Learning to Learn by Zeroth-Order Oracle

Yangjun Ruan[1], Yuanhao Xiong[2],
Sashank Reddi[3], Sanjiv Kumar[3], Cho-Jui Hsieh[2,3]

[1] Zhejiang University, [2] UCLA, [3] Google Research

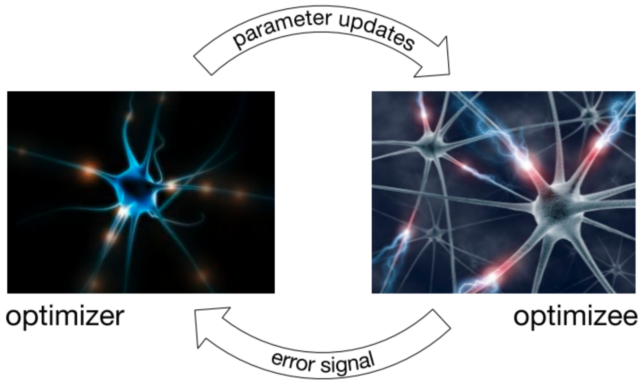
Learning to learn (L2L)

- Use neural networks to automatically learn optimization algorithms

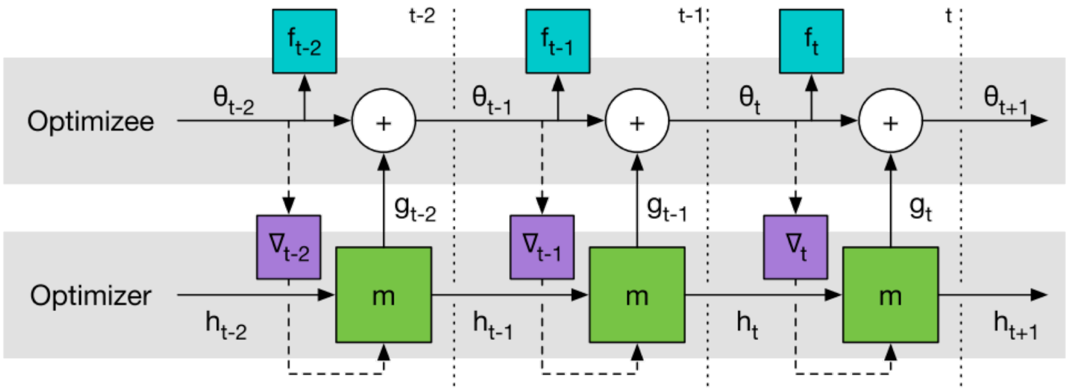
$$\theta_{t+1} = \theta_t + g_t(\nabla f(\theta_t), \varphi)$$

  - f: the optimizee (optimization problems) specified by its parameters θ
  - g: the learned optimizer specified by its parameters φ

- The optimizer g is usually modeled as recurrent neural networks (RNNs)



(a) Overview



(b) Computational graph for training the optimizer

Figure: Andrychowicz et al., 2016

Learning to learn (L2L)

✓ Improve hand-designed algorithms with learned optimization rules

Learning to learn (L2L)

✓ Improve hand-designed algorithms with learned optimization rules

✗ Gradient-based: cannot be applied when gradients are difficult or infeasible to obtain (i.e., zeroth-order optimization)

Zeroth-order (ZO) optimization

- Setting: explicit gradients are not available
- Widely used application: black-box adversarial attacks

Zeroth-order (ZO) optimization

- Setting: explicit gradients are not available
- Widely used application: black-box adversarial attacks
- Basic method: approximate gradients along Gaussian sampled query directions

$$\hat{\nabla} f(\theta) = \frac{1}{q} \sum_{i=1}^{q} \frac{f(\theta + \mu u_i) - f(\theta)}{\mu} u_i$$

  - $\{u_i\}$: query directions sampled from standard Gaussian distribution
  - q: number of query directions
  - μ: smoothing parameter

