple, propagates it, and collapses it into the derivative estimate δ + wΔ, forming an unbiased estimate of the derivative via Definition 2.1 (Fig. 3, bottom). Thus, stochastic triples enable the automatic construction of a program $\widetilde{X}(p)$ satisfying $\mathbb{E}[\widetilde{X}(p)] = \frac{\mathrm{d}\mathbb{E}[X(p)]}{\mathrm{d}p}$.

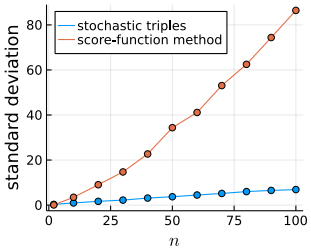
3.2 Inhomogeneous random walk

We consider a Markovian random walk $x_0, \ldots, x_n$ on $\mathbb{Z}_{\geq 0}$, with transition behavior dependent on a parameter p as follows,

$$x_n = \begin{cases} x_{n-1} + 1 \text{ with probability } \exp\left(-\frac{x_{n-1}}{p}\right) \\ x_{n-1} - 1 \text{ with probability } 1 - \exp\left(-\frac{x_{n-1}}{p}\right) \end{cases}, \quad x_0 = 0. \tag{3.1}$$

One can write a program that stochastically simulates this walk and applies an arbitrary non-linear function f to the output x_n. In practice, f may represent a loss or a likelihood estimate; in this toy setting, we take $f(x) = x^2$. We are interested in studying the asymptotic behavior of the variance of our automatically derived gradient estimator, and so we choose the initial value of p to be n so that the transition function varies appreciably over the range of the walk for all n. Fig. 4 compares the asymptotic growth in variance of the score function estimator to that of stochastic triples. Both estimators are unbiased with identical computational cost, but their variances differ by an order of magnitude. Empirically, we find a polynomial fit with leading term $\sim n^{1.5}$ for the variance of the score function, and a fit with leading order $\sim n$ for the variance of stochastic triples.

```julia
using StochasticAD, Distributions
X = 0
for step in 1:n
  i = rand(Categorical(probs[X]))
  X += steps[i]
end
return f(X)
```



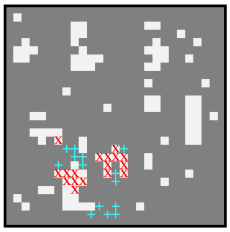


Figure 4: Automatic differentiation of discrete-time Markov processes. **Left:** Generic code for a 1D random walk, which may be automatically differentiated; `probs[X]` represents the transition probabilities at value `X`. **Middle:** The variance of unbiased gradient estimates of the random walk program using the score function and stochastic triple. **Right:** The final board for one run of the stochastic game of life with $N = 12$ and $T = 10$, where the "+" signs represents additional alive cells (white) in the stochastic alternative path, and the "X" signs represent additional dead cells (grey).

Crucially, stochastic triples achieve this variance reduction while operating entirely in discrete space, automatically producing a gradient estimate that is provably unbiased.

3.3 Stochastic game of life

For our next example, we differentiate a program based on a stochastic version of John Conway's game of life, played on a two-dimensional board. Such types of discrete stochastic examples arise in many applications. For example, mixing machine learning with agent-based models found in epidemiology and sociological contexts [26] or rule-based models and discrete stochastic (Gillespie) simulations of discrete chemicals in systems biology [27, 28] require similar program constructs. In the traditional game of life, a dead cell becomes alive when 3 of its neighbours are alive, while an alive cell survives when 2 or 3 of its neighbours are alive. In our stochastic version, each of these events instead has probability 95%, while their complementary events have probability 5%.

Consider a program that populates each cell of an $N \times N$ board with probability p, runs the stochastic game of life for T time steps, and counts the number of alive cells n_{alive}. Our goal is to perform a sensitivity analysis of the final living population with respect to the initial living population, i.e. differentiate the expectation of n_{alive} with respect to p. Stochastic triples propagate fully through this program, leading to an unbiased estimate of the derivative, as we verify with black-box finite differences. The board is depicted in Fig. 4 where the difference of the "alternative path" with infinitesimal probability considered by stochastic triples in a single run of the program (recall Fig. 1) is highlighted. This is a high-dimensional example (state space has dimension $N^2 = 144$) with fundamentally discrete structure, providing support for the algorithmic correctness and generality of stochastic triples. In particular, the program cannot be continuously relaxed by AD since it includes array indexing; in the approach of stochastic triples, integer-valued quantities stay integers, and therefore may be used in highly discrete constructs.

3.4 Differentiable particle filter

Based on the developed formalism, we now present an end-to-end differentiable particle sampler computing a marginal log-likelihood $\mathcal{L}$. We consider a hidden Markov model with random states $X_1, \ldots, X_n$ as specified by a stochastic program $X_1(\theta)$ giving the starting value depending on unknown parameters θ and consecutive states given by pointwise differentiable stochastic programs $X_i(x_{i-1}, \theta)$ depending on the previous state $x_{i-1} = X_{i-1}$ (Markov property) and the parameters. In general, we allow probability densities $p(x_1 \mid \theta)$ of X_1 and transition probability densities $p(x_i \mid x_{i-1}, \theta)$ to depend arbitrarily on the parameters θ. This latent process $X_1, \ldots, X_n$ is indirectly observed as the process $Y_1, \ldots, Y_n$ with observations $Y_i(x_i)$ depending on $x_i = X_i$ which are specified to have smooth conditional probability densities $p(y_i \mid x_i, \theta)$ depending only on x_i and θ. The likelihood of θ given the observations is in general not tractable, but a particle filter can be used to compute an estimate of the likelihood. As a concrete example, we consider the following system with a d-dimensional latent process,

$$
\begin{aligned}
x_i &= \Phi x_{i-1} + w_i && \text{with } w_i \sim \text{Normal}(0, Q), \\
y_i &= x_i + \nu_i && \text{with } \nu_i \sim \text{Normal}(0, R),
\end{aligned}
$$

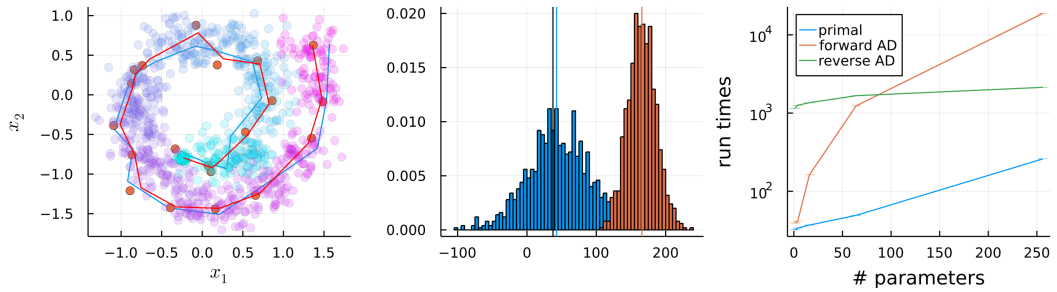


Figure 5: Unbiased differentiable particle sampler using smoothed stochastic derivatives. **Left:** Latent path (blue) and observations (orange) for a particular realization of the process in $d = 2$ dimensions, including a recursively applied Kalman filter estimator (red) and the particles from the particle filter (going from blue at $n = 1$ to pink at $n = 20$). **Middle:** The ground-truth value of the derivative for $d = 2$ calculated by differentiating the Kalman filter is marked by a black line. The blue distribution corresponds to the derivative computed with smoothed stochastic derivatives. The orange distribution represents the biased derivative approach, where the resampling step is not differentiated. The means of the two distributions are highlighted by a line in the respective color. In both cases, we use 1000 samples. **Right:** Averaged run times in ms comparing forward and reverse-mode AD for increasing number of parameters. The number of parameters scales quadratically with the system dimension.

where $Q = 0.02 \cdot \mathbb{1}_{d\times d}$, $R = 0.01 \cdot \mathbb{1}_{d\times d}$, $x_1 \sim \mathrm{Normal}(\mu, 0.001 \cdot \mathbb{1}_{d\times d})$, $\mu \sim \mathrm{Normal}(0, \mathbb{1}_{d\times d})$ is a random initial position, and Φ is a d-dimensional rotation matrix. We wish to differentiate with respect to each entry of Φ, i.e. $\theta = \mathrm{vec}(\Phi)$.

Given n observations $y_1 = Y_1, \ldots, y_n = Y_n$ of the hidden Markov model, a particle filter allows one to estimate the likelihood $\mathcal{L}$ of these observations by simulating a number of "particles" following the dynamical model, and repeatedly performing *resampling* steps to pick the particles that match the observations. We give a more complete introduction to particle samplers in Appendix C, following our terminology, but what is most important for our purposes is that the resampling step assigns an importance weight to each particle that is used in a purely linear way to produce the ultimate likelihood estimate. Therefore, we may differentiate each weight using a smoothed stochastic derivative, as formalized in Appendix B.6 and Appendix C.

