Unbiased Gradient Estimation in Unrolled Computation Graphs with Persistent Evolution Strategies

Paul Vicol, Luke Metz, Jascha Sohl-Dickstein

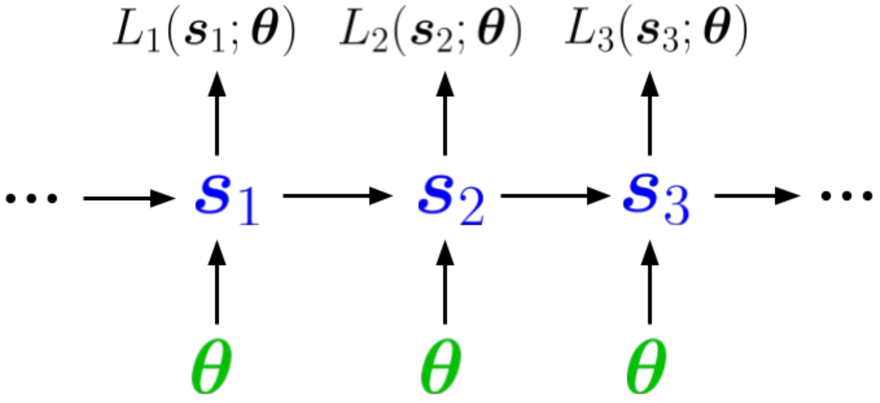


ICML 2021

Google
UNIVERSITY OF TORONTO
VECTOR INSTITUTE

Problem Setup: Unrolled Computation Graphs

- Consider a dynamical system that evolves according to: $\boldsymbol{s}_t = f(\boldsymbol{s}_{t-1}, \boldsymbol{x}_t; \boldsymbol{\theta})$



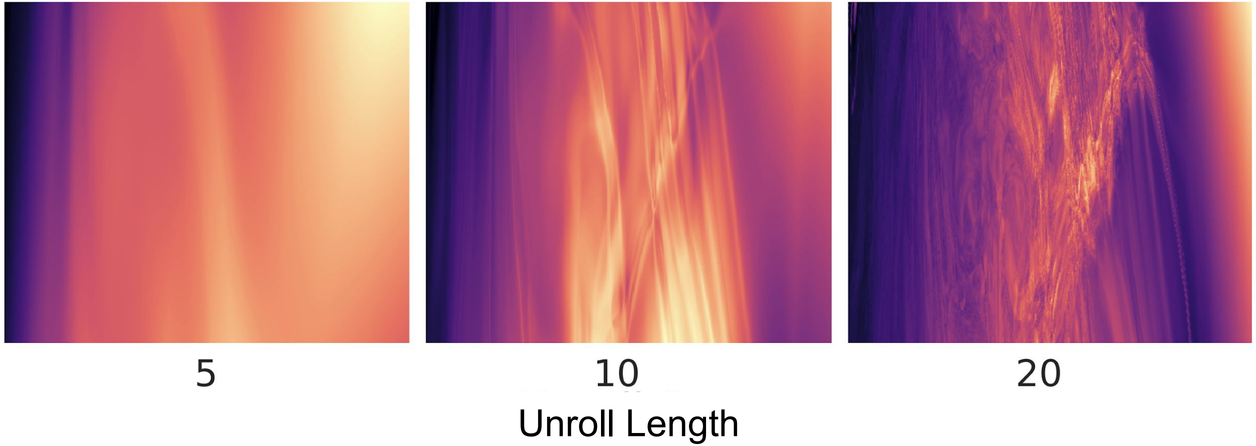
Task	$\boldsymbol{s}_t$	$\boldsymbol{\theta}$
RNN	Hidden State	RNN Params
Hyperparameter Optimization	Model Params	Hyperparameters
Learned Optimizers	Model Params	Learned Optimizer Params
RL	Environment State	Policy Params

Objective: $L(\boldsymbol{\theta}) = \sum_{t=1}^{T} L_t(\boldsymbol{s}_t; \boldsymbol{\theta}_t)$

- **Problem:** Most approaches suffer from *truncation bias, high-variance gradients, slow updates, or high memory usage*

Pathological Meta-Loss Surfaces and ES

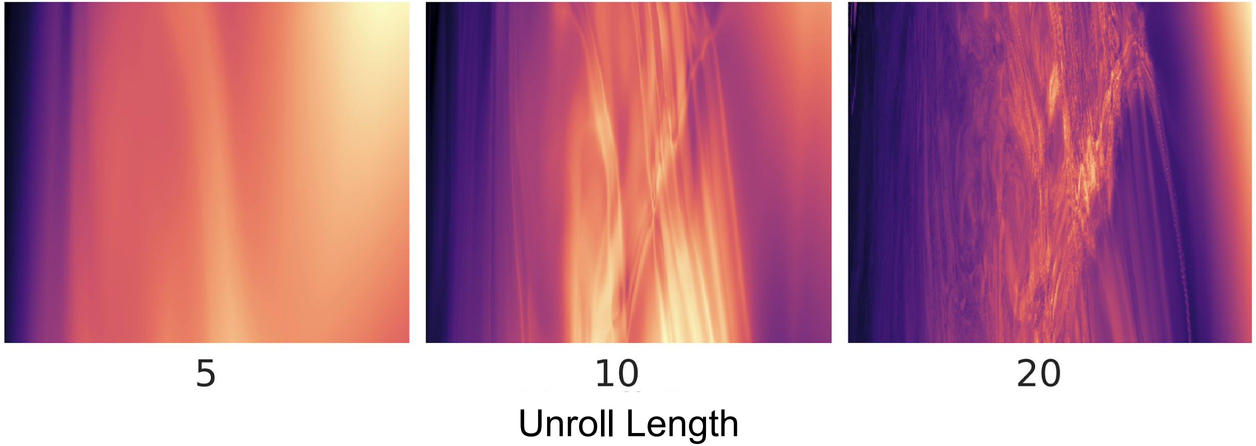
- Another issue: long unrolls can lead to *chaotic or poorly conditioned loss landscapes*
  - This is especially *common for unrolled optimization*



Metz et al., *Understanding and correcting pathologies in the training of learned optimizers.* ICML 2019.

