# Efficient Optimization of Loops and Limits with Randomized Telescoping Sums

**Alex Beatson** [1] **Ryan P. Adams** [1]

## Abstract

We consider optimization problems in which the objective requires an inner loop with many steps or is the limit of a sequence of increasingly costly approximations. Meta-learning, training recurrent neural networks, and optimization of the solutions to differential equations are all examples of optimization problems with this character. In such problems, it can be expensive to compute the objective function value and its gradient, but truncating the loop or using less accurate approximations can induce biases that damage the overall solution. We propose *randomized telescope* (RT) gradient estimators, which represent the objective as the sum of a telescoping series and sample linear combinations of terms to provide cheap unbiased gradient estimates. We identify conditions under which RT estimators achieve optimization convergence rates independent of the length of the loop or the required accuracy of the approximation. We also derive a method for tuning RT estimators online to maximize a lower bound on the expected decrease in loss per unit of computation. We evaluate our adaptive RT estimators on a range of applications including meta-optimization of learning rates, variational inference of ODE parameters, and training an LSTM to model long sequences.

## 1. Introduction

Many important optimization problems consist of objective functions that can only be computed iteratively or as the limit of an approximation. Machine learning and scientific computing provide many important examples. In meta-learning, evaluation of the objective typically requires the training of a model, a case of bi-level optimization. When training a model on sequential data or to make decisions over time, each learning step requires looping over time steps. More broadly, in many scientific and engineering applications one wishes to optimize an objective that is defined as the limit of a sequence of approximations with both fidelity and computational cost increasing according to a natural number $n \geq 1$. Inner-loop examples include: integration by Monte Carlo or quadrature with $n$ evaluation points; solving ordinary differential equations (ODEs) with an Euler or Runge Kutta method with $n$ steps and $\mathcal{O}(\frac{1}{n})$ step size; and solving partial differential equations (PDEs) with a finite element basis with size or order increasing with $n$.

Whether the task is fitting parameters to data, identifying the parameters of a natural system, or optimizing the design of a mechanical part, in this work we seek to more rapidly solve problems in which the objective function demands a tradeoff between computational cost and accuracy. We formalize this by considering parameters $\theta \in \mathbb{R}^D$ and a loss function $\mathcal{L}(\theta)$ that is the uniform limit of a sequence $\mathcal{L}_n(\theta)$:

$$\min_\theta \mathcal{L}(\theta) = \min_\theta \lim_{n \to H} \mathcal{L}_n(\theta)\,. \tag{1}$$

Some problems may involve a finite horizon $H$, in other cases $H = \infty$. We also introduce a cost function $C : \mathbb{N}_+ \to \mathbb{R}$ that is nondecreasing in $n$ to represent the cost of computing $\mathcal{L}_n$ and its gradient.

A principal challenge of optimization problems with the form in Eq. 1 is selecting a finite $N$ such that the minimum of the surrogate $\mathcal{L}_N$ is close to that of $\mathcal{L}$, but without $\mathcal{L}_N$ (or its gradients) being too expensive. Choosing a large $N$ can be computationally prohibitive, while choosing a small $N$ may bias optimization. Meta-optimizing learning rates with truncated horizons can choose wrong hyperparameters by orders of magnitude (Wu et al., 2018). Truncating backpropogation through time for recurrent neural networks (RNNs) favors short term dependencies (Tallec & Ollivier, 2017). Using too coarse a discretization to solve an ODE or PDE can cause error in the solution and bias outer-loop optimization. These optimization problems thus experience a sharp trade-off between efficient computation and bias.

We propose *randomized telescope* (RT) gradient estimators, which provide cheap unbiased gradient estimates to allow efficient optimization of these objectives. RT estimators represent the objective or its gradients as a telescoping series of differences between intermediate values, and draw

[1]Department of Computer Science, Princeton University, Princeton, NJ, USA. Correspondence to: Alex Beatson <abeatson@cs.princeton.edu>.

*Proceedings of the $36^{th}$ International Conference on Machine Learning*, Long Beach, California, PMLR 97, 2019. 

weighted samples from this series to maintain unbiasedness while balancing variance and expected computation.

The paper proceeds as follows. Section 2 introduces RT estimators and their history. Section 3 formalizes RT estimators for optimization, and discusses related work in optimization. Section 4 discusses conditions for finite variance and computation, and proves RT estimators can achieve optimization guarantees for loops and limits. Section 5 discusses designing RT estimators by maximizing a bound on expected improvement per unit of computation. Section 6 describes practical considerations for adapting RT estimators online. Section 7 presents experimental results. Section 8 discusses limitations and future work. Appendix A presents algorithm pseudocode. Appendix B presents proofs. Code may be found at `https://github.com/PrincetonLIPS/randomized_telescopes`.

## 2. Unbiased randomized truncation

In this section, we discuss the general problem of estimating limits through randomized truncation. The first subsection presents the randomized telescope family of unbiased estimators, while the second subsection describes their history (dating back to von Neumann and Ulam). In the following sections, we will describe how this technique can be used to provide cheap unbiased gradient estimates and accelerate optimization for many problems.

### 2.1. Randomized telescope estimators

Consider estimating any quantity $Y_H := \lim_{n\to H} Y_n$ for $n \in \mathbb{N}_+$ where $H \in \mathbb{N}_+ \cup \{\infty\}$. Assume that we can compute $Y_n$ for any finite $n \in \mathbb{N}_+$, but since the cost is nondecreasing in $n$ there is a point at which this becomes impractical. Rather than truncating at a fixed value short of the limit, we may find it useful to construct an unbiased estimator of $Y_H$ and take on some randomness in return for reduced computational cost.

Define the backward difference $\Delta_n$ and represent the quantity of interest $Y_H$ with a telescoping series:

$$Y_H = \sum_{n=1}^{H} \Delta_n \quad \text{where} \quad \Delta_n = \begin{cases} Y_n - Y_{n-1} & n > 1 \\ Y_1 & n = 1 \end{cases}.$$

We may sample from this telescoping series to provide unbiased estimates of $Y_H$, introducing variance to our estimator in exchange for reducing expected computation. We use the name *randomized telescope* (RT) to refer to the family of estimators indexed by a distribution $q$ over the integers $1, \ldots, H$ (for example, a geometric distribution) and a weight function $W(n, N)$:

$$\hat{Y}_H = \sum_{n=1}^{N} \Delta_n W(n, N) \qquad N \in \{1, \ldots, H\} \sim q\,. \quad (2)$$

**Proposition 2.1.** ***Unbiasedness of RT estimators.*** *The RT estimators in (2) are unbiased estimators of $Y_H$ as long as*

$$\mathbb{E}_{N\sim q}[W(n, N)\mathbb{1}\{N \geq n\}] = \sum_{N=n}^{H} W(n, N)q(N) = 1,\ \forall n\,.$$

See Appendix B for a short proof. Although we are coining the term "randomized telescope" to refer to the family of estimators with the form of Eq. 2, the underlying trick has a long history, discussed in the next section. The literature we are aware of focusses on one or both of two special cases of Eq. 2, defined by choice of weight function $W(n, N)$. We will also focus on these two variants of RT estimators, but we observe that there is a larger family.