Particle samplers are a powerful, generic approach to this problem, but our particular choice of system allows the calculation of a ground-truth gradient of the log-likelihood with respect to Φ by differentiating the Kalman filter algorithm. For our computations, we fix the number of observations $n = 20$. Fig. 5 visualizes the latent process, the observations, as well as the Kalman and the particle filter trajectories in $d = 2$ dimensions. Our estimator agrees with the ground-truth value from the Kalman filter, unlike biased estimators [29, 30, 31, 32] that neglect the resampling step' contribution to the derivative or perform it with entropy-regularized optimal transport (Fig. 5). Furthermore, thanks to the similarity of smoothed stochastic derivatives and classical derivatives, we can also use reverse-mode instead of forward-mode AD. This is convenient, as eventually reverse-mode AD becomes superior than forward-mode AD for functions f from $\mathbb{R}^n$ to $\mathbb{R}^m$ with $m \gg n$ [2]. For our stochastic program, we observe reverse-mode AD to perform better for more than ≈ 100 parameters.

4 Limitations and outlook

We have presented a method for unbiased AD of programs with discrete randomness, which, we have argued, is a natural generalization of the often-used reparameterization trick. However, more work will need to be done to turn our software demonstration `StochasticAD.jl` into an AD system capable of handling the full complexity of applications such as machine learning. This includes support for if statements with discrete random input; currently, such branches need to be rewritten using array indexing, which is supported. Further, an interesting direction for future work is to automatically handle constructs which *deterministically* turn a continuous random quantity into a discrete one, such as an inequality comparison `X(p) > 0`, and furthermore handle constructs such as while loops based on such comparisons. In a different direction, going beyond smoothing for reverse-mode differentiation, and ideally achieving the unbiasedness of stochastic triples while remaining coupled to the primal, is an important open problem for large-scale applications.

References

[1] Andreas Griewank et al. On automatic differentiation. *Mathematical Programming: recent developments and applications*, 6(6):83–107, 1989.

[2] Atilim Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *Journal of Machine Learning Research*, 18:1–43, 2018.

[3] George Corliss, Christele Faure, Andreas Griewank, Laurent Hascoet, and Uwe Naumann. *Automatic differentiation of algorithms: from simulation to optimization*. Springer Science & Business Media, 2002.

[4] Søren Asmussen and Peter W Glynn. *Stochastic simulation: algorithms and analysis*, volume 57. Springer, 2007.

[5] Shakir Mohamed, Mihaela Rosca, Michael Figurnov, and Andriy Mnih. Monte Carlo gradient estimation in machine learning. *J. Mach. Learn. Res.*, 21(132):1–62, 2020.

[6] Peter W Glynn. Likelihood ratio gradient estimation for stochastic systems. *Communications of the ACM*, 33(10):75–84, 1990.

[7] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

[8] Jack PC Kleijnen and Reuven Y Rubinstein. Optimization and sensitivity analysis of computer simulation models by the score function method. *European Journal of Operational Research*, 88(3):413–427, 1996.

[9] G Ch Pflug. Sampling derivatives of probabilities. *Computing*, 42(4):315–328, 1989.

[10] Bernd Heidergott and FJ Vázquez-Abad. Measure-valued differentiation for Markov chains. *Journal of Optimization Theory and Applications*, 136(2):187–209, 2008.

[11] Paul Glasserman and Yu-Chi Ho. *Gradient estimation via perturbation analysis*, volume 116. Springer Science & Business Media, 1991.

[12] Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. *arXiv:1312.6114*, 2013.

[13] John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel. Gradient estimation using stochastic computation graphs. *Advances in Neural Information Processing Systems*, 28, 2015.

[14] George Tucker, Andriy Mnih, Chris J Maddison, John Lawson, and Jascha Sohl-Dickstein. REBAR: Low-variance, unbiased gradient estimates for discrete latent variable models. *Advances in Neural Information Processing Systems*, 30, 2017.

[15] Will Grathwohl, Dami Choi, Yuhuai Wu, Geoffrey Roeder, and David Duvenaud. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. *arXiv:1711.00123*, 2017.

[16] Peter W Glynn and Roberto Szechtman. Some new perspectives on the method of control variates. In *Monte Carlo and Quasi-Monte Carlo Methods 2000*, pages 27–49. Springer, 2002.

[17] Bernd Heidergott, Felisa J Vázquez-Abad, and Warren Volk-Makarewicz. Sensitivity estimation for Gaussian systems. *European Journal of Operational Research*, 187(1):193–207, 2008.

[18] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with Gumbel-softmax. *arXiv:1611.01144*, 2016.

[19] Wei-Bo Gong and Yu-Chi Ho. Smoothed (conditional) perturbation analysis of discrete event dynamical systems. *IEEE Transactions on Automatic Control*, 32(10):858–866, 1987.

[20] Paul Glasserman and W-B Gong. Smoothed perturbation analysis for a class of discrete-event systems. *IEEE Transactions on Automatic Control*, 35(11):1218–1230, 1990.

[21] Emile Krieken, Jakub Tomczak, and Annette Ten Teije. Storchastic: A framework for general stochastic automatic differentiation. *Advances in Neural Information Processing Systems*, 34, 2021.

[22] Emile Krieken, Jakub Tomczak, and Annette Ten Teije. Openreview: Storchastic: A framework for general stochastic automatic differentiation.

[23] Luc Devroye. Nonuniform random variate generation. *Handbooks in operations research and management science*, 13:83–121, 2006.

[24] J. Revels, M. Lubin, and T. Papamarkou. Forward-mode automatic differentiation in Julia. *arXiv:1607.07892 [cs.MS]*, 2016.

[25] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):65–98, 2017.

[26] Eric Silverman, Umberto Gostoli, Stefano Picascia, Jonatan Almagor, Mark McCann, Richard Shaw, and Claudio Angione. Situating agent-based modelling in population health research. *Emerging Themes in Epidemiology*, 18(1):1–15, 2021.

[27] Lily A Chylek, Leonard A Harris, Chang-Shung Tung, James R Faeder, Carlos F Lopez, and William S Hlavacek. Rule-based modeling: a computational approach for studying biomolecular site dynamics in cell signaling systems. *Wiley Interdisciplinary Reviews: Systems Biology and Medicine*, 6(1):13–36, 2014.

[28] James R Faeder, Michael L Blinov, and William S Hlavacek. Rule-based modeling of biochemical systems with bionetgen. In *Systems biology*, pages 113–167. Springer, 2009.

[29] Rico Jonschkowski, Divyam Rastogi, and Oliver Brock. Differentiable particle filters: End-to-end learning with algorithmic priors. *arXiv:1805.11122*, 2018.

[30] Adrien Corenflos, James Thornton, George Deligiannidis, and Arnaud Doucet. Differentiable particle filtering via entropy-regularized optimal transport. In *International Conference on Machine Learning*, pages 2100–2111. PMLR, 2021.

[31] Conor Rosato, Paul Horridge, Thomas B Schön, and Simon Maskell. Efficient learning of the parameters of non-linear models using differentiable resampling in particle filters. *arXiv:2111.01409*, 2021.

[32] Adam Ścibior and Frank Wood. Differentiable particle filtering without modifying the forward pass. *arXiv:2106.10314*, 2021.

Table 1: Probability mass functions and cumulative distributions of integer-valued stochastic programs, where x is assumed in the domain of $X(p)$.

$X(p)$	Parameter	$\mathbb{P}(X(p) = x)$	$\mathbb{P}(X(p) \leq x)$	$\partial_p \mathbb{P}(X(p) \leq x)$
$\mathrm{Ber}(p)$	Probability	$px + (1-p)(1-x)$	$(1-p) + p\,[\![x = 1]\!]$	$-1 + x$
$\mathrm{Geo}(p)$	Probability	$p(1-p)^x$	$1 - (1-p)^x$	$x(1-p)^{x-1}$
$\mathrm{Pois}(p)$	Rate	$p^x/(e^p x!)$	$\sum_{k=0}^{x} e^{-p} p^k / k!$	$-p^{x+1}/(e^p(x+1)!)$

In the supplement, we:

- provide a number of examples of stochastic derivatives and present hand-worked examples of how they compose (Appendix A),
- prove our formal statements regarding stochastic derivatives, and state and prove formal statements regarding smoothed stochastic derivatives (Appendix B),
- provide an introduction to the particle filter methodology, and show how smoothed stochastic derivatives leads to unbiased differentiation of the resampling step (Appendix C),
- provide details about the hardware used for the experiments of the main text and the software dependencies of `StochasticAD.jl` (Appendix D).

