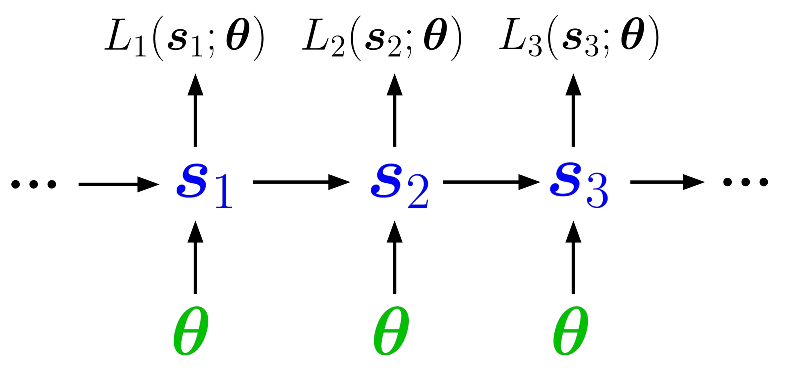
Unbiased Gradient Estimation in Unrolled Computation Graphs with Persistent Evolution Strategies

Paul Vicol[1,2,3], Luke Metz[1], Jascha Sohl-Dickstein[1]
[1]Google Research, [2]University of Toronto, [3]Vector Institute

Motivation & Summary

- Unrolled computation graphs arise in many scenarios
  - Training RNNs, tuning hyperparameters through unrolled optimization, reinforcement learning, training learned optimizers.
- Current approaches to optimizing parameters in such computation graphs suffer from high variance gradients, bias, slow updates, or large memory usage.
- 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.
- PES is unbiased, allowing it to converge to correct solutions that are not found by TBPTT or truncated ES
- Loss surface smoothing induced by PES is beneficial for HO, overcoming erratic meta-loss surfaces

Problem Setup



Task	s_t	θ
RNN	Hidden State	RNN Params
Hyperparameter Optimization	Model Params	Hyperparameters
Learned Optimizers	Model Params	Learned Optimizer Params
RL	Environment State	Policy Params

- Dynamical system with state s_t governed by parameters θ: $s_t = f(s_{t-1}, x_t; \theta)$. We wish to minimize $L(\theta) = \sum_{t=1}^{T} L_t(s_t;\theta)$.
- BPTT and RTRL are expensive and have high latency; TBPTT suffers from truncation bias; approximations to RTRL have higher variance.
- Long unrolls can lead to chaotic or poorly conditioned loss landscapes

Evolution Strategies

- Evolution Strategies (ES) is a method for estimating a descent direction for arbitrary black-box functions using stochastic finite differences.

$$\nabla_\theta \mathbb{E}_{\tilde{\theta}\sim\mathcal{N}(\theta,\sigma^2 I)}\left[L(\tilde{\theta})\right] \approx \hat{g}^{\text{ES}} = \frac{1}{\sigma^2}\mathbb{E}_{\epsilon\sim\mathcal{N}(0,\sigma^2 I)}\left[\epsilon L(\theta+\epsilon)\right]$$

- ES is trivially parallelizable, and thus highly scalable
- ES optimizes a Gaussian-smoothed loss surface
  - Helps overcome pathological structure in long-unroll meta-objectives
- Can optimize arbitrary black-box functions, e.g., non-differentiable objectives like accuracy rather than loss
- However, ES suffers from truncation bias similarly to TBPTT
- **Goal: Can we design an algorithm with the benefits of ES, that does not suffer from truncation bias?**

Persistent Evolution Strategies (PES)

- PES divides the computation graph into a series of truncated unrolls, and performs an ES-based update step after each unroll.
- g^{PES} decomposes into a sum of sequential gradient estimates,