Pathological Meta-Loss Surfaces and ES

- Another issue: long unrolls can lead to *chaotic or poorly conditioned loss landscapes*
  - This is especially *common for unrolled optimization*



Metz et al., *Understanding and correcting pathologies in the training of learned optimizers.* ICML 2019.

- Consider optimizing a *Gaussian-smoothed meta-objective* $\mathbb{E}_{\tilde{\boldsymbol{\theta}} \sim \mathcal{N}(\boldsymbol{\theta}, \sigma^2 I)}[L(\tilde{\boldsymbol{\theta}})]$
- *Evolution strategies (ES)* is a method for estimating a descent direction using stochastic finite differences:

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

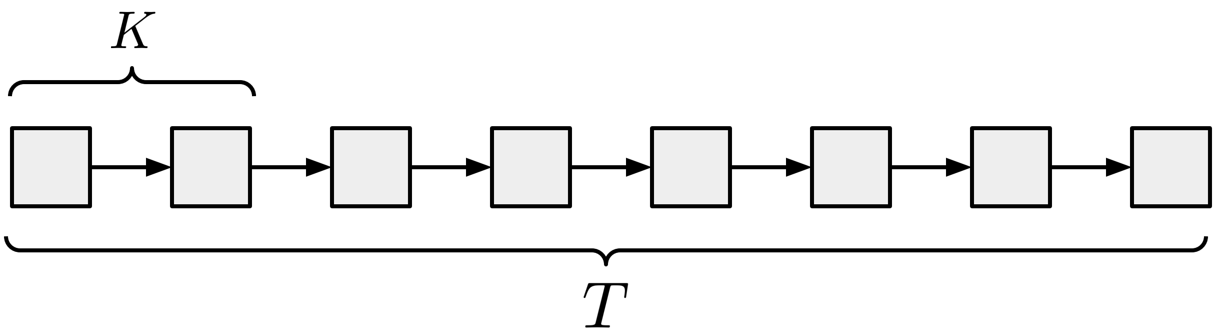
Pros & Cons of ES

- ES optimizes a *Gaussian-smoothed loss surface*
- Does not use backprop, so does not require storing states in memory
- Can optimize *arbitrary black-box functions*, e.g., non-differentiable objectives like accuracy rather than loss
- Is *highly scalable on parallel compute*, and can have low variance with antithetic sampling

- In principle, using ES on *full unrolls* of the computation graph would work well
  - **Problem:** we have to do a full unroll for each parameter update, which is slow
- In practice, ES is applied to truncated unrolls
  - **Problem:** Suffers from *truncation bias* similarly to TBPTT

PES High-Level Overview

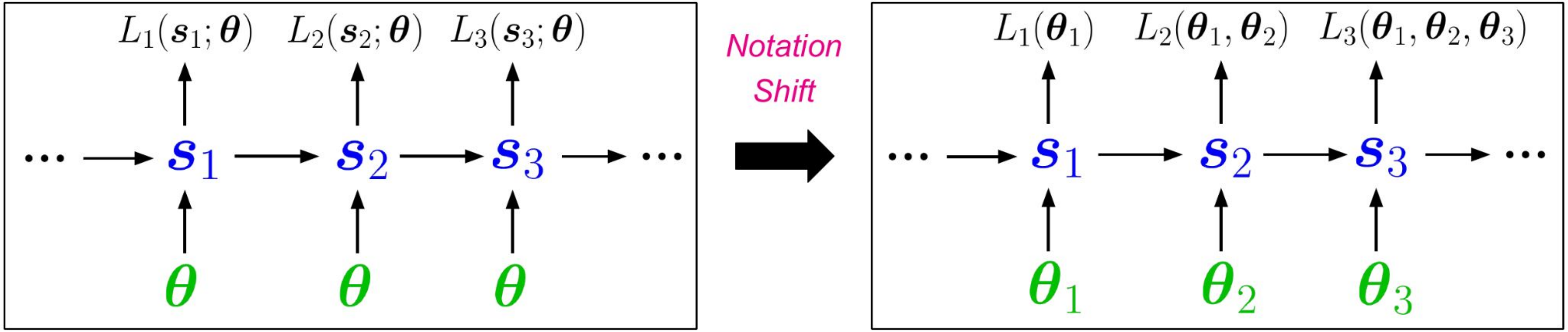
PES splits the computation graph into a *series of truncated unrolls*
Performs an ES-style parameter *update after each unroll*



Eliminates bias from the truncations by accumulating correction terms over the full sequence of unrolls

- Allows for *rapid parameter updates*
- Inherits useful properties from ES:
  - Has low memory usage, does not require storing states for backprop
  - *Smooths the loss surface*, which is useful for unrolled computations

PES Derivation: Notation Shift



Unrolled computation graphs depend on shared parameters $\boldsymbol{\theta}$ at every timestep

In order to account for how the applications of $\boldsymbol{\theta}$ contribute to the overall gradient, $\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta})$ we use subscripts to distinguish between applications of $\boldsymbol{\theta}$ at different steps, $\forall t : \boldsymbol{\theta}_t = \boldsymbol{\theta}$

We drop the dependence on s_t and explicitly include the dependence on each $\boldsymbol{\theta}_t$

We also define $\Theta = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_T)^\top$

Then we can write $L_t(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_t) = L_t(\Theta)$