Most related work uses the "Russian roulette" estimator originally discovered and named by von Neumann and Ulam (Kahn, 1955), which we term ***RT-RR*** and has the form

$$W(n, N) = \frac{1}{1 - \sum_{n'=1}^{n-1} q(n')} \mathbb{1}\{N \geq n\}\,. \quad (3)$$

It can be seen as summing the iterates $\Delta_n$ while flipping a biased coin at each iterate. With probability $q(n)$, the series is truncated at term $N = n$. With probability $1 - q(n)$, the process continues, and all future terms are upweighted by $\frac{1}{1-q(n)}$ to maintain unbiasedness.

The other important special case of Eq. 2 is the "single sample" estimator ***RT-SS***, referred to as "single term weighted truncation" in Lyne et al. (2015). RT-SS takes

$$W(n, N) = \frac{1}{q(N)} \mathbb{1}\{n = N\}\,. \quad (4)$$

This is directly importance sampling the differences $\Delta_n$.

We will later prove conditions under which RT-SS and RT-RR should be preferred. Of all estimators in the form of Eq. 2 which obey proposition 2.1 and for all $q$, RT-SS minimizes the variance across worst-case diagonal covariances $\mathrm{Cov}(\Delta_i, \Delta_j)$. Within the same family, RT-RR achieves minimum variance when $\Delta_i$ and $\Delta_j$ are independent for all $i, j$.

### 2.2. A brief history of unbiased randomized truncation

The essential trick—unbiased estimation of a quantity via randomized truncation of a series—dates back to unpublished work from John von Neumann and Stanislaw Ulam. They are credited for using it to develop a Monte Carlo method for matrix inversion in Forsythe & Leibler (1950), and for a method for particle diffusion in Kahn (1955).

It has been applied and rediscovered in a number of fields and applications. The early work from von Neumann and Ulam led to its use in computational physics, in neutron

transport problems (Spanier & Gelbard, 1969), for studying lattice fermions (Kuti, 1982), and to estimate functional integrals (Wagner, 1987). In computer graphics Arvo & Kirk (1990) introduced its use for ray tracing; it is now widely used in rendering software. In statistical estimation, it has been used for estimation of derivatives (Rychlik, 1990), unbiased kernel density estimation (Rychlik, 1995), doubly-intractable Bayesian posterior distributions (Girolami et al., 2013; Lyne et al., 2015; Wei & Murray, 2016), and unbiased Markov chain Monte Carlo (Jacob et al., 2017).

The underlying trick has been rediscovered by Fearnhead et al. (2008) for unbiased estimation in particle filtering, by McLeish (2010) for debiasing Monte Carlo estimates, by Rhee & Glynn (2012; 2015) for unbiased estimation in stochastic differential equations, and by Tallec & Ollivier (2017) to debias truncated backpropagation. The latter also uses RT estimators for optimization; however, it only considers fixed "Russian roulette"-style randomized telescope estimators and does not consider convergence rates or how to adapt the estimator online (our main contributions).

## 3. Optimizing loops and limits

In this paper, we consider optimizing functions defined as limits. Consider a problem where, given parameters $\theta$ we can obtain a series of approximate losses $\mathcal{L}_n(\theta)$, which converges uniformly to some limit $\lim_{n\to H} \mathcal{L}_n := \mathcal{L}$, for $n \in \mathbb{N}_+$ and $H \in \mathbb{N}_+ \cup \{\infty\}$. We assume the sequence of gradients with respect to $\theta$, denoted $G_n(\theta) := \nabla_\theta \mathcal{L}_n(\theta)$ converge uniformly to a limit $G(\theta)$. Under this uniform convergence and assuming convergence of $\mathcal{L}_n$, we have $\lim_{n\to H} \nabla_\theta \mathcal{L}_n(\theta) = \nabla_\theta \lim_{n\to H} \mathcal{L}_n(\theta)$ (see Theorem 7.17 in Rudin et al. (1976)), and so $G(\theta)$ is indeed the gradient of our objective $\mathcal{L}(\theta)$. We assume there is a computational cost $C(n)$ associated with evaluating $\mathcal{L}_n$ or $G_n$, nondecreasing with $n$, and we wish to efficiently minimize $\mathcal{L}$ with respect to $\theta$. Loops are an important special case of this framework, where $\mathcal{L}_n$ is the final output resulting from running e.g., a training loop or RNN for some number of steps increasing in $n$.

### 3.1. Randomized telescopes for optimization

We propose using randomized telescopes as a stochastic gradient estimator for such optimization problems. We aim to accelerate optimization much as mini-batch stochastic gradient descent accelerates optimization for large datasets: using Monte Carlo sampling to decrease the expected cost of each optimization step, at the price of increasing variance in the gradient estimates, without introducing bias.

Consider the gradient $G(\theta) = \lim_{n\to H} G_n(\theta)$, and the backward difference $\Delta_n(\theta) = G_n(\theta) - G_{n-1}(\theta)$, where $G_0(\theta) = 0$, so that $G(\theta) = \sum_{n=1}^{H} \Delta_n(\theta)$. We use the randomized telescope estimator

$$\hat{G}(\theta) = \sum_{n=1}^{N} \Delta_n(\theta) W(n, N) \tag{5}$$

where $N \in \{1, 2, \ldots, H\}$ is drawn according to a proposal distribution $q$, and together $W$ and $q$ satisfy proposition 2.1.

Note that due to linearity of differentiation, and letting $\mathcal{L}_0(\theta) := 0$, we have

$$\sum_{n=1}^{N} \Delta_n(\theta) W(n, N) = \nabla_\theta \sum_{n=1}^{N} (\mathcal{L}_n(\theta) - \mathcal{L}_{n-1}(\theta)) W(n, N)\,.$$

Thus, when the computation of $\mathcal{L}_n(\theta)$ can reuse most of the computation performed for $\mathcal{L}_{n-1}(\theta)$, we can evaluate $\hat{G}_N(\theta)$ via forward or backward automatic differentiation with cost approximately equal to computing $G_N(\theta)$, i.e., $\hat{G}_N(\theta)$ has computation cost $\approx C(N)$. This most often occurs when evaluating $\mathcal{L}_n(\theta)$ involves an inner loop with a step size which does not change with $n$, e.g., meta-learning and training RNNs, but not solving ODEs or PDEs. When computing $\mathcal{L}_n(\theta)$ does not reuse computation evaluating $\hat{G}_N(\theta)$ has computation cost $\sum_{n=1}^{N} C(n) \mathbb{1}\{W(n, N) \neq 0\}$.

