Unbiased Gradient Estimation in Unrolled Computation Graphs with Persistent Evolution Strategies (Extended Abstract)

Paul Vicol\textsuperscript{1}, Luke Metz\textsuperscript{2}, Jascha Sohl-Dickstein\textsuperscript{2}

\textsuperscript{1}University of Toronto  
\textsuperscript{2}Google Brain

pvicol@cs.toronto.edu, \{lmetz, jaschasd\}@google.com

Abstract

Current approaches for optimizing parameters in unrolled computation graphs suffer from high variance gradients, bias, slow updates, or large memory usage. We introduce a method called Persistent Evolution Strategies (PES), which divides the computation graph into a series of truncated unrolls, and performs an evolution strategies-based update step after each unroll. PES eliminates bias from these truncations by accumulating correction terms over the entire sequence of unrolls. PES allows for rapid parameter updates, has low memory usage, is unbiased, and has reasonable variance.

1 Introduction

Unrolled computation graphs arise in many scenarios in machine learning, including when training RNNs [Williams and Peng, 1990], tuning hyperparameters through unrolled computation graphs [Franceschi et al., 2018], and training learned optimizers [Wichrowska et al., 2017; Metz et al., 2018].

Classic algorithms for computing gradients in such unrolled computation graphs include reverse- and forward-mode gradient accumulation. Backpropagation through time (BPTT) involves backpropagating through a full unrolled sequence for each parameter update. This faces several difficulties: 1) the memory cost scales linearly with the unroll length, due to storage of intermediate activations for backprop 2) it only performs one parameter update after each full unroll, which is expensive and introduces large latency between parameter updates; 3) long unrolls can lead to exploding or vanishing gradients [Pascanu et al., 2013], and chaotic loss landscapes [Parmas et al., 2018; Metz et al., 2019].

A common technique to alleviate these issues is truncated backprop through time (TBPTT) [Werbos, 1990], which splits the full sequence into shorter subsequences and performs an update after processing each subsequence. However, TBPTT yields biased gradients that can severely impact training.

An alternative to BPTT is real-time recurrent learning (RTRL), which performs forward gradient accumulation [Williams and Zipser, 1989]. RTRL enables online parameter updates (after each partial unroll) and does not suffer from truncation bias; however, its memory and compute requirements render it intractable for large-scale problems. Many approximations to RTRL have been proposed [Tallec and Ollivier, 2017; Mujika et al., 2018; Benzing et al., 2019], but most have high variance, are complicated to implement, or are only applicable to a restricted class of models.

To address the poorly conditioned loss surfaces that often result from sequential computation, it can be useful to minimize a smoothed version of the loss. Evolution strategies (ES) is a family of algorithms that estimate gradients using stochastic finite-differences, and which provide an unbiased estimate of the gradient of the objective smoothed with a Gaussian. We introduce an approach to unbiased gradient estimation using short, truncated unrolls, called Persistent Evolution Strategies (PES). In PES, we accumulate the perturbations experienced by the outer parameters in each partial unroll—rather than starting perturbations from scratch as in ES—which yields an unbiased estimate of the gradient even when using truncated sequences. PES is simple to implement, and because it is ES-based, it retains desirable characteristics such as being parallelizable, memory efficient, and broadly applicable to many problems, including to non-differentiable target functions.

Problem Setup. We consider unrolled computation graphs with state $s_t$ updated based on parameters $\theta$ via the recurrence:

$$s_t = f(s_{t-1}, x_t; \theta)$$

where $x_t$ is an optional input at step $t$. The objective function for optimizing $\theta$ is the sum of per-timestep losses $L_t(s_t; \theta)$:

$$L(\theta) = \sum_{t=1}^T L_t(s_t; \theta)$$

This setup is general, even encompassing situations where we want to consider only the final loss at step $T$, which can be expressed using a telescoping sum of loss differences between successive steps [Beatson and Adams, 2019]. Instances of this problem setup include training RNNs, training learned optimizers, learning policies for control tasks, and hyperparameter optimization, as illustrated in Figure 1.