A Examples of stochastic derivatives

In this section, we present examples of stochastic derivatives for a number of stochastic programs, recalling Definition 2.1. We consider both elementary programs (e.g. a Bernoulli variable) for which Theorem 2.2 applies, as well as general programs (e.g. a two-step random walk).

Example A.1 (Integer-valued stochastic programs)**.** The Bernoulli, Binomial, Geometric, and Poisson distributions are all discrete, and in fact (nonnegative) integer-valued. Suppose $X(p)$ follows one of these distributions, parameterized using the inversion method [23] over the interval $[0, 1]$. We present a worked derivation of the stochastic derivative of $X(p)$.

Let $X(p)$ have stochastic derivative (δ, w, Y) at p. The program is integer-valued, and hence $\delta = 0$, i.e. the point-wise derivative $\partial_p X(p)(z)$ is 0 for all z. Further, for small enough ε, $\mathrm{d}X(\varepsilon) \in \{-1, 0, 1\}$, and thus $Y \in \{X(p) - 1, X(p) + 1\}$. This comes from the use of the inversion method: the only finite perturbations with non-zero probability switch to the next smallest or next largest discrete output of the program.

For a small change ε in the program input, let $A_+(\varepsilon)$ denote the event $X(p + \varepsilon) = x + 1$ (switching to the next largest output), and let $A_-(\varepsilon)$ denote the event $X(p - \varepsilon) = x - 1$ (switching to the next smallest output). Then, $A(\varepsilon)$ is $A_-(\varepsilon) \cup A_+(\varepsilon)$. We derive,

$$\mathbb{P}\left(A_+(\varepsilon) | X(p) = x\right) = \frac{\mathbb{P}(X(p + \varepsilon) = x + 1 \text{ and } X(p) = x)}{\mathbb{P}(X(p) = x)} \tag{A.1}$$

$$= \frac{\max\left(0, -\varepsilon \cdot \partial_p P(X(p) \leq x)\right)}{P(X(p) = x)} + \mathcal{O}(\varepsilon^2). \tag{A.2}$$

We are interested in the derivative w_+ of this probability at $\varepsilon = 0$,

$$w_+ = \lim_{\varepsilon \to 0} \frac{\mathbb{P}\left(A_+(\varepsilon) | X(p) = x\right)}{\varepsilon} = \frac{-\partial_p \mathbb{P}(X(p) \leq x)}{\mathbb{P}(X(p) = x)} \cdot [\![\mathrm{sgn}(\varepsilon) \cdot \partial_p \mathbb{P}(X(p) \leq x) < 0]\!], \tag{A.3}$$

where $[\![P]\!]$ is the Iverson bracket of a condition P.

Informally, one can think of w_+ as the scaling factor of the "infinitesimal" probability that $X(p+\varepsilon) = x + 1$, i.e. this probability is $w_+\varepsilon + \mathcal{O}(\varepsilon^2)$. Note that the sign of ε matters, where $\varepsilon > 0$ represents the right stochastic derivative and $\varepsilon < 0$ the left stochastic derivative. This is reflected through the inclusion of $\mathrm{sgn}(\varepsilon)$ (an abuse of notation representing the direction of the limit) on the RHS of

Table 2: Stochastic derivatives of integer-valued stochastic programs

	Conditionally on $X(p) = x$	
$X(p)$	w_-	w_+
$\mathrm{Ber}(p)$	$x/p \cdot [\![x = 1 \text{ and } \varepsilon < 0]\!]$	$1/(1-p) \cdot [\![x = 0 \text{ and } \varepsilon > 0]\!]$
$\mathrm{Bin}(n,p)$	$n/p \cdot [\![x > 0 \text{ and } \varepsilon < 0]\!]$	$(n-x)/(1-p) \cdot [\![x < n \text{ and } \varepsilon > 0]\!]$
$\mathrm{Geo}(p)$	$x/\left(p(1-p)\right) \cdot [\![x > 0 \text{ and } \varepsilon > 0]\!]$	$(x+1)/p \cdot [\![\varepsilon < 0]\!]$
$\mathrm{Pois}(p)$	$x/p \cdot [\![x > 0 \text{ and } \varepsilon < 0]\!]$	$[\![\varepsilon > 0]\!]$

Eq. (A.3): the magnitude of ε has cancelled out in the limit but a dependence on its sign remains. In an analogous fashion, we can derive

$$w_- = \lim_{\varepsilon \to 0} \frac{\mathbb{P}\left(A_-(\varepsilon) | X(p) = x\right)}{\varepsilon} = \frac{\partial_p \mathbb{P}(X(p) \leq x - 1)}{\mathbb{P}(X(p) = x)} \cdot [\![\mathrm{sgn}(\varepsilon) \cdot \partial_p \mathbb{P}(X(p) \leq x - 1) > 0]\!] \,. \tag{A.4}$$

Putting these two cases together, we may write,

$$\begin{aligned} w &= \lim_{\varepsilon \to 0} \mathbb{P}\left(A(\varepsilon) | X(p) = x\right) && \text{(A.5)} \\ &= \lim_{\varepsilon \to 0} \mathbb{P}\left(A_+(\varepsilon) | X(p) = x\right) + \lim_{\varepsilon \to 0} \mathbb{P}\left(A_-(\varepsilon) | X(p) = x\right) && \text{(A.6)} \\ &= w_+ + w_- . && \text{(A.7)} \end{aligned}$$

and express the probability distribution of Y as

$$\mathbb{P}\left(Y = x + 1 \mid X(p) = x\right) = \frac{w_+}{w_+ + w_-}, \tag{A.8}$$

$$\mathbb{P}\left(Y = x - 1 \mid X(p) = x\right) = \frac{w_-}{w_+ + w_-}. \tag{A.9}$$

Table 2 lists the weights w_+ and w_- for the Bernoulli, Binomial, Poisson, and Geometric distributions, derived using the probability distributions given in Table 1. Using w_+ and w_-, we can easily express w and Y as above. Note that w_+ and w_- for a Binomial variable may be derived from the composition of Bernoulli variables, by writing the primal evaluation $X(p) = x$ as the sum of x Bernoulli successes and $n - x$ Bernoulli failures.

Note that the stochastic derivative for a binomial variable has weight w of order n and therefore samples from the estimator $w \cdot (Y - X(p))$ of $\partial_p \mathbb{E}[X(p)]$ for $X(p) \sim \mathrm{Bin}(n,p)$ have variance of order n. In contrast, the score method gives an estimator

$$\frac{X(p)(X(p) - np)}{p(1-p)}$$

which is also unbiased, but its variance

$$n^3 \frac{p}{1-p} + \mathcal{O}(n^2)$$

is growing with cubic rate in n.

Example A.2 (Differentiably parameterized continuous stochastic programs)**.** For continuous stochastic programs, our method reduces to the standard reparameterization trick. Specifically, for a differentiably parameterized continuous random variable, the stochastic derivative w.r.t a parameter p of the distribution has the form $(X'(p), 0, 0)$. For instance, the stochastic derivative of $X(p) \sim \mathrm{Exp}(p)$ w.r.t. p, the scale of the exponential distribution, is

$$\left(\frac{x}{p}, 0, 0\right) \tag{A.10}$$

conditionally on $X(p) = x$. Such rules are already implemented in the Julia AD ecosystem in `Distributions.jl` [1] and `DistributionsAD.jl` [2], and may be used without modification as the continuous case is a special case of our formalism.

Example A.3 (Categorical variable)**.** Consider a categorical variable $\text{Categorical}_{\mathbf{a}}(p_1, p_2, p_3, \ldots, p_n)$ assuming fixed outputs $\mathbf{a} = a_1, a_2, \ldots, a_n$ with probabilities $p_1, p_2, p_3, \ldots, p_n$. Assume that $p_1, p_2, p_3, \ldots, p_n$ implicitly depend on a single parameter p, to fit our definition of stochastic derivative w.r.t. a single parameter, and let $X(p) \sim \text{Categorical}_{\mathbf{a}}(p_1, p_2, p_3, \ldots, p_n)$.