Zeroth-order (ZO) optimization

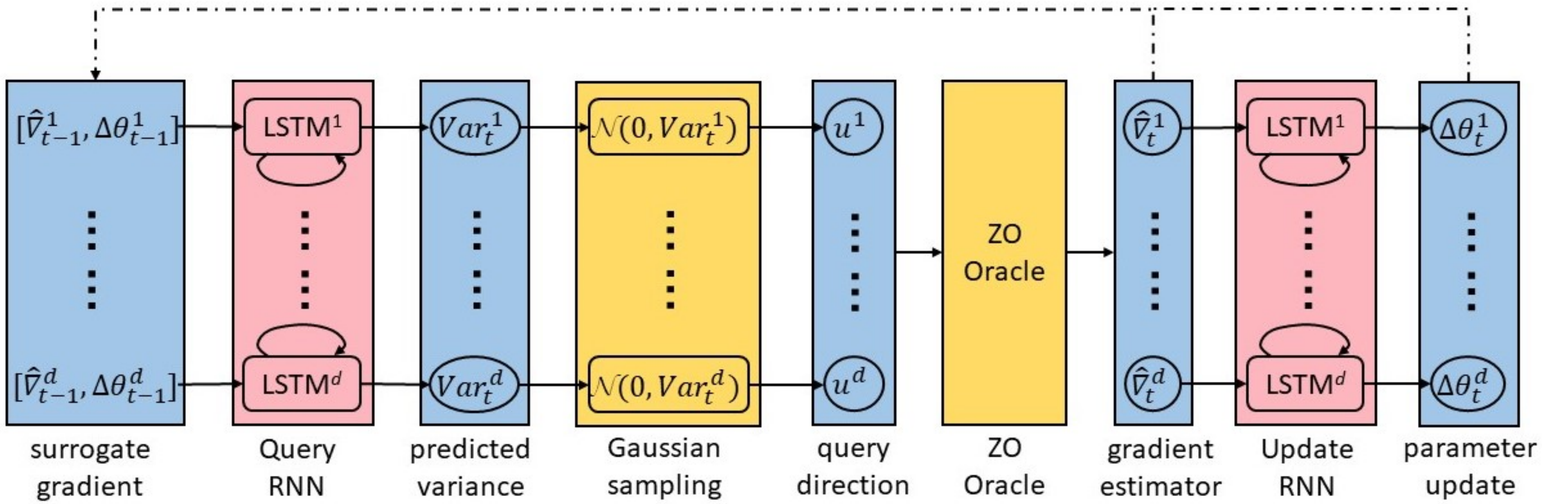
- Existing ZO algorithms: suffer from the high variance of ZO gradient estimator
  - Mainly results from random query directions
  - Hamper convergence: usually d (parameter size) times slower than its first-order counterpart

Zeroth-order (ZO) optimization

- Existing ZO algorithms: suffer from the high variance of ZO gradient estimator
  - Mainly results from random query directions
  - Hamper convergence: usually d (parameter size) times slower than its first-order counterpart

- Our work: apply the L2L framework to learn an efficient ZO optimizer

Method



$[\hat{\nabla}^1_{t-1}, \Delta\theta^1_{t-1}]$
$[\hat{\nabla}^d_{t-1}, \Delta\theta^d_{t-1}]$
LSTM¹
LSTMᵈ
Var_t^1
Var_t^d
$\mathcal{N}(0, Var_t^1)$
$\mathcal{N}(0, Var_t^d)$
u^1
u^d
ZO Oracle
$\hat{\nabla}_t^1$
$\hat{\nabla}_t^d$
$\Delta\theta_t^1$
$\Delta\theta_t^d$
surrogate gradient
Query RNN
predicted variance
Gaussian sampling
query direction
ZO Oracle
gradient estimator
Update RNN
parameter update