### 3.2. Related work in optimization

Gradient-based bilevel optimization has seen extensive work in literature. See Jameson (1988) for an early example of optimizing implicit functions, Christianson (1998) for a mathematical treatment, and Maclaurin et al. (2015); Franceschi et al. (2017) for recent treatments in machine learning. Shaban et al. (2018) propose truncating only the backward pass by only backpropagating through the final few optimization steps to reduce memory requirements. Metz et al. (2018) propose linearly increasing the number of inner steps over the course of the outer optimization.

An important case of bi-level optimization is optimization of architectures and hyperparameters. Truncation causes bias, as shown by Wu et al. (2018) for learning rates and by Metz et al. (2018) for neural optimizers.

Bi-level optimization is also used for meta-learning across related tasks (Schmidhuber, 1987; Bengio et al., 1992). Ravi & Larochelle (2016) train an initialization and optimizer, and Finn et al. (2017) only an initialization, to minimize validation loss. The latter paper shows increasing performance with the number of steps used in the inner optimization. However, in practice the number of inner loop steps must be kept small to allow training over many tasks.

Bi-level optimization can be accelerated by amortization. Variational inference can be seen as bi-level optimization; variational autoencoders (Kingma & Welling, 2014) amortize the inner optimization with a predictive model of the

solution to the inner objective. Recent work such as Brock et al. (2018); Lorraine & Duvenaud (2018) amortizes hyperparameter optimization in a similar fashion.

However, amortizing the inner loop induces bias. Cremer et al. (2018) demonstrate this in VAEs, while Kim et al. (2018) show that in VAEs, combining amortization with truncation by taking several gradient steps on the output of the encoder can reduce this bias. This shows these techniques are orthogonal to our contributions: while fully amortizing the inner optimization causes bias, predictive models of the limit can accelerate convergence of $\mathcal{L}_n$ to $\mathcal{L}$.

Our work is also related to work on training sequence models. Tallec & Ollivier (2017) use the Russian roulette estimator to debias truncated backpropagation through time. They use a fixed geometrically decaying $q(N)$, and show that this improves validation loss for Penn Treebank. They do not consider efficiency of optimization, or methods to automatically set or adapt the hyperparameters of the randomized telescope. Trinh et al. (2018) learn long term dependencies with auxiliary losses. Other work accelerates optimization of sequence models by replacing recurrent models with models which use convolution or attention (Vaswani et al., 2017), which can be trained more efficiently.

## 4. Convergence rates with fixed RT estimators

Before considering more complex large-scale problems, we examine the simple RT estimator for stochastic gradient descent on convex problems. We assume that the sequence $\mathcal{L}_n(\theta)$ and units for $C$ are chosen such that $C(n) = n$. We study RT-SS, with $q(N)$ fixed *a priori*. We consider optimizing parameters $\theta \in \mathcal{K}$, where $\mathcal{K} \subset \mathcal{R}^d$ is a bounded, convex and compact set with diameter bounded by $D$. We assume $\mathcal{L}(\theta)$ is convex in $\theta$, and $G_n(\theta)$ converge according to $||\Delta_n||_2 \leq \psi_n$, where $\psi_n$ converges polynomially or geometrically. The quantity of interest is the instantaneous regret, $R_t = \mathcal{L}(\theta_t) - \min_\theta \mathcal{L}(\theta)$, where $\theta_t$ is the parameter after $t$ steps of SGD.

In this setting, any fixed truncation scheme using $\mathcal{L}_N$ as a surrogate for $\mathcal{L}$, with fixed $N < H$, cannot achieve $\lim_{t\to\infty} R_t = 0$. Meanwhile, the fully unrolled estimator has computational cost which scales with $H$. In the many situations where $H \to \infty$, it is impossible to take even a single gradient step with this estimator.

The randomized telescope estimator overcomes these drawbacks by exploiting the fact that $G_n$ converges according to $||\Delta_n||_2 \leq \psi_n$. As long as $q$ is chosen to have tails no lighter than $\psi_n$, for sufficiently fast convergence, the resulting RT-SS gradient estimator achieves asymptotic regret bounds invariant to $H$ in terms of convergence rate.

All proofs are deferred to Appendix B. We begin by proving bounds on the variance and expected computation for polynomially decaying $q(N)$ and $\psi_n$.

**Theorem 4.1.** ***Bounded variance and compute with polynomial convergence of*** $\psi$. *Assume $\psi$ converges according to $\psi_n \leq c/n^p$ or faster, for constants $p > 0$ and $c > 0$. Choose the RT-SS estimator with $q(N) \propto 1/(N^{p+1/2})$. The resulting estimator $\hat{G}$ achieves expected compute $C \leq (\mathcal{H}_H^{p-1/2})^2$, where $\mathcal{H}_H^i$ is the $H$th generalized harmonic number of order $i$, and expected squared norm $\mathbb{E}[||\hat{G}||_2^2] \leq c_\psi^2(\mathcal{H}_H^{p-1/2})^2 := \tilde{G}^2$. The limit $\lim_{H\to\infty} \mathcal{H}_H^{p-1/2}$ is finite iff $p > 3/2$, in which case it is given by the Riemannian zeta function, $\lim_{H\to\infty} \mathcal{H}_H^{p-1/2} = \zeta(p - 1/2)$. Accordingly, the estimator achieves horizon-agnostic variance and expected compute bounds iff $p > 3/2$.*

The corresponding bounds for geometrically decaying $q(N)$ and $\psi_n$ follow.

**Theorem 4.2.** ***Bounded variance and compute with geometric convergence of*** $\psi$. *Assume $\psi_n$ converges according to $\psi_n \leq cp^n$, or faster, for $0 < p < 1$. Choose RT-SS and with $q(N) \propto p^N$. The resulting estimator $\hat{G}$ achieves expected compute $C \leq (1-p)^{-2}$ and expected squared norm $||\hat{G}||_2^2 \leq \frac{c}{(1-p)^2} := \tilde{G}^2$. Thus, the estimator achieves horizon-agnostic variance and expected compute bounds for all $0 < p < 1$.*

Given a setting and estimator $\hat{G}$ from either 4.1 or 4.2, with corresponding expected compute cost $C$ and upper bound on expected squared norm $\tilde{G}^2$, the following theorem considers regret guarantees when using this estimator to perform stochastic gradient descent.