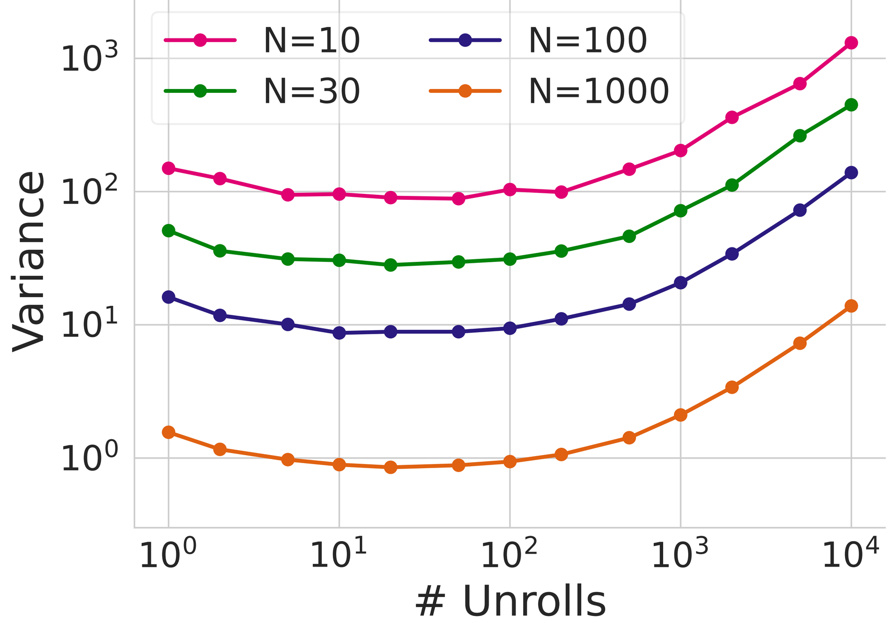
$$\hat{g}^{\text{PES}} = \frac{1}{\sigma^2}\mathbb{E}_\epsilon\left[(\mathsf{I}\otimes \mathbb{1}^\top)\,\mathrm{vec}(\epsilon)\,L(\Theta+\epsilon)\right]$$
$$= \frac{1}{\sigma^2}\mathbb{E}_\epsilon\left[\sum_{t=1}^{T}\xi_t L_t(\theta_1+\epsilon_1,\dots,\theta_t+\epsilon_t)\right]$$

- We obtain unbiased gradient estimates from partial unrolls by:
  1. Not resetting the particles between unrolls
  2. Accumulating perturbations each particle has experienced over all unrolls.
- The PES algorithm (using antithetic sampling) is as follows:

Initialize $s^{(i)} = s_0$ for $i \in \{1,\dots,N\}$
Initialize $\xi^{(i)} \leftarrow 0$ for $i \in \{1,\dots,N\}$
repeat
  $\hat{g}^{\text{PES}} \leftarrow 0$
  for $i = 1,\dots,N$ **do**
    $\epsilon^{(i)} = \begin{cases} \text{draw from } \mathcal{N}(0,\sigma^2 I) & i \text{ odd} \\ -\epsilon^{(i-1)} & i \text{ even}\end{cases}$
    $s^{(i)}, \hat{L}_K^{(i)} \leftarrow \text{unroll}(s^{(i)}, \theta+\epsilon^{(i)}, K)$
    $\xi^{(i)} \leftarrow \xi^{(i)} + \epsilon^{(i)}$
    $\hat{g}^{\text{PES}} \leftarrow \hat{g}^{\text{PES}} + \xi^{(i)}\hat{L}_K^{(i)}$
  end for
  $\hat{g}^{\text{PES}} \leftarrow \frac{1}{N\sigma^2}\hat{g}^{\text{PES}}$