Smoothing & Evolution Strategies (ES). Unrolling optimization for many steps can lead to pathological meta-loss
surfaces that exhibit near-discontinuities and chaotic structure [Parnas et al., 2018; Metz et al., 2019]. Optimization on such non-smooth landscapes fails due to exploding gradients or gets stuck in poor local minima. An effective method to address these pathologies is to smooth the meta-loss surface, e.g. descend the Gaussian-blurred objective $L(\Theta) = \mathbb{E}_{\theta \sim \mathcal{N}(\theta; \sigma^2 I)} [L(\theta)]$ [Staines and Barber, 2012; Metz et al., 2019]. Evolution strategies provide an unbiased estimate of the gradient $\nabla_\theta \mathbb{E}_{\theta \sim \mathcal{N}(\theta; \sigma^2 I)} [L(\theta)]$. The ES estimator is defined as: $g^\text{ES} = \frac{1}{N^2 \sigma} \sum_{i=1}^{N^2} \epsilon^{(i)} L(\theta + \epsilon^{(i)})$, where $\epsilon^{(i)} \sim \mathcal{N}(0, \sigma^2 I)$. ES is trivially parallelizable, and thus is highly scalable; it has seen renewed interest as a viable algorithm for reinforcement learning among other black-box problems [Ha and Schmidhuber, 2018; Cui et al., 2018; Ha, 2020]. However, applying ES to full unrolls is costly, while using partial unrolls leads to truncation bias, similarly to TBPTT.

2 Persistent Evolution Strategies

We introduce a method to obtain unbiased gradient estimates from partial unrolls of a computation graph, called Persistent Evolution Strategies (PES). We first derive the PES gradient estimator, and then present a practical algorithm (Algorithm 2).

Derivation. Unrolled computation graphs (as illustrated in Figure 1) depend on shared parameters $\theta$ at every timestep; in order to account for how these contribute to the overall gradient $\nabla_\theta L(\theta)$, we use subscripts $t$ to distinguish between applications of $\theta$ at different steps, where $\theta_t = \theta \forall t$. We further define $\Theta = (\theta_1, \ldots, \theta_T)^T$, which is a matrix with the per-timestep $\theta_t$ as its rows. For notational simplicity in the following derivation, we drop the dependence on $s_t$ and explicitly include the dependence on each $\theta_t$, writing $L_t(s_t; \theta_t)$ as either $L_t(\theta_1, \ldots, \theta_t)$ or simply $L_t(\theta)$. We wish to compute the gradient $\nabla_\theta L(\theta)$ of the total loss over all unrolls. We begin by writing this gradient in terms of the full gradient $\nabla L(\theta) / \nabla \text{vec}(\Theta) \in \mathbb{R}^{PT \times 1}$, and then using ES to approximate

$$g^\text{ES} = \frac{1}{\sigma^2} \mathbb{E}_\epsilon \left[ \frac{1}{T} \text{vec}(\epsilon) L(\theta + \epsilon) \right]$$

where $\otimes$ denotes the Kronecker product, $\epsilon = (\epsilon_1, \ldots, \epsilon_T)^T$ is a matrix of perturbations $\epsilon_i$ to be added to the $\theta_i$ at each timestep, and the expectation is over entries in $\epsilon$ drawn from an i.i.d. Gaussian with variance $\sigma^2$. This ES approximation is an unbiased estimator of the gradient of the Gaussian-smoothed objective $\mathbb{E}_\epsilon [L(\theta + \epsilon)]$. We next show that $g^\text{PES}$ decomposes into a sum of sequential gradient estimates,

$$g^\text{PES} = \frac{1}{\sigma^2} \mathbb{E}_\epsilon \left[ \frac{1}{T} \sum_{t=1}^{T} L_t(\theta + \epsilon) \right]$$

$$g^\text{PES} = \frac{1}{\sigma^2} \mathbb{E}_\epsilon \left[ \frac{1}{T} \sum_{t=1}^{T} L_t(\theta + \epsilon) \right]$$

where $\xi_t = \sum_{\tau=1}^{t} \epsilon_\tau$, Equation 3 relies on $L_t(\cdot)$ being independent of $\epsilon_\tau$ for $\tau > t$, and Equation 5 similarly relies on $L_t(\cdot)$ only being a function of $\theta_t$ for $\tau \leq t$. The PES estimator consists of Monte Carlo estimates of Equation 4,

$$g^\text{PES} = \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} g^\text{PES}_{\epsilon^{(i)}}$$

where $\epsilon^{(i)}$ are samples of $\epsilon$, and $N$ is the number of Monte Carlo samples. Gradient estimates at each time step can be evaluated sequentially, and used to perform SGD. When using antithetic sampling (e.g., positive and negative perturbation pairs in Eq. 6), we denote the estimator by $g^\text{PES} - A$. See the full paper for a proof of the following Statement 2.1.