**Theorem 4.3.** ***Asymptotic regret bounds for optimizing infinite-horizon programs***. *Assume the setting from 4.1 or 4.2, and the corresponding $C$ and $\tilde{G}$ from those theorems. Let $R_t$ be the instantaneous regret at the $t$th step of optimization, $R_t = \mathcal{L}(\theta_t) - \min_\theta \mathcal{L}(\theta)$. Let $t(B)$ be the greatest $t$ such that a computational budget $B$ is not exceeded. Use online gradient descent with step size $\eta_t = \frac{D}{\sqrt{t}\mathbb{E}[||\hat{G}||_2^2]}$. As $B \to \infty$, the asymptotic instantaneous regret is bounded by $R_{t(B)} \leq \mathcal{O}(\tilde{G}D\sqrt{\frac{C}{B}})$, independent of $H$.*

Theorem 4.3 indicates that if $G_n$ converges sufficiently fast and $\mathcal{L}_n$ is convex, the RT estimator provably optimizes the limit.

## 5. Adaptive RT estimators

In practice, the estimator considered in the previous section may have high variance. This section develops an objective for designing such estimators, and derives closed-form $W(n, N)$ and $q$ which maximize this objective given estimates of $\mathbb{E}[||\Delta_i||_2^2]$ and assumptions on $\mathrm{Cov}(\Delta_i, \Delta_j)$.

### 5.1. Choosing between unbiased estimators

We propose choosing an estimator which achieves the best lower bound on the expected improvement per compute unit spent, given smoothness assumptions on the loss. Our analysis builds on that of Balles et al. (2016): they adaptively choose a batch size using batch covariance information, while we choose between between arbitrary unbiased gradient estimators using knowledge of those estimators' expected squared norms and computation cost.

Here we assume that the true gradient of the objective $\nabla_\theta[\mathcal{L}(\theta)] := \nabla_\theta$ (for compactness of notation) is smooth in $\theta$. We do not assume convexity. Note that $\nabla_\theta$ is not necessarily equal to $G(\theta)$, as the loss $\mathcal{L}(\theta)$ and its gradient $G(\theta)$ may be random variables due to sampling of data and/or latent variables.

We assume that $\mathcal{L}$ is $L$-smooth (the gradients of $\mathcal{L}(\theta)$ are $L$-Lipschitz), i.e., there exists a constant $L > 0$ such that $\nabla_{\theta_b} - \nabla_{\theta_a} \leq L||\theta_b - \theta_a||_2 \quad \forall \theta_a, \theta_b \in \mathbb{R}^d$. It follows (Balles et al., 2016; Bottou et al., 2018) that, when performing SGD with an unbiased stochastic gradient estimator $\hat{G}_t$,

$$\mathbb{E}[\mathcal{L}_H(\theta_t) - \mathcal{L}_H(\theta_{t+1})] \geq \mathbb{E}[\eta_t \nabla_{\theta_t}^T \hat{G}_t(\theta_t)] - \mathbb{E}[\frac{L\eta_t^2}{2}||\hat{G}_t(\theta_t)||_2^2]\,. \quad (6)$$

Unbiasedness of $\hat{G}$ implies $\mathbb{E}[\nabla_{\theta_t}^T \hat{G}_t(\theta_t)] = ||\nabla_{\theta_t}^T||_2^2$, thus:

$$\mathbb{E}[\mathcal{L}_H(\theta_t) - \mathcal{L}_H(\theta_{t+1})] \geq \mathbb{E}[\eta_t||\nabla_\theta||_2^2] - \mathbb{E}[\frac{L\eta_t^2}{2}||\hat{G}_t(\theta_t)||_2^2] := J\,. \quad (7)$$

Above, $J$ is a lower bound on the expected improvement in the loss from one optimization step. Given a fixed choice of $\hat{G}_t(\theta_t)$, how should one pick the learning rate $\eta_t$ to maximize $J$ and what is the corresponding lower bound on expected improvement?

Optimizing $\eta_t$ by finding $\eta_t^\star$ s.t. ${}^{dJ}/_{d\eta_t^\star} = 0$ yields

$$\eta_t^\star = \frac{||\nabla_\theta||_2^2}{L\mathbb{E}[||\hat{G}_t(\theta_t)||_2^2]} \propto \frac{1}{\mathbb{E}[||\hat{G}_t(\theta_t)||_2^2]} \quad (8)$$

$$J^\star = \frac{||\nabla_\theta||^4}{2L\mathbb{E}[||\hat{G}_t(\theta_t)||_2^2]} \propto \frac{1}{\mathbb{E}[||\hat{G}_t(\theta_t)||_2^2]}\,. \quad (9)$$

This tells us how to choose $\eta_t$ if we know $L$, $||\hat{G}||_2^2$, etc. In practice, it is unlikely that we know $L$ or even $||\nabla_{\theta_t}||_2$. We instead assume we have access to some "reference" learning rate $\bar{\eta}_t$, which has been optimized for use with a "reference" gradient estimator $\bar{G}_t$, with known $\mathbb{E}[||\bar{G}_t||_2^2]$. When using RT estimators, we may have access to learning rates which have been optimized for use with the un-truncated estimator. Even when we do not know an optimal reference learning rate, this construction means we need only tune one hyperparameter (the reference learning rate), and can still choose between a family of gradient estimators online. Instead of directly maximizing $J$, we choose $\eta_t$ for $\hat{G}$ by maximizing *improvement relative to the reference estimator* in terms of $J$, the lower bound on expected improvement.