  $\theta \leftarrow \theta - \alpha\hat{g}^{\text{PES}}$

Variance

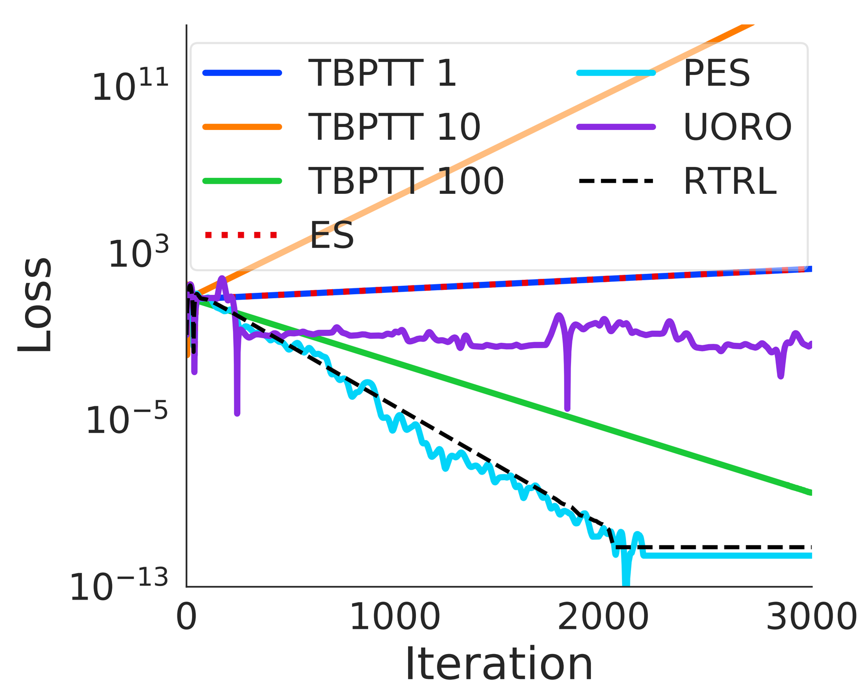


- The variance of the PES gradient estimate depends on the correlation between gradients at each unroll.
- On char-level PTB, we see two regimes as we increase #unrolls: initial decrease in variance, followed by linear increase

Compute and Memory Cost

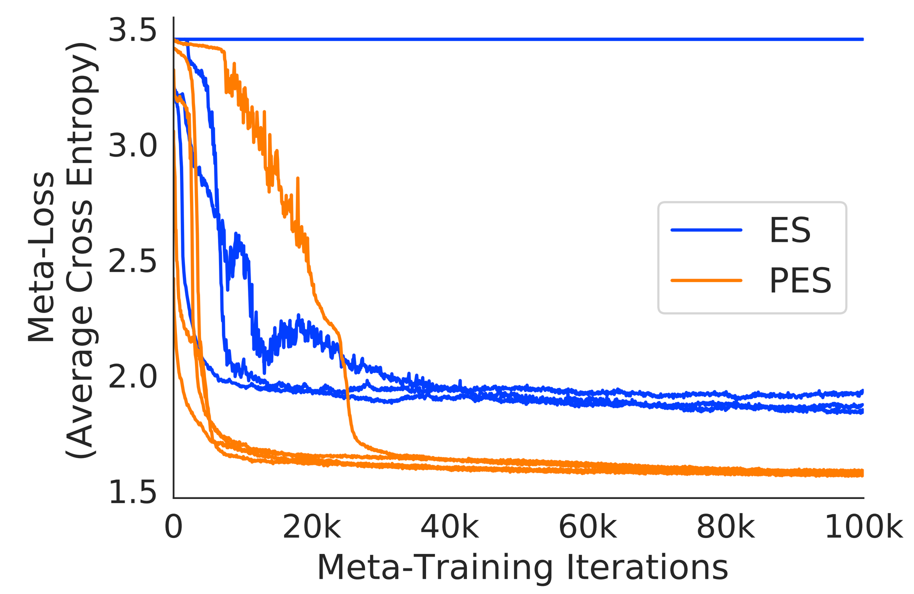
- For each particle, PES uses KF compute, where K is the truncated unroll length and F is the cost of a forward pass
- For each particle, PES stores the state s_t and perturbation accumulator ξ_t.