PES Derivation 1

$$\frac{dL(\Theta)}{d\boldsymbol{\theta}} = \sum_{\tau=1}^{T} \frac{\partial L(\Theta)}{\partial \boldsymbol{\theta}_\tau} = (\mathbf{I} \otimes \mathbf{1}^\top) \frac{\partial L(\Theta)}{\partial \operatorname{vec}(\Theta)}$$

PES Derivation 1

$$\frac{dL(\Theta)}{d\boldsymbol{\theta}} = \sum_{\tau=1}^{T} \frac{\partial L(\Theta)}{\partial \boldsymbol{\theta}_\tau} = (\mathbf{I} \otimes \mathbf{1}^\top) \frac{\partial L(\Theta)}{\partial \operatorname{vec}(\Theta)} \approx \boldsymbol{g}^{\text{PES}} = \left(\mathbf{I} \otimes \mathbf{1}^\top\right) \mathbb{E}_{\boldsymbol{\epsilon}}\left[\frac{1}{\sigma^2} \operatorname{vec}(\boldsymbol{\epsilon})\, L(\Theta + \boldsymbol{\epsilon})\right]$$

Apply ES

PES Derivation 1

$$\frac{dL(\Theta)}{d\boldsymbol{\theta}} = \sum_{\tau=1}^{T} \frac{\partial L(\Theta)}{\partial \boldsymbol{\theta}_\tau} = (\mathbf{I} \otimes \mathbf{1}^\top) \frac{\partial L(\Theta)}{\partial \operatorname{vec}(\Theta)} \approx \boldsymbol{g}^{\text{PES}} = \left(\mathbf{I} \otimes \mathbf{1}^\top\right) \mathbb{E}_{\boldsymbol{\epsilon}} \left[\frac{1}{\sigma^2} \operatorname{vec}(\boldsymbol{\epsilon})\, L(\Theta + \boldsymbol{\epsilon}) \right]$$

Apply ES

Example with *2D theta and T=3 steps:*

$$\Theta = \begin{bmatrix} \text{—} \boldsymbol{\theta}_1^\top \text{—} \\ \text{—} \boldsymbol{\theta}_2^\top \text{—} \\ \text{—} \boldsymbol{\theta}_3^\top \text{—} \end{bmatrix} = \begin{bmatrix} \theta_1^{(1)} & \theta_1^{(2)} \\ \theta_2^{(1)} & \theta_2^{(2)} \\ \theta_3^{(1)} & \theta_3^{(2)} \end{bmatrix}$$

PES Derivation 1

$$\frac{dL(\Theta)}{d\boldsymbol{\theta}} = \sum_{\tau=1}^{T} \frac{\partial L(\Theta)}{\partial \boldsymbol{\theta}_\tau} = \left(\mathbf{I} \otimes \mathbf{1}^\top\right) \frac{\partial L(\Theta)}{\partial \operatorname{vec}(\Theta)} \approx \boldsymbol{g}^{\text{PES}} = \left(\mathbf{I} \otimes \mathbf{1}^\top\right) \mathbb{E}_{\boldsymbol{\epsilon}} \left[\frac{1}{\sigma^2} \operatorname{vec}(\boldsymbol{\epsilon})\, L(\Theta + \boldsymbol{\epsilon}) \right]$$

Apply ES

Example with *2D theta and T=3 steps:*

$$\Theta = \begin{bmatrix} \text{---}\ \boldsymbol{\theta}_1^\top\ \text{---} \\ \text{---}\ \boldsymbol{\theta}_2^\top\ \text{---} \\ \text{---}\ \boldsymbol{\theta}_3^\top\ \text{---} \end{bmatrix} = \begin{bmatrix} \theta_1^{(1)} & \theta_1^{(2)} \\ \theta_2^{(1)} & \theta_2^{(2)} \\ \theta_3^{(1)} & \theta_3^{(2)} \end{bmatrix}$$

$$\operatorname{vec}(\Theta) = \begin{bmatrix} \theta_1^{(1)} & \theta_2^{(1)} & \theta_3^{(1)} & \theta_1^{(2)} & \theta_2^{(2)} & \theta_3^{(2)} \end{bmatrix}^\top$$

$$\frac{\partial L(\Theta)}{\partial \operatorname{vec}(\Theta)} = \begin{bmatrix} \frac{\partial L(\Theta)}{\partial \theta_1^{(1)}} & \frac{\partial L(\Theta)}{\partial \theta_2^{(1)}} & \frac{\partial L(\Theta)}{\partial \theta_3^{(1)}} & \frac{\partial L(\Theta)}{\partial \theta_1^{(2)}} & \frac{\partial L(\Theta)}{\partial \theta_2^{(2)}} & \frac{\partial L(\Theta)}{\partial \theta_3^{(2)}} \end{bmatrix}^\top$$

$$\mathbf{I} \otimes \mathbf{1}^\top = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

PES Derivation (1)

$$\frac{dL(\Theta)}{d\boldsymbol{\theta}} = \sum_{\tau=1}^{T} \frac{\partial L(\Theta)}{\partial \boldsymbol{\theta}_\tau} = \underbrace{\left(\mathbf{I} \otimes \mathbf{1}^\top\right) \frac{\partial L(\Theta)}{\partial \operatorname{vec}(\Theta)}} \approx \boldsymbol{g}^{\text{PES}} = \left(\mathbf{I} \otimes \mathbf{1}^\top\right) \boxed{\mathbb{E}_{\boldsymbol{\epsilon}}\left[\frac{1}{\sigma^2} \operatorname{vec}(\boldsymbol{\epsilon}) L(\Theta + \boldsymbol{\epsilon})\right]}$$