The categorical variable is univariate and its discrete outputs do not depend on p, and hence its stochastic derivative is of the form $(0, w, Y)$ with $Y \in \{a_{x-1}, a_{x+1}\}$. Similar logic to Example A.1 yields

$$w \cdot \mathbb{P}\left(Y = a_{x+1}\right) = w_+ = \frac{|\sum_{i=1}^{x} \partial_p p_i|}{p_x} \cdot \left[\!\left[\varepsilon \cdot \sum_{i=1}^{x} \partial_p p_i < 0\right]\!\right], \tag{A.11}$$

$$w \cdot \mathbb{P}\left(Y = a_{x-1}\right) = w_- = \frac{\left|\sum_{i=1}^{x-1} \partial_p p_i\right|}{p_x} \cdot \left[\!\left[\varepsilon \cdot \sum_{i=1}^{x-1} \partial_p p_i > 0\right]\!\right], \tag{A.12}$$

with w and Y given from w_+ and w_- as in Example A.1. Note that $a_1, a_2, \ldots, a_n$ can be arbitrary discrete objects, such as arrays or strings, as the categorical variable may represent an intermediate value of the program which is ultimately converted into a number. In handling such cases, one must be careful to speak in terms of *alternate* outputs rather than perturbations to the output. Our formalism of stochastic derivatives can already accomodate this idea, as Y takes on the alternate values rather than the values of the perturbations.

We now consider some general stochastic programs. Their stochastic derivatives derive automatically from composition and need not be hand-derived, but we do so for clarity.

Example A.4 (Bernoulli plus Exponential)**.** Consider the program,

$$X_1(p) \sim \text{Ber}(p) \tag{A.13}$$

$$X_2(p) \sim \text{Exp}(p) \tag{A.14}$$

$$X(p) \sim X_1(p) + X_2(p) \sim \text{Ber}(p) + \text{Exp}(p). \tag{A.15}$$

Then, $X(p)$ has right (left) stochastic derivative,

$$\left(\frac{x_2}{p}, w_1, Y_1\right) \tag{A.16}$$

conditionally on $X_1(p) = x_1$ and $X_2(p) = x_2$, where w_1 and Y_1 are given conditionally on the Bernoulli output $X_1(p) = x_1$ for $\varepsilon > 0$ ($\varepsilon < 0$) by Table 2.

Example A.5 (Cubing a geometric)**.** Consider the program,

$$X_1(p) = \text{Geo}(p), \tag{A.17}$$

$$X(p) = X_1(p)^3. \tag{A.18}$$

We know from Example A.1 that $X_1(p)$ has right (left) stochastic derivative $(0, w_1, Y_1)$ with w_1 and Y_1 given by Table 2 for $\varepsilon > 0$ ($\varepsilon < 0$). In particular, $Y_1 \in \{x_1 - 1, x_1 + 1\}$ conditionally on $X_1(p) = x_1$. Thus, intuitively, when propagating through the cube function we only care about the discretely spaced alternate values $(x_1 - 1)^3$ and $(x_1 + 1)^3$; the conventional derivative $3x_1^2$ is irrelevant as the input to the cube function is integer-valued. Indeed, by Proposition 2.4 the stochastic derivative of $X(p)$ reads

$$(0, w_1, Y_1^3), \tag{A.19}$$

where $Y_1^3 \in \{(x_1 - 1)^3, (x_1 + 1)^3\}$.

Example A.6 (Parameter-scaled Bernoulli)**.** Consider the program,

$$X_1(p) = \text{Ber}(p), \tag{A.20}$$

$$X(p) = p \cdot X_1(p). \tag{A.21}$$

By Proposition 2.4, the stochastic derivative of $X(p)$ reads,

$$(X(p), w_1, pY_1), \tag{A.22}$$

where w_1 and Y_1 are given for $X_1(p)$ by Table 2.

Example A.7 (Two-step random walk)**.** Consider the program,

$$X_1(p) \sim \mathrm{Ber}(p), \tag{A.23}$$

$$X_2(x_1) \sim \begin{cases} \mathrm{Ber}(p) & \text{if } x_1 = 0 \\ \mathrm{Ber}(2p) & \text{if } x_1 = 1. \end{cases} \tag{A.24}$$

$$X(p) \sim X_1(p) + (X_2 \circ X_1)(p). \tag{A.25}$$

This represents a two-step random walk with steps given by $X_1(p)$ and $X_2(p)$, where there is a transition $0 \to 1$ with probability p, a transition $1 \to 2$ with probability $2p$, and self loops otherwise. In this example, let us focus on a particular primal evaluation where $X_1(p) = 0$ and $X_2(p) = 0$, so that $X(p) = 0$, and consider only the right stochastic derivative for simplicity. Conditionally on $X_1(p) = 0$, $X_1(p)$ has right stochastic derivative

$$\left(0, \frac{1}{1-p}, 1\right), \tag{A.26}$$

as given by Table 2. Now, conditionally on $X_2(p) = 0$, $X_2(0)$ has right stochastic derivative w.r.t. x_1,

$$\left(0, \frac{1}{1-p}, 1\right). \tag{A.27}$$

Let us now turn our attention to the stacked program $[X_1(p); (X_2 \circ X_1)(p)]$. By Proposition 2.4, the stacked program has right stochastic derivative $(0, w, Y_{12})$ where conditionally on $X_1(p) = 0$ and $X_2(0) = 0$,

$$Y_{12} \quad = \quad \begin{cases} [1; X_2(1)] & \text{with probability} \quad 1/2, \\ [0; 1] & \text{with probability} \quad 1/2, \end{cases} \tag{A.28}$$

and $w = \frac{2}{1-p}$. Note that conditionally on $X_2(0) = 0$, $X_2(1)$ has a chance $\frac{1-2p}{1-p}$ of also being 0, but a $\frac{p}{1-p}$ chance of flipping to 1, so that the first case expands into two events

$$Y_{12} \quad = \quad \begin{cases} [1; 1] & \text{with probability} \quad \frac{p}{2(1-p)}, \\ [1; 0] & \text{with probability} \quad \frac{1-2p}{2(1-p)}, \\ [0; 1] & \text{with probability} \quad 1/2. \end{cases} \tag{A.29}$$

Finally, the expression for $X(p)$ may be thought of as a unary function of this stacked program, with stochastic derivative $(0, w, Y)$ where conditionally on $X_1(p) = 0$ and $X_2(0) = 0$

$$Y \quad = \quad \begin{cases} 2 & \text{with probability} \quad \frac{p}{2(1-p)}, \\ 1 & \text{with probability} \quad \frac{1-2p}{2(1-p)}, \\ 1 & \text{with probability} \quad 1/2. \end{cases} \tag{A.30}$$

B Proofs

B.1 Preliminaries

Let $X(p)$ denote a *stochastic program*, formally defined as a stochastic process with index set the finite closed interval $I = [a, b]$ and values in the Euclidean space E. Let $p \in I$ be the parameter of the process. As in the main text, we use the shorthand

$$\mathrm{d}X(\varepsilon) = X(p + \varepsilon) - X(p). \tag{B.1}$$

Due to asymmetry of stochastic derivatives, a number of statements and propositions have equivalent forms for right and left stochastic derivatives. In this case, we use the notation $\varepsilon \to 0^{+/-}$ to indicate that statements and proofs for right and left stochastic derivatives can both be read off by choosing the appropriate side of the limit for all expressions.

For clarity, we remark upon the (standard) notation,

$$W = \mathbb{E}\left[U \mid V\right], \tag{B.2}$$

where U and V are both random variables defined on a sample space Z. In this setting, W is itself also a random variable defined on Z. In particular, for a particular sample $z \in Z$, $W(z)$ is the conditional expectation $\mathbb{E}[A \mid B = B(z)]$. In the language of σ-algebras, $\mathbb{E}[U \mid V]$ is equivalent to $\mathbb{E}[U \mid \sigma(V)]$, where $\sigma(V)$ is the sub-σ-algebra generated by V.

Finally, we use the notation S^c to denote the complement of a set S.

B.2 Stochastic derivatives of elementary stochastic programs

In Theorem 2.2, w can be understood as the derivative of the probability that $X(p+\varepsilon)$ and $X(p)$ vary by more than infinitesimal amount. The following event quantifies this,

$$A(\varepsilon) = \{|\mathrm{d}X(\delta) - \delta \cdot X'(p)| > |\delta| \text{ for some } \delta \text{ between } 0 \text{ and } \varepsilon\}. \tag{B.3}$$

Outside the event $A(\varepsilon)$, the sensitivity is well-described by the reparameterization trick:

Proposition B.1. *If $f\colon E \to \mathbb{R}$ is smooth with bounded derivative and $X(p)$ is almost surely differentiable having derivative $X'(p)$ with bounded expectation, then*

$$\lim_{\varepsilon \to 0} \mathbb{E}\left[\frac{f(X(p+\varepsilon)) - f(X(p))}{\varepsilon} \mathbf{1}_{A^c(\varepsilon)}\right] = \mathbb{E}\, f'(X(p))X'(p).$$

Proof. Because $f \circ X$ is almost surely differentiable,

$$\lim_{\varepsilon \to 0} \mathbf{1}_{A^c(\varepsilon)} = 1. \tag{B.4}$$

Therefore, it also holds that

$$\lim_{\varepsilon \to 0} \frac{f(X(p+\varepsilon)) - f(X(p))}{\varepsilon} = f'(X(p))X'(p). \tag{B.5}$$

Furthermore,

$$\begin{aligned}
\frac{|f(X(p+\varepsilon)) - f(X(p))|}{\varepsilon} \mathbf{1}_{A^c(\varepsilon)} &\le \frac{1}{\varepsilon}\|f'\|_\infty |\mathrm{d}X(\varepsilon)| \mathbf{1}_{A^c(\varepsilon)} \\
&\le \frac{1}{\varepsilon}\|f'\|_\infty \left(|\mathrm{d}X(\varepsilon) - \varepsilon X'(p)| + |\varepsilon X'(p)|\right) \mathbf{1}_{A^c(\varepsilon)} \\
&\le \|f'\|_\infty (1 + |X'(p)|) =: G
\end{aligned}$$

By assumption $\mathbb{E}\, G < \infty$. The proposition follows by bounded convergence. □

We now give a technical condition under which Theorem 2.2 holds, in which we characterize w as the derivative of the conditional probability of $A(\varepsilon)$ given $X(p)$, and Y as realization of the weak limit of the conditional distribution of $X(p+\varepsilon)$ given $X(p)$ and $A(\varepsilon)$.