Statement 2.1 (PES is unbiased). Let $\theta \in \mathbb{R}^P$ and $L(\theta) = \sum_{t=1}^{T} L_t(\theta)$. Suppose that $\nabla_\theta L(\theta)$ exists, and assume that $L$ is quadratic, so that it is equivalent to its second-order Taylor series expansion: $L(\theta + \epsilon) = L(\theta) + \text{vec}(\epsilon)^\top \nabla \text{vec}(\theta) L(\theta) + \frac{1}{2} \text{vec}(\epsilon)^\top \nabla^2 \text{vec}(\theta) L(\theta) \text{vec}(\epsilon)$. Then, $\mathbb{E}_\epsilon [g^\text{PES} - A] = \mathbb{E}_\epsilon [g^\text{PES} - A] - \nabla_\theta L(\theta) = 0$.

Algorithm. Based on Eq. 6, we see that we can obtain unbiased gradient estimates from partial unrolls by: 1) not resetting the particles between unrolls, and 2) accumulating the perturbations $\xi_t$ each particle has experienced over multiple unrolls.

Figure 1: An unrolled computation graph, illustrating how several tasks can be described using Equations 1 and 2. For example, in unrolled optimization, $s_t$ contains the parameters of the base model and optimizer accumulators (e.g. momentum), $L_t(\cdot)$ is a meta-objective such as validation performance, $\theta$ contains hyperparameters (e.g. the learning rate, weight decay, etc.) that govern the optimization, and $f$ corresponds to the update step of an optimization algorithm such as SGD, RMSprop [Tieleman and Hinton, 2012], or Adam [Kingma and Ba, 2015].
Truncated Evolution Strategies (ES) applied to partial unrolls of a computation graph.

**Input:** $s_0$, initial state
- $K$, truncation length for partial unrolls
- $N$, number of particles
- $\sigma$, standard deviation of perturbations
- $\alpha$, learning rate for ES optimization

Initialize $s = s_0$

While true do
  $g^{ES} \leftarrow 0$
  For $i = 1, \ldots, N$ do
    $\epsilon^{(i)} = \begin{cases} \text{draw from } \mathcal{N}(0, \sigma^2 I) & \text{odd} \\ -\epsilon^{(i-1)} & \text{even} \end{cases}$
    $\hat{L}^{(i)}_K \leftarrow \text{unroll}(s, \theta + \epsilon^{(i)}, K)$
    $\hat{g}^{ES} \leftarrow \hat{g}^{ES} + \epsilon^{(i)} \hat{L}^{(i)}_K$
  End for
  $g^{ES} \leftarrow \frac{1}{N\sigma^2} \hat{g}^{ES}$
  $s \leftarrow \text{unroll}(s, \theta, K)$
  $\theta \leftarrow \theta - \alpha g^{ES}$
End while

Persistent evolution strategies (PES).

**Input:** $s_0$, initial state
- $K$, truncation length for partial unrolls
- $N$, number of particles
- $\sigma$, standard deviation of perturbations
- $\alpha$, learning rate for PES optimization

Initialize $s^{(i)} = s_0$ for $i \in \{1, \ldots, N\}$

Initialize $\xi^{(i)} \leftarrow 0$ for $i \in \{1, \ldots, N\}$

While true do
  $g^{PES} \leftarrow 0$
  For $i = 1, \ldots, N$ do
    $\epsilon^{(i)} = \begin{cases} \text{draw from } \mathcal{N}(0, \sigma^2 I) & \text{odd} \\ -\epsilon^{(i-1)} & \text{even} \end{cases}$
    $s^{(i)}, \hat{L}^{(i)}_K \leftarrow \text{unroll}(s^{(i)}, \theta + \epsilon^{(i)}, K)$
    $\xi^{(i)} \leftarrow \xi^{(i)} + \epsilon^{(i)}$
    $g^{PES} \leftarrow g^{PES} + \xi^{(i)} \hat{L}^{(i)}_K$
  End for
  $g^{PES} \leftarrow \frac{1}{N\sigma^2} g^{PES}$
  $\theta \leftarrow \theta - \alpha g^{PES}$
End while