Apply ES

Example with *2D theta and T=3 steps:*

$$\Theta = \begin{bmatrix} \text{---} \boldsymbol{\theta}_1^\top \text{---} \\ \text{---} \boldsymbol{\theta}_2^\top \text{---} \\ \text{---} \boldsymbol{\theta}_3^\top \text{---} \end{bmatrix} = \begin{bmatrix} \theta_1^{(1)} & \theta_1^{(2)} \\ \theta_2^{(1)} & \theta_2^{(2)} \\ \theta_3^{(1)} & \theta_3^{(2)} \end{bmatrix}$$

$$\operatorname{vec}(\Theta) = \begin{bmatrix} \theta_1^{(1)} & \theta_2^{(1)} & \theta_3^{(1)} & \theta_1^{(2)} & \theta_2^{(2)} & \theta_3^{(2)} \end{bmatrix}^\top$$

$$\frac{\partial L(\Theta)}{\partial \operatorname{vec}(\Theta)} = \begin{bmatrix} \frac{\partial L(\Theta)}{\partial \theta_1^{(1)}} & \frac{\partial L(\Theta)}{\partial \theta_2^{(1)}} & \frac{\partial L(\Theta)}{\partial \theta_3^{(1)}} & \frac{\partial L(\Theta)}{\partial \theta_1^{(2)}} & \frac{\partial L(\Theta)}{\partial \theta_2^{(2)}} & \frac{\partial L(\Theta)}{\partial \theta_3^{(2)}} \end{bmatrix}^\top$$

$$\mathbf{I} \otimes \mathbf{1}^\top = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

$$\left(\mathbf{I} \otimes \mathbf{1}^\top\right) \frac{\partial L(\Theta)}{\partial \operatorname{vec}(\Theta)} = \begin{bmatrix} \frac{\partial L(\Theta)}{\partial \theta_1^{(1)}} + \frac{\partial L(\Theta)}{\partial \theta_2^{(1)}} + \frac{\partial L(\Theta)}{\partial \theta_3^{(1)}} \\ \frac{\partial L(\Theta)}{\partial \theta_1^{(2)}} + \frac{\partial L(\Theta)}{\partial \theta_2^{(2)}} + \frac{\partial L(\Theta)}{\partial \theta_3^{(2)}} \end{bmatrix} = \underbrace{\begin{bmatrix} \frac{\partial L(\Theta)}{\partial \theta_1^{(1)}} \\ \frac{\partial L(\Theta)}{\partial \theta_1^{(2)}} \end{bmatrix}}_{\frac{\partial L(\Theta)}{\partial \boldsymbol{\theta}_1}} + \underbrace{\begin{bmatrix} \frac{\partial L(\Theta)}{\partial \theta_2^{(1)}} \\ \frac{\partial L(\Theta)}{\partial \theta_2^{(2)}} \end{bmatrix}}_{\frac{\partial L(\Theta)}{\partial \boldsymbol{\theta}_2}} + \underbrace{\begin{bmatrix} \frac{\partial L(\Theta)}{\partial \theta_3^{(1)}} \\ \frac{\partial L(\Theta)}{\partial \theta_3^{(2)}} \end{bmatrix}}_{\frac{\partial L(\Theta)}{\partial \boldsymbol{\theta}_3}} = \sum_{\tau=1}^{T} \frac{\partial L(\Theta)}{\partial \boldsymbol{\theta}_\tau} = \frac{dL(\Theta)}{d\boldsymbol{\theta}}$$

PES Derivation 2

- PES decomposes into a *sum of sequential gradient estimates*.
  - Below, $\boldsymbol{\epsilon}$ is a matrix whose rows are per-timestep perturbations $\boldsymbol{\epsilon}_\tau$

$$
\begin{aligned}
\hat{\boldsymbol{g}}^{\mathrm{PES}} &= \frac{1}{\sigma^2}\mathbb{E}_{\boldsymbol{\epsilon}}\left[\left(\mathbf{I}\otimes\mathbf{1}^\top\right)\mathrm{vec}\left(\boldsymbol{\epsilon}\right)L(\Theta+\boldsymbol{\epsilon})\right] \\
&= \frac{1}{\sigma^2}\mathbb{E}_{\boldsymbol{\epsilon}}\left[\left(\sum_{\tau=1}^{T}\boldsymbol{\epsilon}_\tau\right)\sum_{t=1}^{T}L_t(\Theta+\boldsymbol{\epsilon})\right] \\
&= \frac{1}{\sigma^2}\mathbb{E}_{\boldsymbol{\epsilon}}\left[\sum_{t=1}^{T}\boldsymbol{\xi}_t L_t(\boldsymbol{\theta}_1+\boldsymbol{\epsilon}_1,\dots,\boldsymbol{\theta}_t+\boldsymbol{\epsilon}_t)\right]
\end{aligned}
$$

- We obtain unbiased gradient estimates from partial unrolls by: 1) *not resetting the particles between unrolls*, and 2) *accumulating the perturbations each particle has experienced over multiple unrolls*

PES Derivation 2

- PES decomposes into a *sum of sequential gradient estimates*.
  - Below, $\boldsymbol{\epsilon}$ is a matrix whose rows are per-timestep perturbations $\boldsymbol{\epsilon}_\tau$

$$\hat{\boldsymbol{g}}^{\text{PES}} = \frac{1}{\sigma^2}\mathbb{E}_{\boldsymbol{\epsilon}}\left[(\mathbf{I}\otimes\mathbf{1}^\top)\,\text{vec}\,(\boldsymbol{\epsilon})\,L(\Theta+\boldsymbol{\epsilon})\right]$$

$$= \frac{1}{\sigma^2}\mathbb{E}_{\boldsymbol{\epsilon}}\left[\left(\sum_{\tau=1}^{T}\boldsymbol{\epsilon}_\tau\right)\sum_{t=1}^{T}L_t(\Theta+\boldsymbol{\epsilon})\right]$$

$$= \frac{1}{\sigma^2}\mathbb{E}_{\boldsymbol{\epsilon}}\left[\sum_{t=1}^{T}\boldsymbol{\xi}_t L_t(\boldsymbol{\theta}_1+\boldsymbol{\epsilon}_1,\dots,\boldsymbol{\theta}_t+\boldsymbol{\epsilon}_t)\right]$$