Assumption B.2. We assume that:

- For the quantity
$$w(\varepsilon) = \mathbb{P}\left(A(\varepsilon) \mid X(p)\right),$$
the following limit converges in mean,
$$w = \lim_{\varepsilon \to 0^{+/-}} \frac{w(\varepsilon)}{\varepsilon}. \tag{B.6}$$
where $\mathbb{E} w < \infty$.

- The conditional distribution of $X(p+\varepsilon)$ given $X(p)$ and $A(\varepsilon)$ converges in distribution to a random variable Y. Specifically, Y must satisfy
$$\mathbb{E}\left[w \cdot f(Y)\right] = \lim_{\varepsilon \to 0^{+/-}} \mathbb{E}\, w \int_E f(y) \mathbb{P}\left(X(p+\varepsilon) \in \mathrm{d}y \mid X(p) = x, A(\varepsilon)\right), \tag{B.7}$$
for all bounded smooth functions $f\colon E \to \mathbb{R}$ with bounded derivative.

If that $Z \sim \mathrm{U}(0,1]$ and $X(p) = f(p, Z)$ is a continuous differentiable random variable or a discrete random variable with f the left continuous inverse of its distribution function F_p and if the boundaries of the interval $(a_x(p), b_x(p)] = \{z \colon f(p,z) = x\}$ are differentiable in p for all values x with $\mathbb{P}(X(p) = x) > 0$, Assumption B.2 holds. Given Assumption B.2, we can now well-characterize the sensitivity of $X(p)$ given that $A(\varepsilon)$ occurs:

Proposition B.3. *If $f \colon E \to \mathbb{R}$ is bounded and smooth with bounded derivative, then for the event A given by Eq.* (B.3) *and the random variables w and Y constructed by Assumption B.2, it holds that*

$$\lim_{\varepsilon \to 0^{+/-}} \mathbb{E}\left[\frac{f(X(p+\varepsilon)) - f(X(p))}{\varepsilon} \mathbf{1}_{A(\varepsilon)}\right] = \mathbb{E}\left[w\left(f(Y) - f(X(p))\right)\right].$$

Proof. Write

$$\begin{aligned}\mathbb{E}\left[\frac{f(X(p+\varepsilon)) - f(X(p))}{\varepsilon} \mathbf{1}_{A(\varepsilon)}\right] &= \mathbb{E}\left[w \cdot \left(f(X(p+\varepsilon)) - f(X(p))\right) \frac{\mathbf{1}_{A(\varepsilon)}}{w(\varepsilon)}\right] \\ &\quad + \mathbb{E}\left[\left(\frac{w(\varepsilon)}{\varepsilon} - w\right) \cdot \left(f(X(p+\varepsilon)) - f(X(p))\right) \frac{\mathbf{1}_{A(\varepsilon)}}{w(\varepsilon)}\right].\end{aligned}$$

Tackling the first term, by the tower property of conditional expectation,

$$\begin{aligned}&\mathbb{E}\left[w \cdot \left(f(X(p+\varepsilon)) - f(X(p))\right) \frac{\mathbf{1}_{A(\varepsilon)}}{w(\varepsilon)}\right] \\ &= \mathbb{E}\left[\mathbb{E}\left[w \cdot \left(f(X(p+\varepsilon)) - f(X(p))\right) \frac{\mathbf{1}_{A(\varepsilon)}}{w(\varepsilon)} \;\middle|\; X(p)\right]\right] \\ &= \mathbb{E}\left[w \cdot \frac{\mathbb{E}\left[f\left(X(p+\varepsilon)\right) \mathbf{1}_{A(\varepsilon)} \mid X(p)\right]}{w(\varepsilon)}\right] \\ &\quad - \mathbb{E}\left[w \cdot f\left(X(p)\right) \cdot \frac{\mathbb{E}\left[\mathbf{1}_{A(\varepsilon)} \mid X(p)\right]}{w(\varepsilon)}\right],\end{aligned}$$

which converges to $\mathbb{E}\left[w \cdot f(Y)\right] - \mathbb{E}\left[w \cdot f\left(X(p)\right)\right]$ as $\varepsilon \to 0^{+/-}$ by Assumption B.2 and the fact that $\mathbb{E}[\mathbf{1}_{A(\varepsilon)} | X(p)] = \mathbb{P}(A(\varepsilon) | X(p)) = w(\varepsilon)$. Here, the key manipulation has been to introduce a conditional expectation on $X(p)$, and hoist w, $w(\varepsilon)$, and $f(X(p))$ outside of it as they are deterministic functions of $X(p)$. Now, we may bound the magnitude of the second term as

$$\begin{aligned}&\left|\mathbb{E}\left[\left(\frac{w(\varepsilon)}{\varepsilon} - w\right) \cdot \left(f(X(p+\varepsilon)) - f(X(p))\right) \frac{\mathbf{1}_{A(\varepsilon)}}{w(\varepsilon)}\right]\right| \\ &\quad \leq 2\|f\|_\infty \cdot \mathbb{E}\left[\mathbb{E}\left[\left|\left(\frac{w(\varepsilon)}{\varepsilon} - w\right) \frac{\mathbf{1}_{A(\varepsilon)}}{w(\varepsilon)}\right| \;\middle|\; X(p)\right]\right] \\ &\quad \leq 2\|f\|_\infty \cdot \mathbb{E}\left[\mathbb{E}\left[\left|\frac{w(\varepsilon)}{\varepsilon} - w\right|\right] \cdot \mathbb{E}\left[\frac{\mathbf{1}_{A(\varepsilon)}}{w(\varepsilon)} \;\middle|\; X(p)\right]\right] \\ &\quad \leq 2\|f\|_\infty \cdot \mathbb{E}\left[\left|\frac{w(\varepsilon)}{\varepsilon} - w\right|\right] \cdot 1,\end{aligned}$$

which approaches 0 as $\varepsilon \to 0^{+/-}$ by the convergence in mean of $w(\varepsilon)/\varepsilon$ to w. Putting the limits of these two terms together, we obtain the claim. □

Now, recall

Definition 2.1. Suppose $X(p)$ is a stochastic program with real input, whose output belongs to a Euclidean space E. We say that the triple of random variables (δ, w, Y), with $\delta \in E$, weight $w \in \mathbb{R}$ and alternative outcome $Y \in E$ is a *stochastic derivative* of X at the input p if for all smooth bounded real functions f,

$$\frac{\partial}{\partial p}\mathbb{E}\left[f(X(p))\right] = \mathbb{E}\left[f'(X(p)) \cdot \delta + w \cdot \left(f(Y) - f(X(p))\right)\right]. \tag{2.4}$$

We can now provide a proof of Theorem 2.2.

Theorem 2.2. Given a stochastic program $X(p)$ satisfying Assumption B.2 and almost surely differentiable having derivative $X'(p)$ with bounded expectation, there exists a stochastic derivative $(X'(p), w_R, Y_R)$ of p with $w_R \geq 0$, which we call the *right* stochastic derivative. Likewise there exists a triple $(X'(p), w_L, Y_L)$ with $w_L \leq 0$ giving a *left* stochastic derivative.