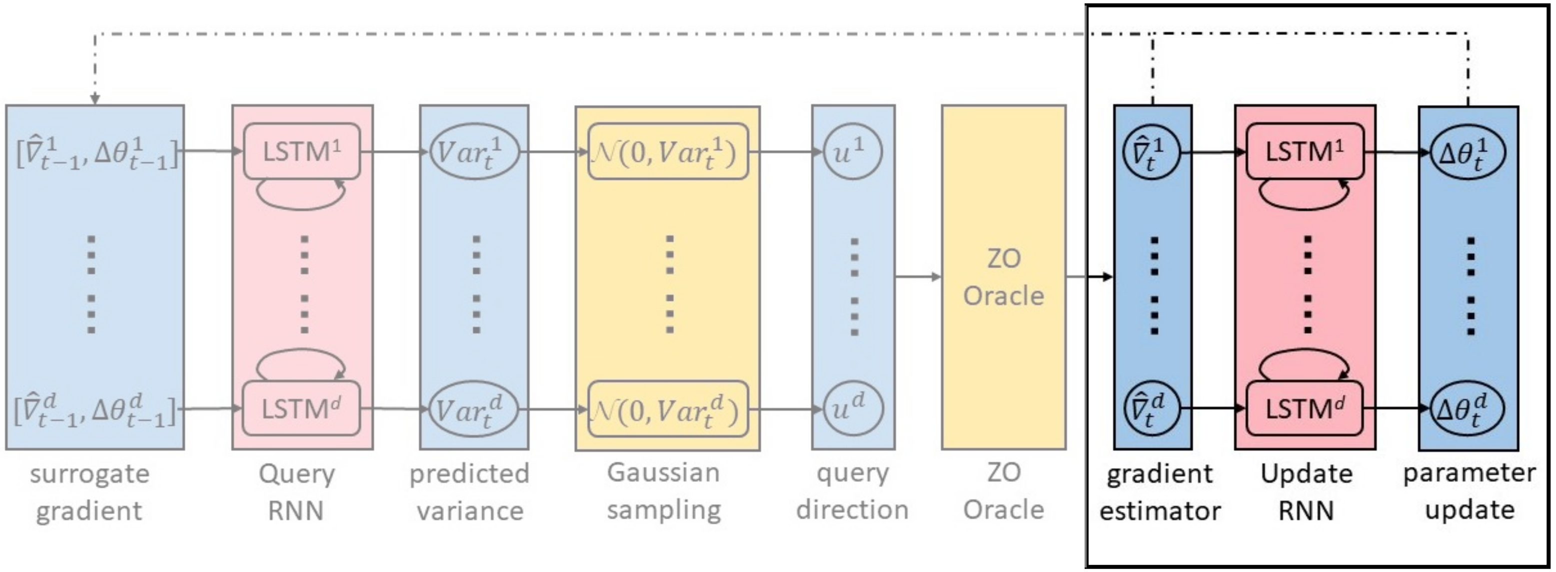
Method

- Jointly learn the parameter update rule and the Gaussian sampling rule
  - UpdateRNN: learn how to propose parameter updates given approximated gradients

$$\theta_t = \theta_{t-1} + \text{UpdateRNN}\left(\hat{\nabla} f(\theta_t)\right)$$

  - QueryRNN: learn to identify the important sampling subspace and adaptively modify the search distribution

$$\Sigma_t = \text{QueryRNN}\left([\hat{\nabla} f(\theta_{t-1}), \Delta\theta_{t-1}]\right)$$