- We obtain unbiased gradient estimates from partial unrolls by: 1) *not resetting the particles between unrolls*, and 2) *accumulating the perturbations each particle has experienced over multiple unrolls*

Monte Carlo PES Estimate

$$\hat{g}^{\text{PES}} = \frac{1}{\sigma^2 N}\sum_{i=1}^{N}\sum_{t=1}^{T}\boldsymbol{\xi}_t^{(i)}L_t(\boldsymbol{\theta}_1+\boldsymbol{\epsilon}_1^{(i)},\dots,\boldsymbol{\theta}_t+\boldsymbol{\epsilon}_t^{(i)})$$

PES Estimate w/ Antithetic Sampling

$$\hat{g}^{\text{PES-A}} = \frac{1}{2N\sigma^2}\sum_{i=1}^{N}\sum_{t=1}^{T}\left(\boldsymbol{\xi}_t^{(i)}L_t(\boldsymbol{\theta}_1+\boldsymbol{\epsilon}_1^{(i)},\dots,\boldsymbol{\theta}_t+\boldsymbol{\epsilon}_t^{(i)}) - \boldsymbol{\xi}_t^{(i)}L_t(\boldsymbol{\theta}_1-\boldsymbol{\epsilon}_1^{(i)},\dots,\boldsymbol{\theta}_t-\boldsymbol{\epsilon}_t^{(i)})\right)$$

ES & PES Algorithms

Algorithm 1 Truncated Evolution Strategies (ES) applied to partial unrolls of a computation graph.

Input: $\boldsymbol{s}_0$, initial state
K, truncation length for partial unrolls
N, number of particles
σ, standard deviation of perturbations
α, learning rate for ES optimization

Initialize $\boldsymbol{s} = \boldsymbol{s}_0$

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

Algorithm 2 Persistent evolution strategies (PES). Differences from ES are highlighted in purple.

Input: $\boldsymbol{s}_0$, initial state
K, truncation length for partial unrolls
N, number of particles
σ, standard deviation of perturbations
α, learning rate for PES optimization

Initialize $\boldsymbol{s}^{(i)} = \boldsymbol{s}_0$ for $i \in \{1, \ldots, N\}$
Initialize $\boldsymbol{\xi}^{(i)} \leftarrow \mathbf{0}$ for $i \in \{1, \ldots, N\}$

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

Example Implementation in JAX

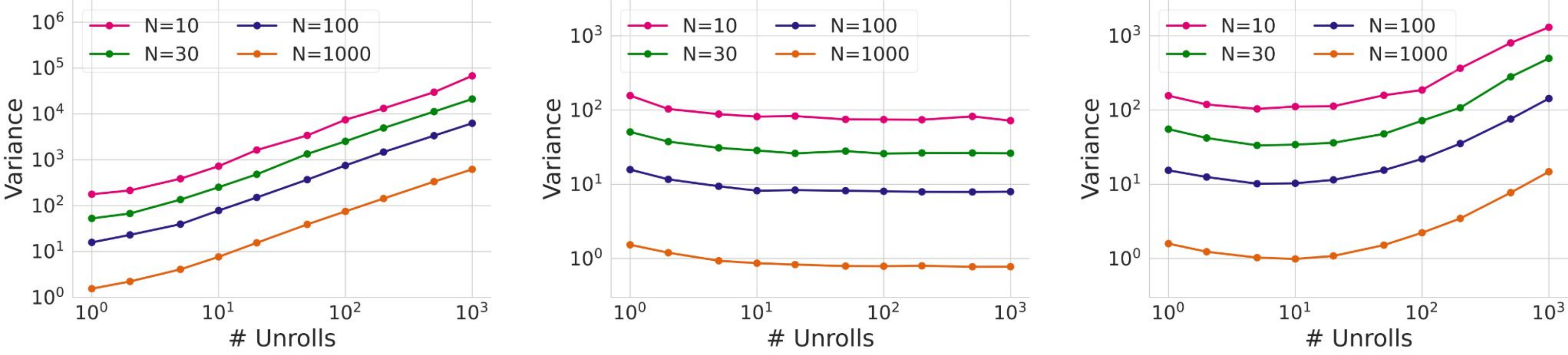
```
def pes_grad(key, xs, pert_accum, theta, t0, T, K, sigma, N):
    # Generate antithetic perturbations
    pos_perts = jax.random.normal(key, (N//2, theta.shape[0])) * sigma  # Antithetic positives
    neg_perts = -pos_perts  # Antithetic negatives
    perts = jnp.concatenate([pos_perts, neg_perts], axis=0)

    # Unroll the inner problem for K steps using the antithetic perturbations of theta
    L, xs = jax.vmap(unroll, in_axes=(0,0,None,None,None))(xs, theta + perts, t0, T, K)
    # Add the perturbations from this unroll to the perturbation accumulators
    pert_accum = pert_accum + perts
    # Compute the PES gradient estimate
    theta_grad = jnp.mean(pert_accum * L.reshape(-1, 1) / (sigma**2), axis=0)
    return theta_grad, xs, pert_accum
```

PES Variance

- The variance of the PES gradient estimate depends on the correlation between gradients at each unroll