Assume that $\bar{\eta}_t$ has been set optimally for a problem and reference estimator $\bar{G}$ up to some constant $k$, i.e.,

$$\bar{\eta}_t = k\frac{||\nabla_{\theta_t}||_2^2}{L\mathbb{E}[||\bar{G}_t(\theta_t)||_2^2]}\,. \quad (10)$$

Then the expected improvement $\bar{J}$ obtained by the reference estimator $\bar{G}$ is:

$$\bar{J} = (k - \frac{k^2}{2})\frac{||\nabla_{\theta_t}||^4}{2L\mathbb{E}[||\bar{G}_t(\theta_t)||_2^2]} \quad (11)$$

We assume that $0 < k < 2$, such that $\bar{J}$ is positive and the reference has guaranteed expected improvement. Now set the learning rate according to

$$\eta_t = \bar{\eta}_t \frac{\mathbb{E}[||\hat{G}_t||_2^2]}{\mathbb{E}[||\bar{G}_t||_2^2]}\,. \quad (12)$$

It follows that the expected improvement $\hat{J}$ obtained by the estimator $\hat{G}$ is

$$\hat{J} = \frac{\mathbb{E}[||\bar{G}_t(\theta_t)||_2^2]}{\mathbb{E}[||\hat{G}_t(\theta_t)||_2^2]}\bar{J} \quad (13)$$

Let the expected computation cost of evaluating $\hat{G}$ be $\hat{C}$. We want to maximize ${}^{\hat{J}}/_{\hat{C}}$. If we use the above method to choose $\eta_t$, we have ${}^{\hat{J}}/_{\hat{C}} \propto \left(\hat{C}\mathbb{E}||\hat{G}_t(\theta_t)||_2^2\right)^{-1}$. We call $\left(\hat{C}\mathbb{E}||\hat{G}_t(\theta_t)||_2^2\right)^{-1}$ the *relative optimization efficiency*, or ROE. We decide between gradient estimators $\hat{G}$ by choosing the one which maximizes the ROE. Once an estimator is chosen, one should choose a learning rate according to (12) relative to a reference learning rate $\bar{\eta}$ and estimator $\bar{G}$.

### 5.2. Optimal weighted sampling for RT estimators

Now that we have an objective, we can consider designing RT estimators which optimize the ROE. For the classes of single sample and Russian roulette estimators, we prove conditions under which that class maximizes the ROE across an arbitrary choice of RT estimators. We also derive closed-form expressions for the optimal sampling distribution $q$ for each class, under the conditions where that class is optimal.

We assume that computation can be reused and evaluating $\hat{G}_H = \sum_{n=1}^N \Delta_n W(n, N)$ has computation cost $C(N)$. As described in Section 3.1, this is approximately true for many objectives. When it is not, the cost of computing $\sum_{n=1}^N \Delta_n W(n, N)$ is $\sum_{n=1}^N C(n)\mathbb{1}\{(W(n, N) \neq 0) \text{ or } (W(n+1, N) \neq 0)\}$. This would penalize the ROE of dense $W(n, N)$ and favor

sparse $W(n, N)$, possibly impacting the optimality conditions for RT-RR. We mitigate this inaccuracy by subsequence selection (described in the following subsection), which allows construction of sparse sampling strategies.

We begin by showing the RT-SS estimator is optimal with regards to worst-case diagonal covariances $\mathrm{Cov}(\Delta_i, \Delta_j)$, and deriving the optimal $q(N)$.

**Theorem 5.1.** ***Optimality of RT-SS under adversarial correlation.*** *Consider the family of estimators presented in Equation 2. Assume $\theta$, $\nabla_\theta$, and $G$ are univariate. For any fixed sampling distribution $q$, the single-sample RT estimator RT-SS minimizes the worst-case variance of $\hat{G}$ across an adversarial choice of covariances $\mathrm{Cov}(\Delta_i, \Delta_j) \leq \sqrt{\mathrm{Var}(\Delta_i)}\sqrt{\mathrm{Var}(\Delta_j)}$.*

**Theorem 5.2.** ***Optimal q under adversarial correlation.*** *Consider the family of estimators presented in Equation 2. Assume $\mathrm{Cov}(\Delta_i, \Delta_i)$ and $\mathrm{Cov}(\Delta_i, \Delta_j)$ are diagonal. The RT-SS estimator with $q_n \propto \sqrt{\frac{\mathbb{E}[||\Delta_n||_2^2}{C(n)}}$ maximizes the ROE across an adversarial choice of diagonal covariance matrices $\mathrm{Cov}(\Delta_i, \Delta_j)_{kk} \leq \sqrt{\mathrm{Cov}(\Delta_i, \Delta_i)_{kk}\mathrm{Cov}(\Delta_j, \Delta_j)_{kk}}$.*

We next show the RT-RR estimator is optimal when $\mathrm{Cov}(\Delta_i, \Delta_i)$ is diagonal and $\Delta_i$ and $\Delta_j$ are independent for $j \neq i$, and derive the optimal $q(N)$.

**Theorem 5.3.** ***Optimality of RT-RR under independence***. *Consider the family of estimators presented in Eq. 2. Assume the $\Delta_j$ are univariate. When the $\Delta_j$ are uncorrelated, for any importance sampling distribution $q$, the Russian roulette estimator achieves the minimum variance in this family and thus maximizes the optimization efficiency lower bound.*

**Theorem 5.4.** ***Optimal q under independence***. *Consider the family of estimators presented in Equation 2. Assume $\mathrm{Cov}(\Delta_i, \Delta_i)$ is diagonal and $\Delta_i$ and $\Delta_j$ are independent. The RT-RR estimator with $Q(i) \propto \sqrt{\frac{\mathbb{E}[||\Delta_i||_2^2}{C(i)-C(i-1)}]}$, where $Q(i) = \Pr(n \geq i) = \sum_{j=i}^{H} q(j)$, maximizes the ROE.*

### 5.3. Subsequence selection

The scheme for designing RT estimators given in the previous subsection contains assumptions which will often not hold in practice. To partially alleviate these concerns, we can design the *sequence of iterates over which we apply the RT estimator* to maximize the ROE.

Some sequences may result in more efficient estimators, depending on how the intermediate iterates $G_n$ correlate with $G$. The variance of the estimator, and the ROE, will be reduced if we choose a sequence $\mathcal{L}_n$ such that $G_n$ is positively correlated with $G$ for all $n$.

We begin with a reference sequence $\bar{\mathcal{L}}_i$, $\bar{G}_i$, with cost function $\bar{C}$, where $i, j \in \mathcal{N}$ and $i, j \leq \bar{H}$, and where $\bar{G}_i$ has cost $\bar{c}_i$. We assume knowledge of $\mathbb{E}[||\bar{G}_i - \bar{G}_j||_2^2]$. We aim to find a subsequence $S \in \mathcal{S}$, where $\mathcal{S}$ is the set of subsequences over the integers $1, \ldots, \bar{H}$ which have final element $S_{-1} = \bar{H}$. Given $S$, we take $\mathcal{L}_n = \bar{\mathcal{L}}_{S_n}$, $G_n = \bar{G}_{S_n}$, $C(n) = \bar{C}(S_n)$, $H = |S|$, and $\Delta_n = G_n - G_{n-1}$, where $G_0 := 0$.