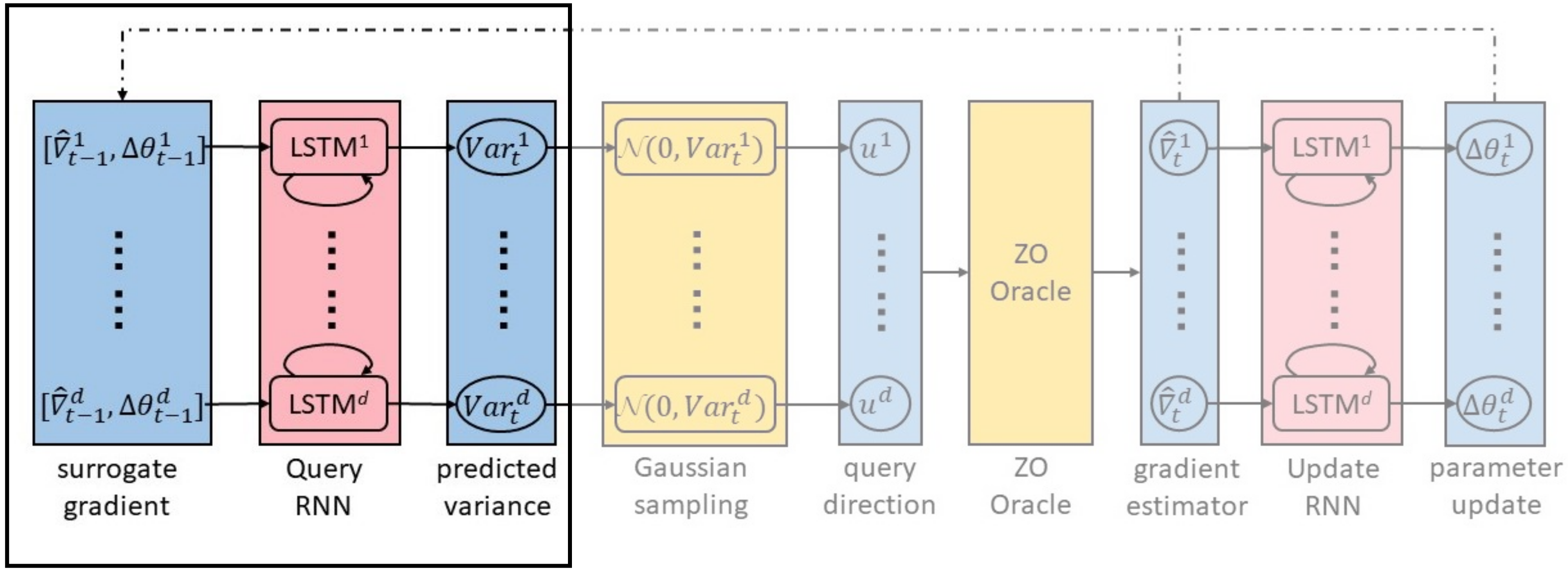
Method

- Jointly learn the parameter update rule and the Gaussian sampling rule
  - UpdateRNN: learn how to propose parameter updates given approximated gradients

$$\theta_t = \theta_{t-1} + \text{UpdateRNN}\left(\hat{\nabla} f(\theta_t)\right)$$

  - QueryRNN: learn to identify the important sampling subspace and adaptively modify the search distribution

$$\Sigma_t = \text{QueryRNN}\left([\hat{\nabla} f(\theta_{t-1}), \Delta\theta_{t-1}]\right)$$