1. *If we assume that the gradients for each unroll are i.i.d., then variance scales linearly in the number of unrolls*
2. *If we assume that the gradients from each unroll are identical, then variance scales as O(cons + cons/#unrolls)*
3. *Real data exhibits characteristics of both synthetic scenarios*



(a) Random sequence (b) Single character repeated (c) Real PTB sequence

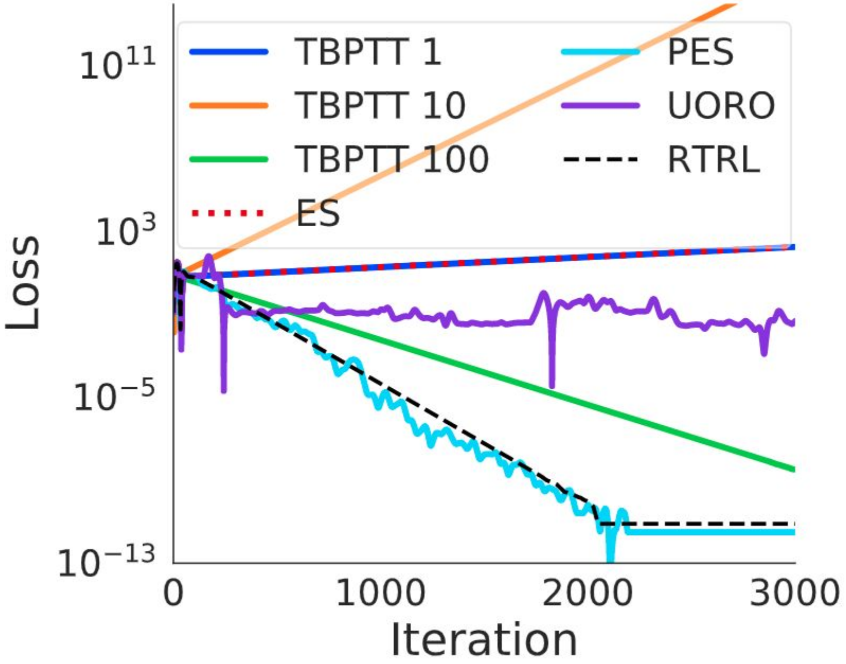
PES Experiments

- Our experiments aim to demonstrate that:
  1. PES is unbiased, allowing it to converge to correct solutions that are not found by TBPTT or truncated ES
  2. *Loss surface smoothing induced by PES is beneficial for meta-optimization*, overcoming erratic meta-loss surfaces
  3. PES can target *non-differentiable objectives* such as validation accuracy

- We apply PES to several *illustrative scenarios*:
  1. Optimizing hyperparameters
  2. Meta-training a learned optimizer
  3. Learning a policy for continuous control

Experiments: Influence Balancing

- Synthetic task introduced by Tallec et al. (2017), designed to have *arbitrarily long-term dependencies*
- Learn a scalar parameter that has a *positive influence in the short term* but a *negative influence in the long term*
- *Truncated algorithms like TBPTT fail* as the parameter explodes in the wrong direction
- *PES performs nearly identically to exact RTRL* given sufficiently many particles to reduce variance

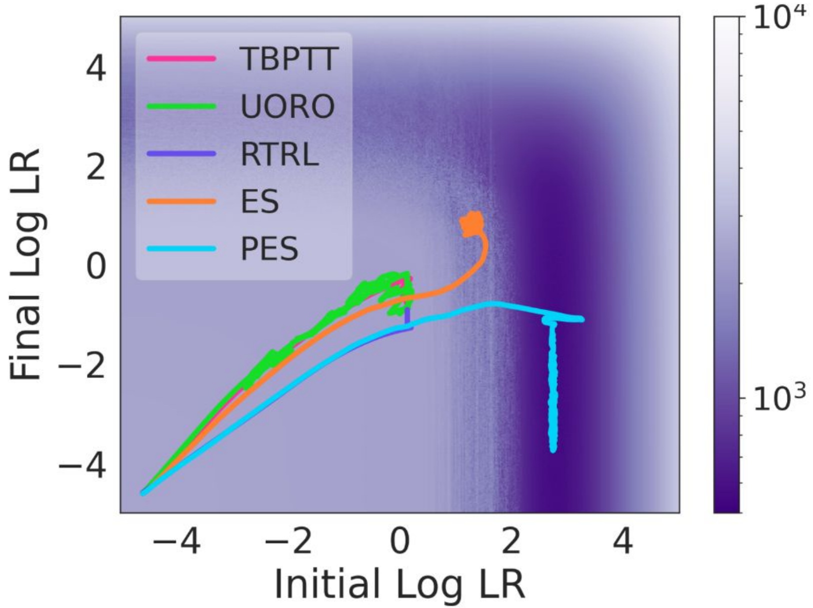


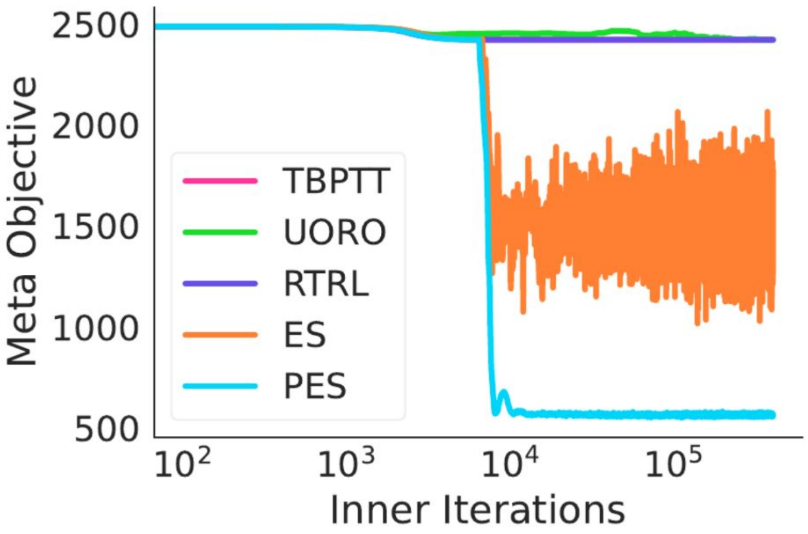
* Note that this is intended to show that PES is unbiased; it is not a compute-time comparison