Proof. Construct w_R and Y_R by Assumption B.2 applied with the right-sided limit $\varepsilon \to 0^+$, and construct w_L and Y_L by Assumption B.2 applied with the left-sided limit $\varepsilon \to 0^-$. Now by Proposition B.1 and Proposition B.3,

$$\begin{aligned}\lim_{\varepsilon \to 0^{+/-}} \frac{\mathbb{E}\left[f(X(p+\varepsilon)) - f(X(p))\right]}{\varepsilon} &= \lim_{\varepsilon \to 0^{+/-}} \mathbb{E}\left[\frac{f(X(p+\varepsilon)) - f(X(p))}{\varepsilon}\left(\mathbf{1}_{A^c(\varepsilon)} + \mathbf{1}_{A(\varepsilon)}\right)\right] \\ &= \mathbb{E}\left[f'(X(p))X'(p)\right] + \mathbb{E}\left[w_{R/L}\left(f(Y_{R/L}) - f(X(p))\right)\right].\end{aligned}$$

Thus, the triples $(X'(p), w_R, Y_R)$ and $(X'(p), w_L, Y_L)$ are stochastic derivatives of $X(p)$ by Proposition B.3. Note that $w_R \geq 0$ and $w_L \leq 0$ by Eq. (B.6) and the non-negativity of $w(\varepsilon)$. So $(X'(p), w_R, Y_R)$ is a right stochastic derivative of $X(p)$ at p and $(X'(p), w_L, Y_L)$ is a left stochastic derivative of $X(p)$ at p, as desired. □

B.3 Composition of stochastic derivatives

Recall,

Proposition 2.4 (Chain rule)**.** Consider independent stochastic programs X_1 and X_2 and their composition $X_2 \circ X_1$. Suppose that X_1 has a stochastic derivative at p given by (δ_1, w_1, Y_1) and X_2 has a stochastic derivative (δ_2, w_2, Y_2) in the direction δ_1 given conditionally on its input $X_1(p)$, as constructed by Theorem 2.2. Then the stacked program $[X_1; X_2 \circ X_1]$ has a stochastic derivative given by (δ, w, Y) where $\delta = [\delta_1; |\delta_1|\delta_2]$ and

$$Y = \begin{cases} [Y_1; X_2(Y_1)] & \text{with probability} \quad \dfrac{w_1}{w_1 + |\delta_1| \cdot w_2}, \\ [X_1; Y_2] & \text{with probability} \quad \dfrac{|\delta_1| \cdot w_2}{w_1 + |\delta_1| \cdot w_2}, \end{cases} \tag{2.7}$$

and $w = w_1 + |\delta_1| \cdot w_2$.

Proof. Let $A_1(\varepsilon)$ denote the event (B.3) for X_1 at p. Since X_2 may have multivariate input, conditionally on $X_1(p)$ define the program $X_2(X_1(p) + \delta_1/|\delta_1| \cdot p')$ with real input p' so that (δ_2, w_2, Y_2) is the stochastic derivative of this program for input $p' = 0$, and let $A_2(\varepsilon)$ denotes the event (B.3) for the program at $p' = 0$. Further, let

$$\widetilde{A}_2(\varepsilon) = A_2\left(\varepsilon \cdot |X_1'(p)|\right). \tag{B.8}$$

The events $A_1(\varepsilon)$ and $\widetilde{A}_2(\varepsilon)$ may be used to form a partition of the sample space:

$$1 = \mathbf{1}_{A_1^c(\varepsilon) \cap \widetilde{A}_2^c(\varepsilon)} + \mathbf{1}_{A_1(\varepsilon) \cap \widetilde{A}_2^c(\varepsilon)} + \mathbf{1}_{A_1^c(\varepsilon) \cap \widetilde{A}_2(\varepsilon)} + \mathbf{1}_{A_1(\varepsilon) \cap \widetilde{A}_2(\varepsilon)}. \tag{B.9}$$

Informally, the first event corresponds to the case of no large perturbation in either variable, the second and third correspond to finite perturbations in X_1 and X_2 respectively, and the fourth corresponds to a finite perturbation in both. As shorthands, define $X = X_2 \circ X_1$ and $\bar{X} = [X_1; X]$. Our approach will be to characterize the sensitivity of $X(p)$ under the first event using the reparameterizaton trick, under the second and third events using the argument of Proposition B.3, and finally show that the fourth event may be neglected.

Given the event $A_1^c(\varepsilon) \cap \widetilde{A}_2^c(\varepsilon)$, we may bound $|\mathrm{d}X(\varepsilon) - X'(p)|$ by first bounding $|\mathrm{d}X_1(\varepsilon)|$ as $\mathcal{O}(\varepsilon)$,

$$\left|\mathrm{d}X_1\left(\frac{\varepsilon\,|X_1'(p)|}{1 + |X_1'(p)|}\right)\right| \mathbf{1}_{A_1^c(\varepsilon)} \leq \varepsilon \cdot |X_1'(p)| \tag{B.10}$$

and then bounding $|\mathrm{d}X(\varepsilon)|$ for the composition $X = X_2 \circ X_1$,

$$\left|\mathrm{d}X\left(\frac{\varepsilon\,|X_1'(p)|}{1+|X_1'(p)|}\right)\right| \mathbf{1}_{A_1^c(\varepsilon)\cap A_2^c(\varepsilon)} = \left|\mathrm{d}X_2\left(\mathrm{d}X_1\left(\frac{\varepsilon\,|X_1'(p)|}{1+|X_1'(p)|}\right)\right)\right| \mathbf{1}_{A_1^c(\varepsilon)\cap A_2^c(\varepsilon)} \tag{B.11}$$

$$\leq \varepsilon\,|X_1'(p)|\,. \tag{B.12}$$

In addition to this bound, we have that the event $A_1^c(\varepsilon) \cap \widetilde{A}_2^c(\varepsilon)$ holds almost surely as $\varepsilon \to 0^{+/-}$, since X_1 and X_2 are almost surely differentiable. Then, an analogous bounded convergence argument as in Proposition B.1 applies, with the only difference being that the bound Eq. (B.12) has accrued a prefactor in front of ε, that does not affect the asymptotic argument. So,

$$\lim_{\varepsilon\to 0} \mathbb{E}\left[\frac{f(\bar{X}(p+\varepsilon)) - f(\bar{X}(p))}{\varepsilon}\mathbf{1}_{A_1^c(\varepsilon)\cap\widetilde{A}_2^c(\varepsilon)}\right] = \mathbb{E}\left[f'(\bar{X}(p))\cdot[X_1'(p); X_2'(X_1(p))X_1'(p)]\right]$$

$$= \mathbb{E}\left[f'(\bar{X}(p))\cdot[\delta_1; |\delta_1|\delta_2]\right]. \tag{B.13}$$

for all smooth functions $f\colon E \to \mathbb{R}$ with bounded derivative. Now, for the events $A_1(\varepsilon) \cap \widetilde{A}_2^c(\varepsilon)$ and $A_1^c(\varepsilon) \cap \widetilde{A}_2(\varepsilon)$, we have

$$\lim_{\varepsilon\to 0^{+/-}} \frac{\mathbb{P}\left(A_1(\varepsilon)\cap\widetilde{A}_2^c(\varepsilon)\;\middle|\; X_1(p), X(p)\right)}{\varepsilon} = w_1 \tag{B.14}$$

and

$$\lim_{\varepsilon\to 0^{+/-}} \frac{\mathbb{P}\left(A_1^c(\varepsilon)\cap\widetilde{A}_2(\varepsilon)\;\middle|\; X_1(p), X(p)\right)}{\varepsilon} = w_2\cdot|X_1'(p)| \tag{B.15}$$

by the almost sure convergence of $A_1^c(\varepsilon)$ and $\widetilde{A}_2^c(\varepsilon)$ to 1. Furthermore, by the definitions of Y_1 and Y_2, and again using the almost sure convergence of $A_1^c(\varepsilon)$ and $\widetilde{A}_2^c(\varepsilon)$ to 1, we have that the stacked program $[X_1; X]$ converges in distribution to $[Y_1; X_2(Y_1)]$ conditional on $A_1(\varepsilon) \cap \widetilde{A}_2^c(\varepsilon)$, $X_1(p)$, and $X(p)$, and converges in distribution to $[X_1; Y_2]$ conditional on $A_1^c(\varepsilon) \cap \widetilde{A}_2(\varepsilon)$, $X_1(p)$, and $X(p)$. Therefore, by similiar reasoning to Proposition B.3 we have

$$\lim_{\varepsilon\to 0^{+/-}} \mathbb{E}\left[\frac{f(\bar{X}(p+\varepsilon)) - f(\bar{X}(p))}{\varepsilon}\mathbf{1}_{A_1(\varepsilon)\cap\widetilde{A}_2^c(\varepsilon)}\right] = \mathbb{E}\left[w_1\left(f([Y_1; X_2(Y_1)]) - f(\bar{X}(p))\right)\right]$$

and

$$\lim_{\varepsilon\to 0^{+/-}} \mathbb{E}\left[\frac{f(\bar{X}(p+\varepsilon)) - f(\bar{X}(p))}{\varepsilon}\mathbf{1}_{A_1^c(\varepsilon)\cap\widetilde{A}_2(\varepsilon)}\right] = \mathbb{E}\left[w_2|X_1'(p)|\left(f([X_1; Y_2]) - f(\bar{X}(p))\right)\right],$$