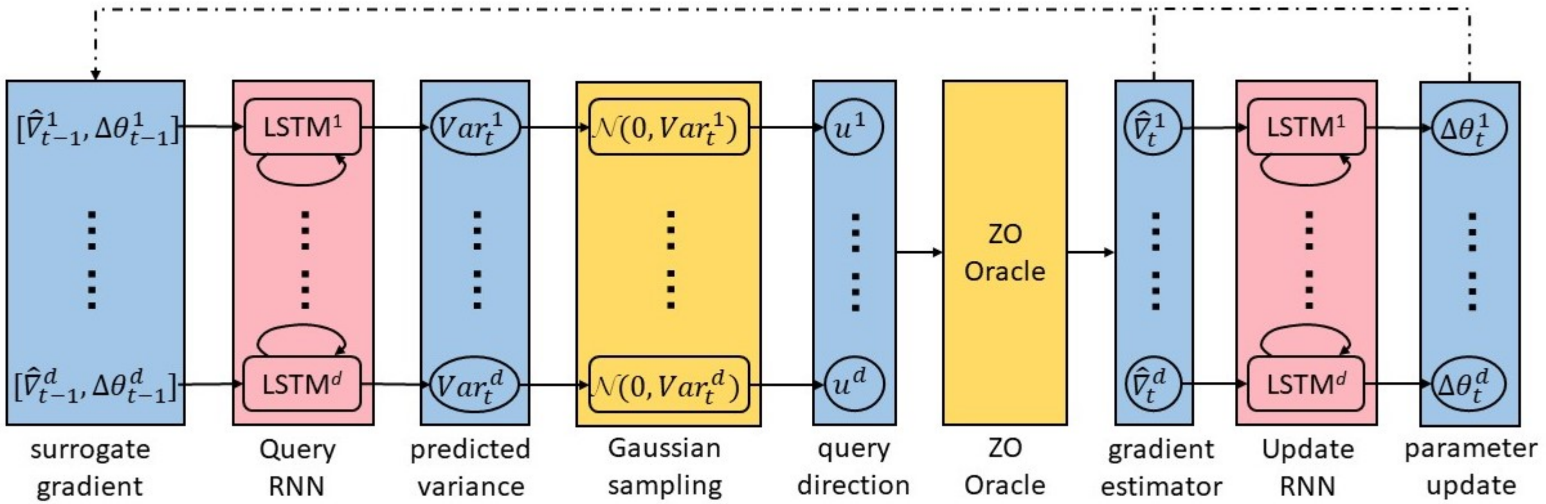
Method

- Jointly learn the parameter update rule and the Gaussian sampling rule
  - UpdateRNN: learn how to propose parameter updates given approximated gradients

$$\theta_t = \theta_{t-1} + \text{UpdateRNN}\left(\hat{\nabla} f(\theta_t)\right)$$

  - QueryRNN: learn to identify the important sampling subspace and adaptively modify the search distribution

$$\Sigma_t = \text{QueryRNN}\left([\hat{\nabla} f(\theta_{t-1}), \Delta\theta_{t-1}]\right)$$



Training the ZO optimizer

- Backpropagate through the Gaussian sampling module (non-differentiable)
  - ✓ Apply reparamerization trick to generate query directions $u \sim N(0, \Sigma_t)$

$$z \sim N(0, I)$$
$$u = \Sigma_t^{1/2} z$$

Training the ZO optimizer

- Backpropagate through the Gaussian sampling module (non-differentiable)
  - ✓ Apply reparamerization trick to generate query directions $u \sim N(0, \Sigma_t)$

$$z \sim N(0, I)$$
$$u = \Sigma_t^{1/2} z$$

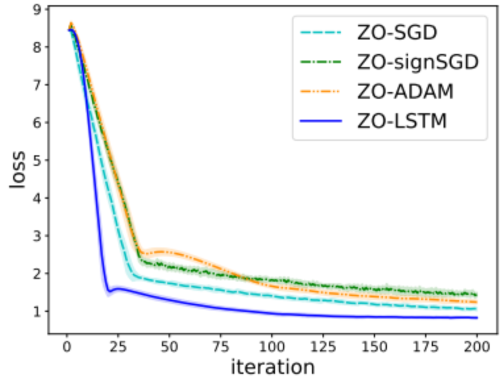
- Backpropagate through the optimizee (zeroth-order)
  - ✓ Apply coordinatewise ZO gradient estimator (optional)

$$\hat{\nabla} f(\theta) = \sum_{i=1}^{d} \frac{f(\theta + \mu e_i) - f(\theta - \mu e_i)}{2\mu} e_i$$