Experiments: LR Optimization for 2D Regression

- We defined a toy 2D regression problem that has one global minimum, but many *local minima to which truncated gradient methods could converge*
- We learn a linearly-decaying LR schedule parameterized by: $\alpha_t = (1 - \frac{t}{T})e^{\theta_0} + \frac{t}{T}e^{\theta_1}$

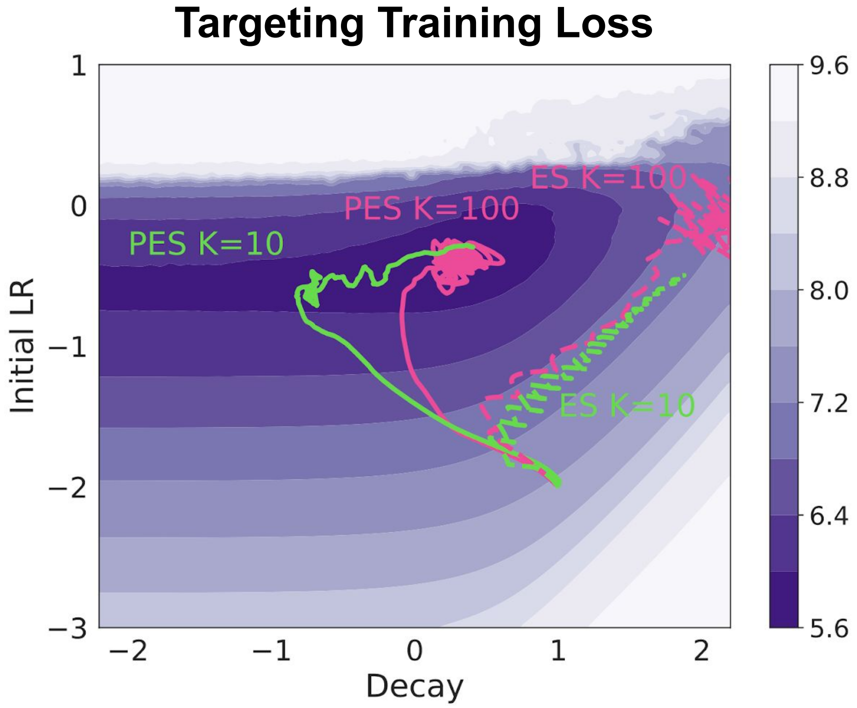
Meta-Loss Surface

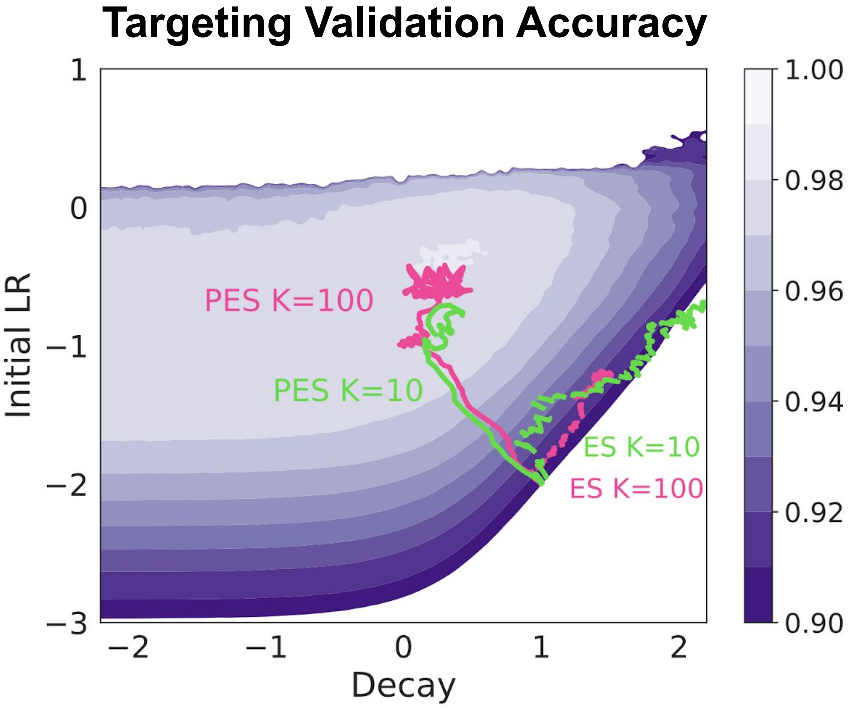




Experiments: MNIST Learning Rate Schedule

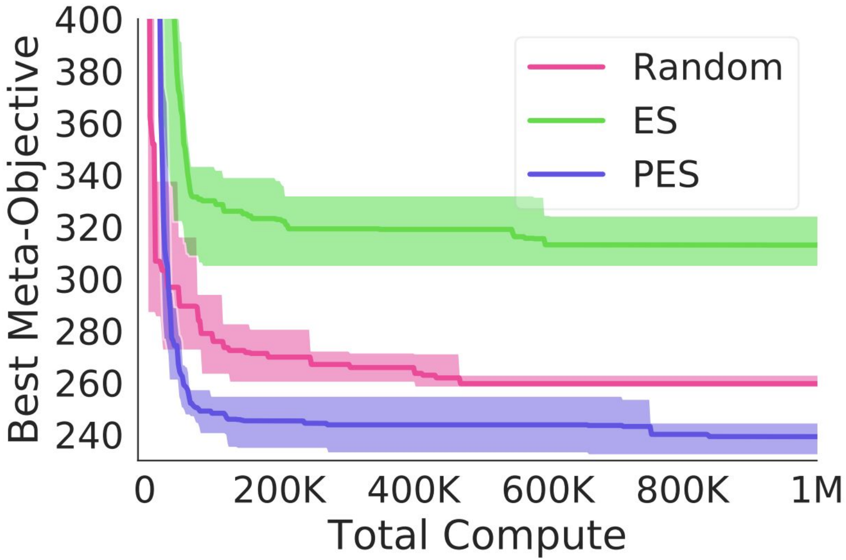
- *Meta-learning a learning rate schedule* for an MLP (784-100-100-10) on MNIST
- Here, the full inner-problem length is T=5000, and we run ES and PES with truncation lengths $K \in \{10, 100\}$





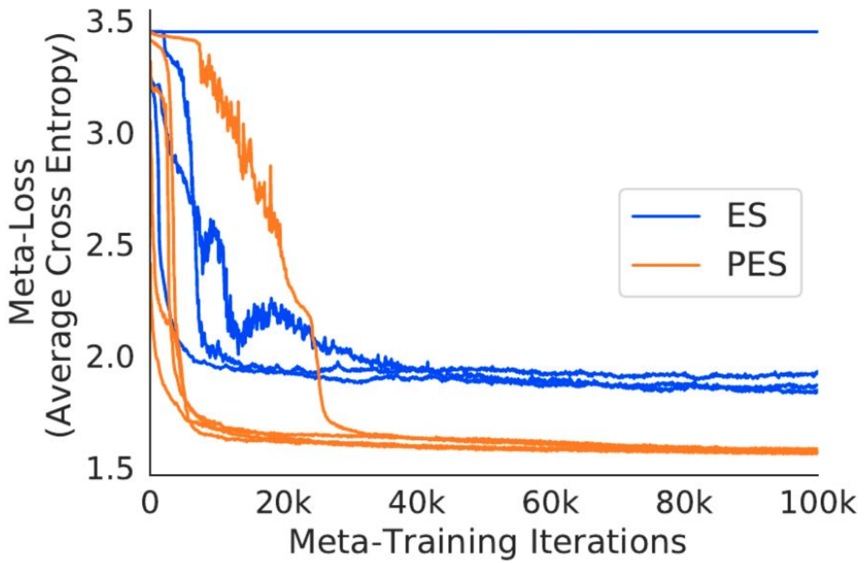
Experiments: Tuning Many Hyperparameters

- 4-layer MLP trained on MNIST, total inner problem length T=1000
- Tuning *separate LR and momentum for each parameter block* (weight matrix and bias vector)
  - *20 hyperparameters total*
- Random search, truncated ES, and PES are run w/ four diff. random seeds
- ES performs poorly compared to RS, because it does not move in the correct direction