The resulting algorithm is simple to implement, requiring only minor modifications from vanilla ES. Algorithm 1 describes truncated ES applied to partial unrolls, where it suffers from short horizon bias. Algorithm 2 shows PES applied to the same problem, where it provides unbiased gradient estimates. Both algorithms (Fig. 2) are shown with antithetic sampling (perturbations are paired with their negations), which drastically reduces variance.

### 3 Experiments

![Swimmer-v2](image)

**Figure 4:** Learning a policy for continuous control. We train a linear policy on the Swimmer-v2 MuJoCo environment, following [Mania et al., 2018]. We find that PES applied to truncated unrolls performs similarly to ES applied to full episodes, while truncated ES fails due to bias. We plot the ARS V1 result from [Mania et al., 2018] (dashed curve) to show that our full-unroll baseline is comparable to theirs. The dotted line shows the maximum reward reported for the ES approach in [Salimans et al., 2017], which does not solve the Swimmer task.

In experiments, we use PES to meta-train a learned optimizer (Figure 3), and learn a policy for continuous control (Figure 4). We additionally demonstrate the use of PES for hyperparameter optimization, where we show that PES performs well when the meta-loss has many local minima (Figure 5), does not suffer from truncation bias (Figure 6a), can be applied to non-differentiable objectives (Figure 6b), and can be used to optimize many hyperparameters (both continuous
Figure 5: The meta-objective surface (left), and meta-objective vs inner problem timesteps (right), for a 2D regression problem with many local minima to which truncated methods could converge. We plot meta-optimization trajectories for TBPTT, UORO, RTRL, ES, and PES starting from the same initialization, $(-4.5, -4.5)$ in log-space. All techniques except PES either suffer from truncation bias, or become stuck due to high-frequency structure in the meta-objective surface. PES is both unbiased, and smooths the outer-objective removing high-frequency structure.

Figure 6: Meta-optimization of a learning rate schedule for an MLP on MNIST. We tune the initial learning rate and decay factor, both parameterized in log-space. Here we show the meta-loss landscape, and the optimization trajectories taken by ES and PES, for unroll lengths $K \in \{10, 100\}$. The meta-objective in (a) is the training loss, while the meta-objective in (b) is the validation accuracy. In both visualizations, darker colors denote better values. We see that ES converges to a suboptimal region of the hyperparameter space due to truncation bias, while PES finds the correct solution. Because PES only requires function evaluations and not gradients, it can optimize non-differentiable objectives such as accuracy rather than loss. This is demonstrated in subfigure (b).

Figure 7: Meta-optimization of per-parameter-block learning rates and momentum coefficients (29 hyperparameters total). We tuned both continuous and discrete hyperparameters: the number of units per hidden layer (discrete architectural hyperparameters) and per-parameter-block learning rates and momentum coefficients (continuous hyperparameters). We trained a 5-hidden-layer MLP (6 layers including the output layer mapping to logits) on FashionMNIST, yielding 29 hyperparameters in total.

and discrete) simultaneously (Figure 7). A detailed description of each experimental setup is given in the full paper, [Vicol et al., 2021]. A tutorial Colab notebook demonstrating the technique, and reproducing some of the experimental results, is available at this link.

4 Conclusion

We introduce a method called Persistent Evolution Strategies (PES), which divides a computation graph into a series of truncated unrolls, and performs an evolution strategies-based update after each unroll. PES eliminates bias from these truncations by accumulating correction terms over the entire sequence of unrolls. PES allows for rapid parameter updates, has low memory usage, is unbiased, and has reasonable variance characteristics. We show that PES is broadly applicable, with experiments demonstrating its use in hyperparameter optimization, reinforcement learning, and meta-training of learned optimizers.
Acknowledgements

This work was done while on internship at Google. We thank Sergey Ioffe and Niru Maheswaranathan for very helpful discussions and feedback on the paper.

References