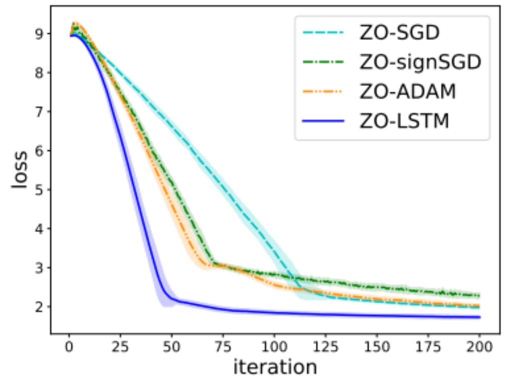
- $\{e_i\}$: standard basis vector with i^{th} coordinate being 1, and others being 0s
- d: optimizee dimension
- μ: smoothing parameter

Experiments

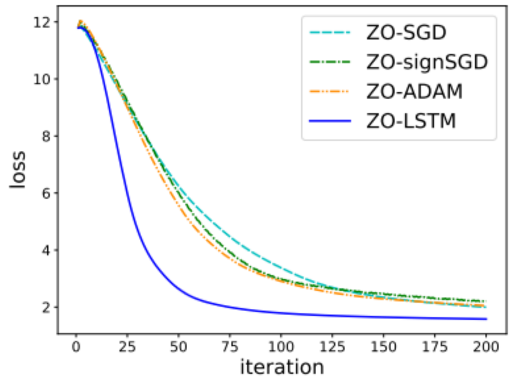
- Black-box adversarial attack



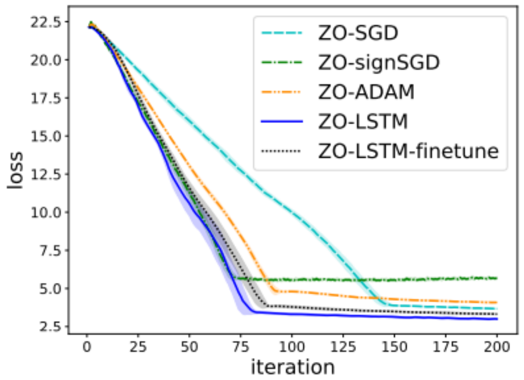
(a) MNIST - Test ID 1094



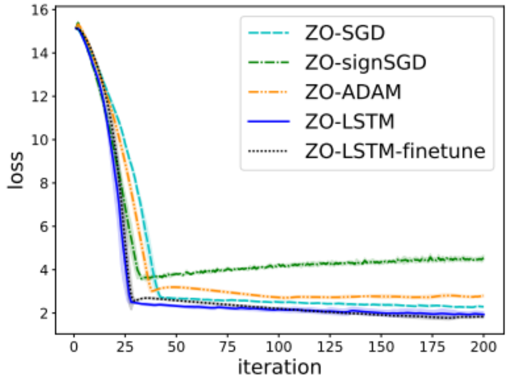
(b) MNIST - Test ID 1933



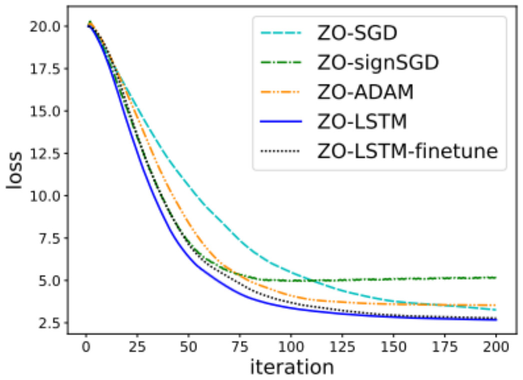
(c) MNIST - Average



(d) CIFAR - Test ID 4293