Finally, note that,

$$\lim_{\varepsilon\to 0^{+/-}} \frac{\mathbb{P}(A_1(\varepsilon)\cap\widetilde{A}_2(\varepsilon))}{\varepsilon} = \lim_{\varepsilon\to 0^{+/-}} \frac{\mathbb{P}(A_1(\varepsilon))}{\varepsilon}\cdot\mathbb{P}(\widetilde{A}_2(\varepsilon)|A_1(\varepsilon)) = 0, \tag{B.16}$$

so by boundedness of f,

$$\lim_{\varepsilon\to 0^{+/-}} \mathbb{E}\left[\frac{f(\bar{X}(p+\varepsilon)) - f(\bar{X}(p))}{\varepsilon}\mathbf{1}_{A_1^c(\varepsilon)\cap\widetilde{A}_2(\varepsilon)}\right] = 0.$$

All together, we have

$$\begin{aligned}
&\lim_{\varepsilon\to 0^{+/-}} \mathbb{E}\left[\frac{f(\bar{X}(p+\varepsilon)) - f(\bar{X}(p))}{\varepsilon}\right] \\
&= \lim_{\varepsilon\to 0^{+/-}} \mathbb{E}\left[\frac{f(\bar{X}(p+\varepsilon)) - f(\bar{X}(p))}{\varepsilon}\left(\mathbf{1}_{A_1^c(\varepsilon)\cap\widetilde{A}_2^c(\varepsilon)} + \mathbf{1}_{A_1(\varepsilon)\cap\widetilde{A}_2^c(\varepsilon)} + \mathbf{1}_{A_1^c(\varepsilon)\cap\widetilde{A}_2(\varepsilon)} + \mathbf{1}_{A_1(\varepsilon)\cap\widetilde{A}_2(\varepsilon)}\right)\right] \\
&= \mathbb{E}\left[f'(\bar{X}(p))\cdot[\delta_1; |\delta_1|\delta_2]\right] + \mathbb{E}\left[w_1\left(f([Y_1; X_2(Y_1)]) - f(\bar{X}(p))\right)\right] \\
&\quad + \mathbb{E}\left[w_2|X_1'(p)|\left(f([X_1; Y_2]) - f(\bar{X}(p))\right)\right] + 0 \\
&= \mathbb{E}\left[f'(\bar{X}(p))\cdot[\delta_1; |\delta_1|\delta_2] + w\left(f(Y) - f(\bar{X}(p))\right)\right],
\end{aligned}$$

so (δ, w, Y) is a stochastic derivative of $\bar{X} = [X_1; X_2 \circ X_1]$ by Definition 2.1.

□

B.4 Unbiasedness of pruning strategy

In the main text, we note that we can employ a pruning strategy so that we only ever track one alternative path (i.e. one sample from the stochastic derivative component Y) and yet still obtain an unbiased estimate. The following shows that the pruning method, whereby one chooses between two samples of Y by picking with probability proportional to its weight, is indeed unbiased.

We prove by induction that the currently tracked alternative path is an unbiased choice amongst all possible alternative paths seen so far. The base case, where the first alternative path observed is followed, is trivial as it is the only choice. Now, suppose that n alternative branches have been observed so far, where $w = w_1 + w_2 + \cdots + w_n$ is the summed weight so far. Suppose we observe an $(n+1)$th branch, with weight w_{n+1}. By our pruning strategy it is chosen with probability

$$\frac{w_{n+1}}{w_{n+1} + w} = \frac{w_{n+1}}{w_1 + w_2 + \cdots + w_{n+1}}, \tag{B.17}$$

while the jth path for some $j \leq n$ will have been chosen after this step with probability

$$\frac{w_j}{w_1 + w_2 + \cdots + w_n} \cdot \frac{w_1 + w_2 + \cdots + w_n}{w_1 + w_2 + \cdots + w_{n+1}} = \frac{w_j}{w_1 + w_2 + \cdots + w_{n+1}}, \tag{B.18}$$

as desired for unbiasedness. Note that this enables us to prune *online*, i.e. without knowing the full structure of the computation a priori, which we exploit in `StochasticAD.jl`.

B.5 Smoothing

Definition B.4 (Smoothed stochastic derivative)**.** For a stochastic process $X(p)$ with right (left) stochastic derivative (δ, w, Y) at input p, the random variable $\widetilde{\delta}$ is a right (left) smoothed stochastic derivative of $X(p)$ at input p if

$$\mathbb{E}\left[\widetilde{\delta} \;\middle|\; X(p)\right] = \mathbb{E}[\delta + w(Y - X(p)) | X(p)]. \tag{B.19}$$

Using Definition 2.1, we can write the above in an alternative form that does not rely on the definition of the stochastic derivative,

$$\begin{aligned} \mathbb{E}\left[\widetilde{\delta} \;\middle|\; X(p)\right] &= \frac{\partial}{\partial y}\left[\mathbb{E}\left[X(y) - X(p) \mid X(p)\right]\right]_{y=X(p)} \\ &= \lim_{\varepsilon \to 0} \frac{\mathbb{E}[\mathrm{d}X(\varepsilon) | X(p)]}{\varepsilon}. \end{aligned} \tag{B.20}$$

Smoothed stochastic derivatives propagate through functions that are *locally* linear over the range of Y conditionally on $X(p)$, as we now formalize. Note that this is a much weaker requirement than global linearity over the full range of $X(p)$, as the range of Y conditional on $X(p)$ is generally more restricted (e.g. $Y \in \{X(p) - 1, X(p) + 1\}$ for a binomial variable).

Proposition B.5. *Suppose that f is linear over the range of Y conditionally on $X(p) = x$, for all x. Then,*

$$\frac{\partial}{\partial p}\mathbb{E} f(X(p)) = \mathbb{E}\left[f'(X(p))\widetilde{\delta}\right]. \tag{B.21}$$

Proof. For such an f, note that Eq. (2.4) of Definition 2.1 simplifies to

$$\frac{\partial}{\partial p}\mathbb{E}\left[f(X(p))\right] = \mathbb{E}\left[f'(X(p)) \cdot \delta + w \cdot f'(X(p))(Y - X(p))\right] \tag{B.22}$$

$$= \mathbb{E}\left[f'(X(p))\left(\delta + w \cdot (Y - X(p))\right)\right] \tag{B.23}$$

$$= \mathbb{E}\left[f'(X(p))\widetilde{\delta}\right]. \tag{B.24}$$

□

In most cases, local linearity will only hold approximately, leading to bias in the estimate produced by propagating smoothed stochastic derivatives. However, in the case of a particle filter resampling step an exact estimate is produced, as we show in the following.

B.6 Sequential Monte-Carlo

Let $X(p)$ be a stochastic program approximated by the program $\widetilde{X}(p)$. Assume we can sample from $\widetilde{X}(p)$. Assuming absolute continuity, we may write expectations as $\mathbb{E}\varpi(p)f(\widetilde{X}(p)) = \mathbb{E}f(X(p))$ where $\varpi(p)$ is the Radon-Nikodym derivative of the law of $\widetilde{X}(p)$ with respect to the law of $X(p)$, evaluated in $\widetilde{X}(p)$. We thus consider the program which returns the pair of weight and value

$$(\varpi(p), \widetilde{X}(p)).$$

Proposition B.6. *If $\widetilde{X}(p)$ is a continuous program, reparameterized such that it is differentiable pointwise, and $\varpi(p)$ has derivative (δ, W, Y)*

$$\frac{\partial}{\partial p}\mathbb{E}g(X(p)) = \mathbb{E}\left[\varpi(p)g'(\widetilde{X}(p))\widetilde{X}'(p) + (\varpi'(p)(\delta + W \cdot \mathbb{E}[Y - \varpi(p) \mid \varpi(p)])g(\widetilde{X}(p))\right].$$

Proof. As the function $(\varpi, x) \mapsto \varpi g(x)$ is bilinear the statement follows from Proposition B.5. □

Programs returning weighted samples can be monadically composed: If $\mathbb{E}\varpi_i(x_{i-1})\widetilde{X}_i(x_{i-1}) = \mathbb{E}X_i(x_{i-1})$, $i = 1, 2$, then

$$\mathbb{E}[X_2(X_1(p))] = \mathbb{E}\left[\varpi_2(\widetilde{X}_1(p))\varpi_1(p)\widetilde{X}_2(\widetilde{X}_1(p))\right]$$

by conditional independence.

Note that also the product $\varpi_2(\widetilde{X}_1(p))\varpi_1(p)$ is linear in each of its arguments $\varpi_2(p)$ and $\varpi_1(p)$.