Training Learned Optimizers

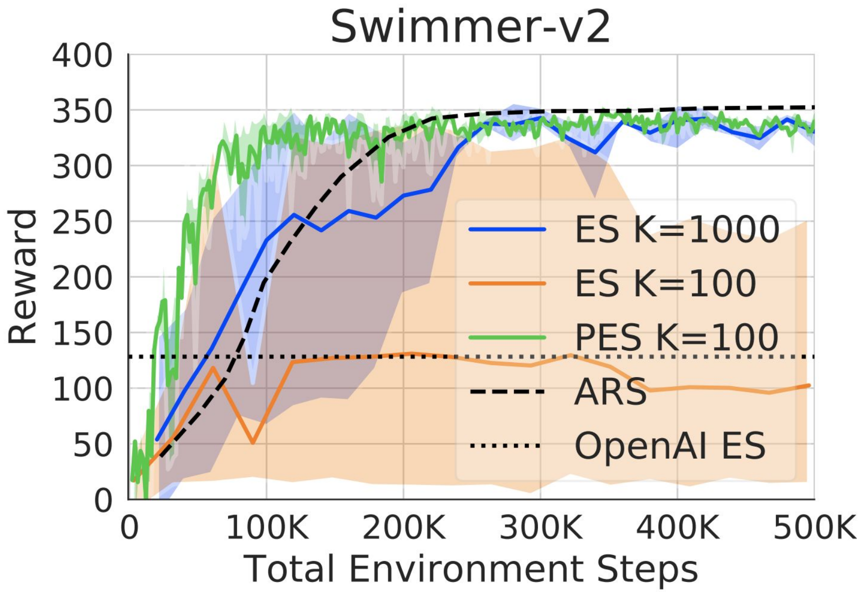
- We *meta-train an MLP-based learned optimizer* as described in Metz et al. (2019)
- This optimizer is used to train a *two hidden-layer, 128 unit, MLP on CIFAR-10*
- Due to PES's unbiased nature, *PES achieves both lower losses, and is more consistent across random initializations of the learned optimizer*



Learning a Policy for Continuous Control

- PES can be used to *train a policy for a continuous control problem using partial unrolls*

- We found that *PES is more efficient than ES applied to full episodes*, while truncated ES fails due to bias



Conclusion

- Algorithmically, PES is an *easy-to-implement modification of ES*
- *Provides unbiased gradient estimates from partial unrolls*
- *Inherits useful characteristics* from ES:
  - Parallelizability
  - Works with arbitrary non-differentiable functions
  - Smooths the meta-loss surface
- PES has *tractable compute and memory cost*
- Can be applied to various unrolled problems (hyperopt, learned optimizers, RL)

Thank you!