Influence Balancing



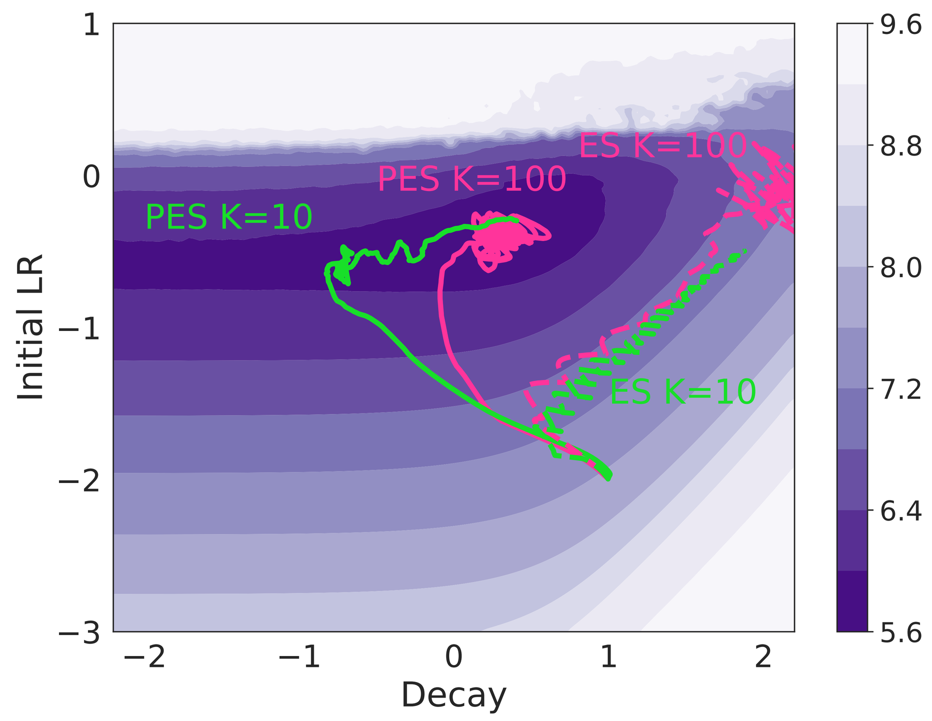
- Synthetic task with arbitrarily long-term dependencies
- Learn a scalar that has a positive short-term influence but a negative long-term influence
- Truncated algorithms fail; PES performs similarly to RTRL given enough particles

Learned Optimizer Meta-Optimization



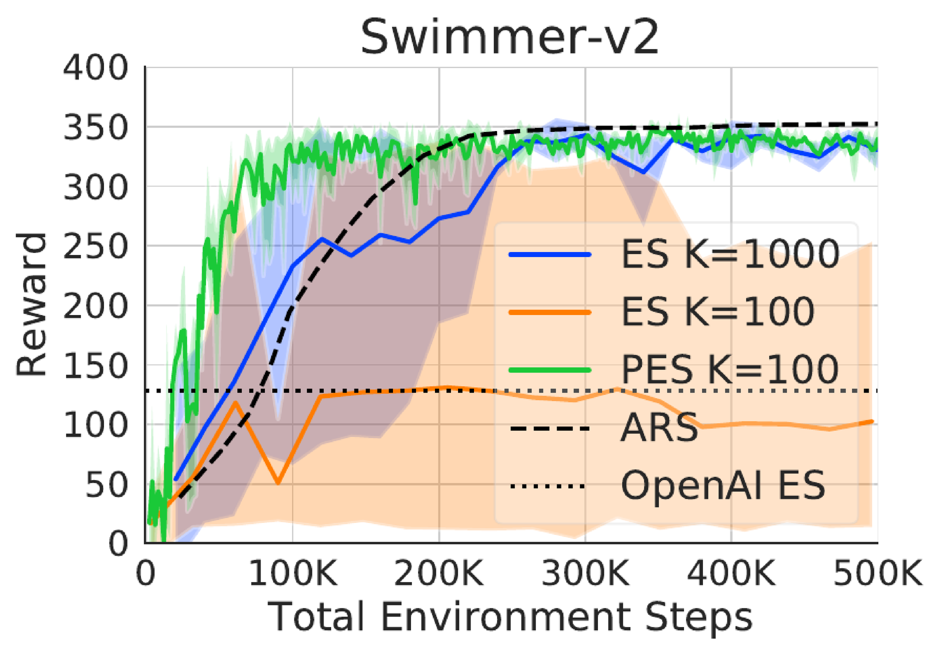
- We meta-train an MLP-based learned optimizer.
- Used to train an MLP on CIFAR-10.
- PES achieves lower losses, and is more consistent across random initializations of the learned optimizer.

Hyperparameter Optimization



- Tuning LR schedule for an MLP on MNIST
- The inner-problem length is $T = 5000$, and we used truncations of length $\{10, 100\}$
- PES can also optimize non-differentiable objectives such as validation accuracy

Learning Policy for Continuous Control



- PES can train a policy for continuous control using partial unrolls
- We found that PES is more efficient than ES applied to full episodes, while truncated ES fails due to bias