In practice, we greedily construct $S$ by adding indexes $i$ to the sequence $[\bar{H}]$ or removing indexes $i$ from the sequence $[1, \ldots, \bar{H}]$. As this step requires minimal computation, we perform both greedy adding and greedy removal and return the $S$ with the best ROE. The minimal subsequence $S = [\bar{H}]$ is always considered, allowing RT estimators to fall back on the original full-horizon estimator.

## 6. Practical implementation

### 6.1. Tuning the estimator

We estimate the expected squared distances $\mathbb{E}[||\bar{G}_i - \bar{G}_j||_2^2]$ by maintaining exponential moving averages. We keep track of the computational budget $B$ used so far by the RT estimator, and "tune" the estimator every $K\bar{C}(\bar{H})$ units of computation, where $\bar{C}(\bar{H})$ is the compute required to evaluate $\bar{G}_{\bar{H}}$, and $K$ is a "tuning frequency" hyperparameter. During tuning, the gradients $G_i$ are computed, the squared norms $||\bar{G}_i - \bar{G}_j||_2^2$ are computed, and the exponential moving averages are updated. At the end of tuning, the estimator is updated using the expected squared norms; i.e. a subsequence is selected, $q$ is set according to section 5.2 with choice of RT-RR or RT-SS left as a hyperparameter, and the learning rate is adapted according to section 5.1

### 6.2. Controlling sequence length

Tuning and subsequence selection require computation. Consider using RT to optimize an objective with an inner loop of size $M$. If we let $\bar{G}_i$ be the gradient of the loss after $i$ inner steps, we must maintain $M^2 - M$ exponential moving averages $\mathbb{E}||\bar{G}_i - \bar{G}_j||_2^2$, and compute $M$ gradients $\bar{G}_i$ each time we tune the estimator. The computational cost of the tuning step under this scheme is $\mathcal{O}(M^2)$. This is unacceptable if we wish our method to scale well with the size of loops we might wish to optimize.

To circumvent this, we choose base subsequences such that $\bar{C}_i \propto 2^i$. This ensures that $\bar{H} = \mathcal{O}(\log_2 M)$, where $M$ is the maximum number of steps we wish to unroll. We must maintain $\mathcal{O}(\log_2^2 M)$ exponential moving averages. Computing the gradients $\bar{G}_i$ during each tuning step requires compute $C_{\text{tune}} = \sum_{i=1}^{\bar{H}} k * 2^i$. Noting that $\bar{C}_{\bar{H}} = k * 2^{\bar{H}}$ and that $\sum_{i=1}^{N} 2^i < 2^{N+1} \forall N$ yields $C_{\text{tune}} < 2\bar{C}_{\bar{H}} = 2M$.

## 7. Experiments

For all experiments, we tune learning rates for the full-horizon un-truncated estimator via grid search over all $a \times 10^{-b}$, for $a \in \{1.0, 2.2, 5.5\}$ and $b \in \{0.0, 1.0, 2.0, 3.0, 5.0\}$. The same learning rates are used for the truncated estimators and (as reference learning rates) for the RT estimators. We do not decay the learning rate. Experiments are run with the random seeds $0, 1, 2, 3, 4$ and we plot means and standard deviation.

We use the same hyperparameters for our online tuning procedure for all experiments: the tuning frequency $K$ is set to 5, and the exponential moving average weight $\alpha$ is set to $0.9$. These hyperparameters were not extensively tuned. For each problem, we compare deterministic, RT-SS, and RT-RR estimators, each with a range of truncations.

### 7.1. Lotka-Volterra ODE

We first experiment with variational inference of parameters of a Lotka-Volterra (LV) ODE. LV ODEs are defined by the predator-prey equations, where $u_2$ and $u_1$ are predator and prey populations, respectively:

$$\frac{du_1}{dt} = Au_1 - Bu_1u_2 \qquad \frac{du_2}{dt} = Cu_1u_2 - Du_2$$

We aim to infer the parameters $\lambda = [u_1(t=0), u_2(t=0), A, B, C, D]$. The true parameters are drawn from $\mathcal{U}([1.0, 0.4, 0.8, 0.4, 1.5, 0.4], [1.5, 0.6, 1.2, 0.6, 2.0, 0.6])$, chosen empirically to ensure stability solving the equations. We generate ground-truth data by solving the equations using RK4 (a common 4th-order Runge Kutta method) from $t = 0$ to $t = 5$ with 10000 steps. The learner is given access to five equally spaced noisy observations $y(t)$, generated according to $y(t) = u(t) + \mathcal{N}(0, 0.1)$.

We place a diagonal Gaussian prior on $\theta$ with the same mean and standard deviation as the data-generating distribution. The variational posterior is a diagonal Gaussian $q(\lambda)$ with mean $\mu$ and standard deviation $\sigma$. The parameters optimized are $\theta = [\tilde{\mu}, \tilde{\sigma}]$. We let $\mu = g(\tilde{\mu})$ and $\sigma = g(\tilde{\sigma})$, where $g(\tilde{x}) = \log(1 + e^{\tilde{x}})$, to ensure positivity. We use a reflecting boundary to ensure positivity of parameter samples from $q$. The variational posterior is initialized to have mean equal to the prior and standard deviation 0.1.

The loss considered is the negative evidence lower bound (negative ELBO). The ELBO is:

$$\text{ELBO}(q(\theta)) = \mathbb{E}_{q(\theta)} \sum_t \log p(y(t)|u_\theta(t)) + D_{\text{KL}}\big(q(\theta)||p(\theta)\big)$$

Above, $u_\theta(t)$ is the value of the solution $u_\theta$ to the LV ODE with parameters $\theta$, evaluated at time $t$. We consider a sequence $\mathcal{L}_n(\theta)$, where in computing the ELBO, $u_\theta(t)$ is approximated by solving the ODE using RK4 with $2^n + 1$ steps, and linearly interpolating the solution to the 5 observation times. The outer-loop optimization is performed with a batch size of 64 (i.e., 64 samples of $\theta$ are performed at each step) and a learning rate of $0.01$. Evaluation is performed with a batch size of 512.

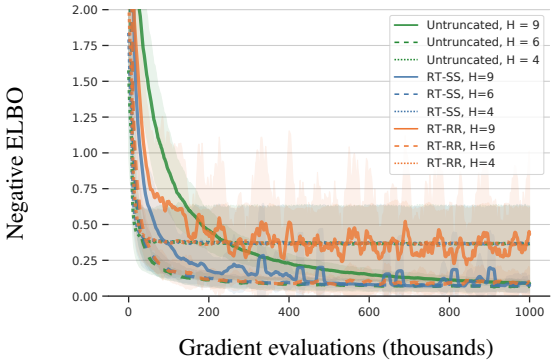


*Figure 1.* Lotka-Volterra parameter inference

Figure 7.1 shows the loss of the different estimators over the course of training. RT-SS estimators outperform the un-truncated estimator without inducing bias. They are competitive with the truncation $H = 6$, while avoiding the bias present with the truncation $H = 4$, at the cost of some variance. Some RT-RR estimators experience issues with optimization, appearing to obtain the same biased solution as the $H = 4$ truncation.