C Formalism of the particle filter

Using a bootstrap particle sampler, we can approximate the posterior distributions of the states given noisy and partial observations of the state of a hidden Markov model (Section 3.4) by a cloud of K *weighted particles* which we denote by $x_i^{(k)}$. The particles evolve independently according to the program, $x_i^{(k)} = X_i(x_{i-1}^{(k)}, \theta)$. Each particle has a corresponding likelihood weight $\varpi_i^{(k)}$ measuring how well the trajectory of the particle so far is matching the observations, updated by $\varpi_i^{(k)} = p(y_i \mid x_i^{(k)}, \theta)\varpi_{i-1}^{(k)}$, such that the empirical measure $\sum_k \varpi_i^{(k)}\delta_{x_i^{(k)}} / \sum_k \varpi_i^{(k)}$ of the weighted particles approximates the filtering distribution $p(x_i \mid y_1, \ldots, y_i, \theta)$. Most importantly for our application, the marginal likelihood of a parameter θ can be read off the weights at the last step, $p(y_1, \ldots, y_n \mid \theta) \approx \sum_{k=1}^{K} \varpi_n^{(k)}$.

So far, the particles and weight trajectories are differentiable with respect to the parameter [3]. However, weight degeneracy, the collapse of all but a few weights, is a common problem. The strategy to circumvent this issue is to include multinomial resampling steps, which are inherently discrete, but differentiable with (smoothed) stochastic duals: The population of particles with a population of particles of *equal* weight $(x_i'^{(k)}, \varpi_i')_{k \in \{1,\ldots,K\}}$, i.e. where the new particles are chosen by sampling indices from the multinomial distribution with the normalized weight vector $w = (w^{(1)}, \ldots, w^{(K)}) = (\varpi_i^{(1)}/\varpi, \ldots, \varpi_i^{(K)}/\varpi)$. After this step,

$$\sum_k \varpi_i^{(k)} f(x_i^{(k)}) \approx \sum_k \varpi_i'^{(k)} f(x_i'^{(k)})$$

and $\varpi := \sum_k \varpi_i'^{(k)} = \sum_k \varpi_i^{(k)}$. Resampling is numerically attractive because unlikely particles are discarded so numerical resources are not wasted on particles with vanishing weight. One achieves a similar effect by retaining each particle independently with probability proportional to its weight.

In first approximation, a selected particle whose weight $\varpi^{(k)}$ decreases by $h^{(k)}$ has a chance of $\frac{h^{(k)}(1-w^{(k)})}{\varpi^{(k)}}$ not being selected under the new weight. We can disregard the effect $1 - w^{(k)}$ on the other selected particles without incurring a bias, as shown below.

We thus propose to set the new weight of particle j spawned from a particle k to $\frac{\varpi}{K} + \frac{h^{(k)}}{K w^{(k)}}$. This is readily accomplished using a formally differentiable weight function

$$\varpi^{(j)} = \varpi \, \mathrm{newweight}(w^{(k)})/K$$

with definition:

```
# primal
new_weight(p::Real) = 1

# forward-mode AD rule
function new_weight(p::ForwardDiff.Dual{T}) where {T}
    ∂_p = ForwardDiff.partials(p)
    value_p = ForwardDiff.value(p)
    value_p = max(1e-5, value_p)
    ForwardDiff.Dual{T}(1, ∂_p / value_p)
end

# reverse-mode AD rule
function ChainRulesCore.rrule(::typeof(new_weight), p)
    function new_weight_pullback(∇Ω)
        return (ChainRulesCore.NoTangent(), ∇Ω / p)
    end
    return (one(p), new_weight_pullback)
end
```

We now have:

Proposition C.1 (Resampling)**.** *Consider a vector of weights $\varpi^{(1)}, \ldots, \varpi^{(K)}$ with $\varpi = \sum_{k=1}^{K} \varpi^{(k)}$ and a vector of particles $x^{(1)}, \ldots, x^{(K)}$ as representation of $\mu = \sum_{k=1}^{K} w^{(k)} f(\varpi^{(k)})$ where $w^{(j)} = \dfrac{\varpi^{(j)}}{\varpi}$ are normalized weights and f a functional of interest. If J is a categorical random variable with $\mathbb{P}(J = j) = w^{(j)}$, then*

$$\varpi \mathbb{E} \, f(x_J) = \mu$$

and for infinitesimal differentials $h^{(1)}, \ldots, h^{(K)}$, $\bar{h}^{(1)}, \ldots, \bar{h}^{(K)}$ for weights and particles,

$$\mathbb{E}\left[\left(\varpi + \frac{h^{(J)}}{\varpi^{(J)}}\right) f(x^{(J)} + \bar{h}^{(J)})\right] = \sum_{k=1}^{K} (\varpi^{(k)} + h^{(k)}) f(x^{(k)} + \bar{h}^{(k)}).$$

This is as required, giving the unbiasedness of our rule for the resampling step of the particle filter.

D Implementation details

D.1 Experiment details

All computation times in Fig. 5 were measured on an Intel Xeon Platinum 8260 CPU and Julia version 1.6. Garbage collection times are included in the total run time. We provide code and instructions to run the examples in the tutorials folder of `StochasticAD.jl`.

D.2 Software dependencies

`StochasticAD.jl` is implemented in the Julia Language [4] and uses internally `Distributions.jl` and `DistributionsAD.jl` [2, 1], `ChainRulesCore.jl` [5], and `ForwardDiff.jl` [6].

The presented examples are using the additional packages: `BenchmarkTools.jl` [7], `Zygote.jl` [8], `GaussianDistributions.jl` [9], and `Plots.jl` [10].

We refer the reader to the documentation of `StochasticAD.jl` for more detailed information on the package implementation.

Supplemental References

[SI1] Dahua Lin, John Myles White, Simon Byrne, Douglas Bates, Andreas Noack, John Pearson, Alex Arslan, Kevin Squire, David Anthoff, Theodore Papamarkou, Mathieu Besançon, Jan Drugowitsch, Moritz Schauer, and contributors. JuliaStats/Distributions.jl: a Julia package for probability distributions and associated functions, July 2019.

[SI2] Mathieu Besançon, Theodore Papamarkou, David Anthoff, Alex Arslan, Simon Byrne, Dahua Lin, and John Pearson. Distributions.jl: Definition and modeling of probability distributions in the juliastats ecosystem. *Journal of Statistical Software*, 98(16):1–30, 2021.

[SI3] Rico Jonschkowski, Divyam Rastogi, and Oliver Brock. Differentiable particle filters: End-to-end learning with algorithmic priors. *arXiv:1805.11122*, 2018.

[SI4] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):65–98, 2017.

[SI5] Frames Catherine White, Michael Abbott, Miha Zgubic, Jarrett Revels, Alex Arslan, Seth Axen, Simeon Schaub, Nick Robinson, Yingbo Ma, Gaurav Dhingra, Will Tebbutt, Niklas Heim, David Widmann, Andrew David Werner Rosemberg, Niklas Schmitz, Christopher Rackauckas, Rainer Heintzmann, Frank Schäfer, Carlo Lucibello, Keno Fischer, Alex Robson, Jerry Ling, Matt Brzezinski, Andrei Zhabinski, Daniel Wennberg, Mathieu Besançon, Pietro Vertechi, Shashi Gowda, and Andrew Fitzgibbon. Juliadiff/chainrules.jl: v1.35.0, May 2022.

[SI6] J. Revels, M. Lubin, and T. Papamarkou. Forward-mode automatic differentiation in Julia. *arXiv:1607.07892 [cs.MS]*, 2016.

[SI7] Jiahao Chen and Jarrett Revels. Robust benchmarking in noisy environments, Aug 2016.

[SI8] Michael Innes. Don't unroll adjoint: Differentiating SSA-form programs, 2018.

[SI9] Moritz Schauer and contributors. GaussianDistributions.jl, 2018.

[SI10] Simon Christ, Daniel Schwabeneder, and Christopher Rackauckas. Plots.jl–a user extendable plotting api for the julia programming language. *arXiv preprint arXiv:2204.08775*, 2022.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 3 and Section 4 for discussions on potentially large variance of the estimator and language-wide AD support. In addition, we provide a continuously updated list about the current limitations of the `StochasticAD.jl` package in its README.
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Appendix.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix.
3. If you ran experiments...

(a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See Appendix D and the README of `StochasticAD.jl`.

(b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] For the differentiable particle sampler, we have included the error on the mean value of the run times as error bars.

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix D.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes] We cite the Julia packages on which this work is based.

(b) Did you mention the license of the assets? [No] We only use open-source software packages.

(c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We have included our package in the supplemental material.

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] We do not use any previously collected data.

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]