### 7.2. MNIST learning rate

We next experiment with meta-optimization of a learning rate on MNIST. We largely follow the procedure used by Wu et al. (2018). We use a feedforward network with two hidden layers of 100 units, with weights initialized from a Gaussian with standard deviation 0.1, and biases initialized to zero. Optimization is performed with a batch size of 100.

The neural network is trained by SGD with momentum using Polyak averaging, with the momentum parameter fixed to $0.9$. We aim to learn a learning rate $\eta_0$ and decay $\lambda$ for the inner-loop optimization. These are initialized to $0.01$ and $0.1$ respectively. The learning rate for the inner optimization at an inner optimization step $t$ is $\eta_t = \eta_0(1 + \frac{t}{5000})^{-\lambda}$.

As in Wu et al. (2018), we pre-train the net for 50 steps with a learning rate of $0.1$. $\mathcal{L}_n$ is the evaluation loss after $2^n + 1$ training steps with a batch size of 100. The evaluation loss is measured over $2^n + 1$ validation batches or the entire validation set, whichever is smaller. The outer optimization is performed with a learning rate of $0.01$.

RT-SS estimators achieve faster convergence than fixed-truncation estimators. RT-RR estimators suffer from very poor convergence. Truncated estimators appear to obtain biased solutions. The un-truncated estimator achieves a slightly better loss than the RT estimators, but takes significantly longer to converge.

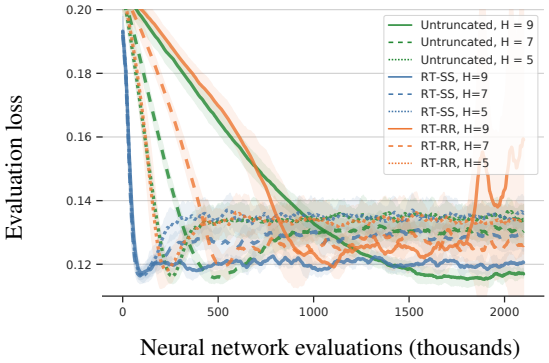


*Figure 2.* MNIST learning rate meta-optimization

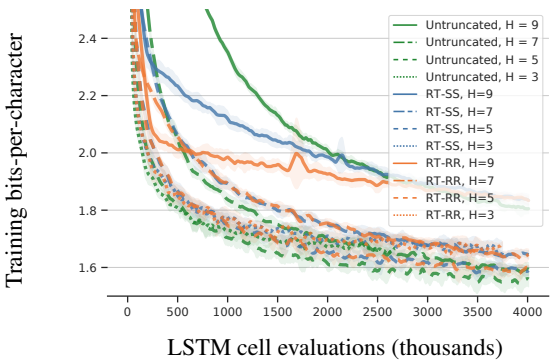


*Figure 3.* LSTM training on `enwik8`

### 7.3. `enwik8` LSTM

Finally, we study a high-dimensional optimization problem: training an LSTM to model sequences on `enwik8`. These data are the first 100M bytes of a Wikipedia XML dump. There are 205 unique tokens. We use the first 90M, 5M, and 5M characters as the training, evaluation, and test sets.

We build on code[1] from Merity et al. (2017; 2018). We train an LSTM with 1000 hidden units and 400-dimensional input and output embeddings. The model has 5.9M parameters. The only regularization is an $\ell_2$ penalty on the weights with magnitude $10^{-6}$. The optimization is performed with a learning rate of 2.2. This model is not state-of-the-art: our aim to investigate performance of RT estimators for optimizing high-dimensional neural networks, rather than to maximize performance at a language modeling task.

We choose $\mathcal{L}_n$ to be the mean cross-entropy after unrolling the LSTM training for $2^{n-1}+1$ steps. We choose the horizon $H = 9$, such that the un-truncated loop has 257 steps, chosen to be close to the 200-length training sequences used by Merity et al. (2018).

Figure 7.3 shows the training bits-per-character (proportional to the training cross-entropy loss). RT estimators provide some acceleration over the un-truncated $H = 9$ estimator early in training, but after about 200k cell evaluations, fall back on the un-truncated estimator, subsequently progressing slightly more slowly due to computational cost of tuning. We conjecture that the diagonal covariance assumption in Section 5 is unsuited to high-dimensional problems, and leads to overly conservative estimators.

## 8. Limitations and future work

**Other optimizers**. We develop the lower bound on expected improvement for SGD. Important future directions would investigate adaptive and momentum-based SGD methods such as Adam (Kingma & Ba, 2014).

**Tuning step**. Our method includes a tuning step which requires computation. It might be possible to remove this tuning step by estimating covariance structure online using just the values of $\hat{G}$ observed during each optimization step.

**RT estimators beyond RT-SS and RT-RR**. There is a rich family defined by choices of $q$ and $W(n, N)$. The optimal member depends on covariance structure between the $G_i$. We explore RT-SS and RT-RR under strict covariance assumptions. Relaxing these assumptions and optimizing $q$ and $W$ across a wider family could improve adaptive estimator performance for high-dimensional problems such as training RNNs.

**Predictive models of the sequence limit**. Using any sequence $G_n$ with RT yields an unbiased estimator as long as the sequence is consistent, i.e. its limit $G$ is the true gradient. Combining randomized telescopes with predictive models of the gradients (Jaderberg et al., 2017; Weber et al., 2019) might yield a fast-converging sequence, leading to estimators with low computation and variance.

## 9. Conclusion

We investigated the use of randomly truncated unbiased gradient estimators for optimizing objectives which involve loops and limits. We proved these estimators can achieve horizon-independent convergence rates for optimizing loops and limits. We derived adaptive variants which can be tuned online to maximize a lower bound on expected improvement per unit computation. Experimental results matched theoretical intuitions that the single sample estimator is more robust than Russian roulette for optimization. The adaptive RT-SS estimator often significantly accelerates optimization, and can otherwise fall back on the un-truncated estimator.

## 10. Acknowledgements

We would like to thank Matthew Johnson, Peter Orbanz, and James Saunderson for helpful discussions. This work was funded by the Alfred P. Sloan Foundation and NSF IIS-1421780.

[1] `http://github.com/salesforce/awd-lstm-lm`

## References


Arvo, J. and Kirk, D. Particle transport and image synthesis. *ACM SIGGRAPH Computer Graphics*, 24(4):63–66, 1990.

Balles, L., Romero, J., and Hennig, P. Coupling adaptive batch sizes with learning rates. In *Uncertainty in Artificial Intelligence*, 2016.

Bengio, S., Bengio, Y., Cloutier, J., and Gecsei, J. On the optimization of a synaptic learning rule. In *Conference on Optimality in Artificial and Biological Neural Networks*, 1992.

Bottou, L., Curtis, F. E., and Nocedal, J. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018.

Brock, A., Lim, T., Ritchie, J., and Weston, N. SMASH: One-shot model architecture search through hypernetworks. In *International Conference on Learning Representations*, 2018.

Christianson, B. Reverse accumulation and implicit functions. *Optimization Methods and Software*, 9(4):307–322, 1998.

Cremer, C., Li, X., and Duvenaud, D. Inference suboptimality in variational autoencoders. *arXiv preprint arXiv:1801.03558*, 2018.

Fearnhead, P., Papaspiliopoulos, O., and Roberts, G. O. Particle filters for partially observed diffusions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(4):755–777, 2008.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, 2017.

Forsythe, G. E. and Leibler, R. A. Matrix inversion by a Monte Carlo method. *Mathematics of Computation*, 4 (31):127–129, 1950.

Franceschi, L., Donini, M., Frasconi, P., and Pontil, M. Forward and reverse gradient-based hyperparameter optimization. In *International Conference on Machine Learning*, 2017.

Girolami, M., Lyne, A.-M., Strathmann, H., Simpson, D., and Atchade, Y. Playing Russian roulette with intractable likelihoods. Technical report, Citeseer, 2013.

Jacob, P. E., O'Leary, J., and Atchadé, Y. F. Unbiased Markov chain Monte Carlo with couplings. *arXiv preprint arXiv:1708.03625*, 2017.

Jaderberg, M., Czarnecki, W. M., Osindero, S., Vinyals, O., Graves, A., Silver, D., and Kavukcuoglu, K. Decoupled neural interfaces using synthetic gradients. In *International Conference on Machine Learning*, pp. 1627–1635. JMLR. org, 2017.

Jameson, A. Aerodynamic design via control theory. *Journal of Scientific Computing*, 3(3):233–260, 1988.

Kahn, H. Use of different Monte Carlo sampling techniques. 1955.

Kim, Y., Wiseman, S., Miller, A. C., Sontag, D., and Rush, A. M. Semi-amortized variational autoencoders. In *International conference on Machine Learning*, 2018.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2014.

Kingma, D. P. and Welling, M. Auto-encoding variational Bayes. In *International Conference on Learning Representations*, 2014.

Kuti, J. Stochastic method for the numerical study of lattice fermions. *Physical Review Letters*, 49(3):183, 1982.

Lorraine, J. and Duvenaud, D. Stochastic hyperparameter optimization through hypernetworks. *arXiv preprint arXiv:1802.09419*, 2018.

Lyne, A.-M., Girolami, M., Atchadé, Y., Strathmann, H., Simpson, D., et al. On Russian roulette estimates for Bayesian inference with doubly-intractable likelihoods. *Statistical science*, 30(4):443–467, 2015.

Maclaurin, D., Duvenaud, D., and Adams, R. Gradient-based hyperparameter optimization through reversible learning. In *International Conference on Machine Learning*, pp. 2113–2122, 2015.

McLeish, D. A general method for debiasing a Monte Carlo estimator. *Monte Carlo Methods and Applications*, 2010.

Merity, S., Keskar, N. S., and Socher, R. Regularizing and optimizing LSTM language models. *arXiv preprint arXiv:1708.02182*, 2017.

Merity, S., Keskar, N. S., and Socher, R. An analysis of neural language modeling at multiple scales. *arXiv preprint arXiv:1803.08240*, 2018.

Metz, L., Maheswaranathan, N., Nixon, J., Freeman, C. D., and Sohl-Dickstein, J. Learned optimizers that outperform SGD on wall-clock and validation loss. *arXiv preprint arXiv:1810.10180*, 2018.

Ravi, S. and Larochelle, H. Optimization as a model for few-shot learning. In *International Conference on Learning Representations*, 2016.

Rhee, C.-h. and Glynn, P. W. A new approach to unbiased estimation for SDEs. In *Proceedings of the Winter Simulation Conference*, pp. 17. Winter Simulation Conference, 2012.

Rhee, C.-h. and Glynn, P. W. Unbiased estimation with square root convergence for SDE models. *Operations Research*, 63(5):1026–1043, 2015.

Rudin, W. et al. *Principles of Mathematical Analysis*, volume 3. McGraw-hill New York, 1976.

Rychlik, T. Unbiased nonparametric estimation of the derivative of the mean. *Statistics & probability letters*, 10 (4):329–333, 1990.

Rychlik, T. A class of unbiased kernel estimates of a probability density function. *Applicationes Mathematicae*, 22 (4):485–497, 1995.

Schmidhuber, J. *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook*. PhD thesis, Technische Universität München, 1987.

Shaban, A., Cheng, C.-A., Hatch, N., and Boots, B. Truncated back-propagation for bilevel optimization. *arXiv preprint arXiv:1810.10667*, 2018.

Spanier, J. and Gelbard, E. M. *Monte Carlo Principles and Neutron Transport Problems*. Addison-Wesley Publishing Company, 1969.

Tallec, C. and Ollivier, Y. Unbiasing truncated backpropagation through time. *arXiv preprint arXiv:1705.08209*, 2017.

Trinh, T. H., Dai, A. M., Luong, T., and Le, Q. V. Learning longer-term dependencies in RNNs with auxiliary losses. *arXiv preprint arXiv:1803.00144*, 2018.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Neural Information Processing Systems*, pp. 5998–6008, 2017.

Wagner, W. Unbiased Monte Carlo evaluation of certain functional integrals. *Journal of Computational Physics*, 71(1):21–33, 1987.

Weber, T., Heess, N., Buesing, L., and Silver, D. Credit assignment techniques in stochastic computation graphs. *arXiv preprint arXiv:1901.01761*, 2019.

Wei, C. and Murray, I. Markov chain truncation for doubly-intractable inference. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2016.

Wu, Y., Ren, M., Liao, R., and Grosse., R. Understanding short-horizon bias in stochastic meta-optimization. In *International Conference on Learning Representations*, 2018.