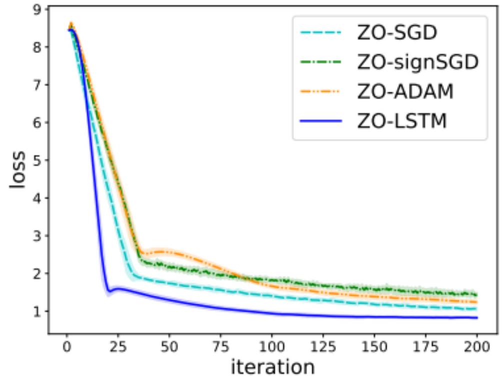
(e) CIFAR - Test ID 9208



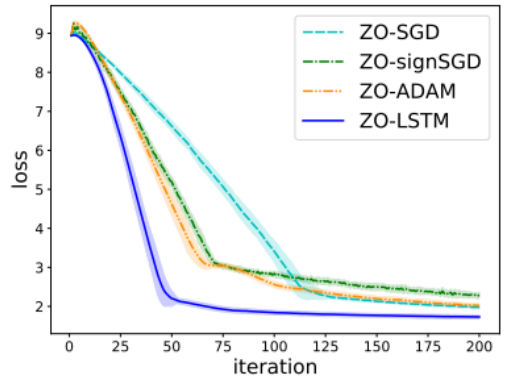
(f) CIFAR - Average

Experiments

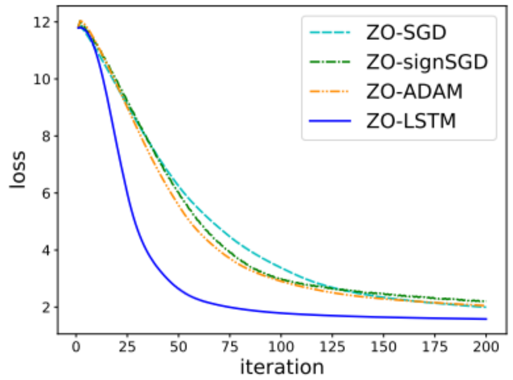
- Black-box adversarial attack



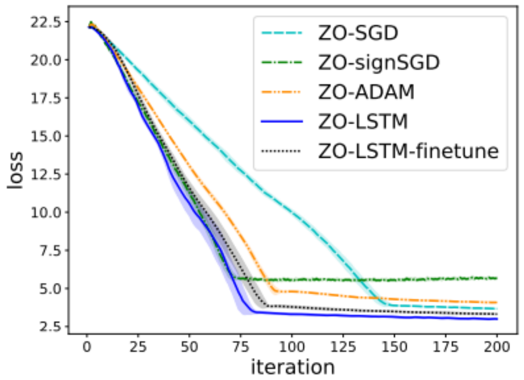
(a) MNIST - Test ID 1094



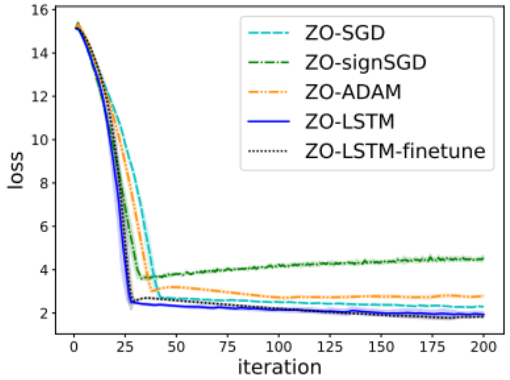
(b) MNIST - Test ID 1933



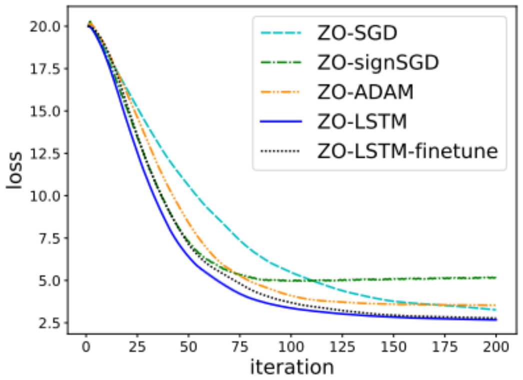
(c) MNIST - Average



(d) CIFAR - Test ID 4293



(e) CIFAR - Test ID 9208

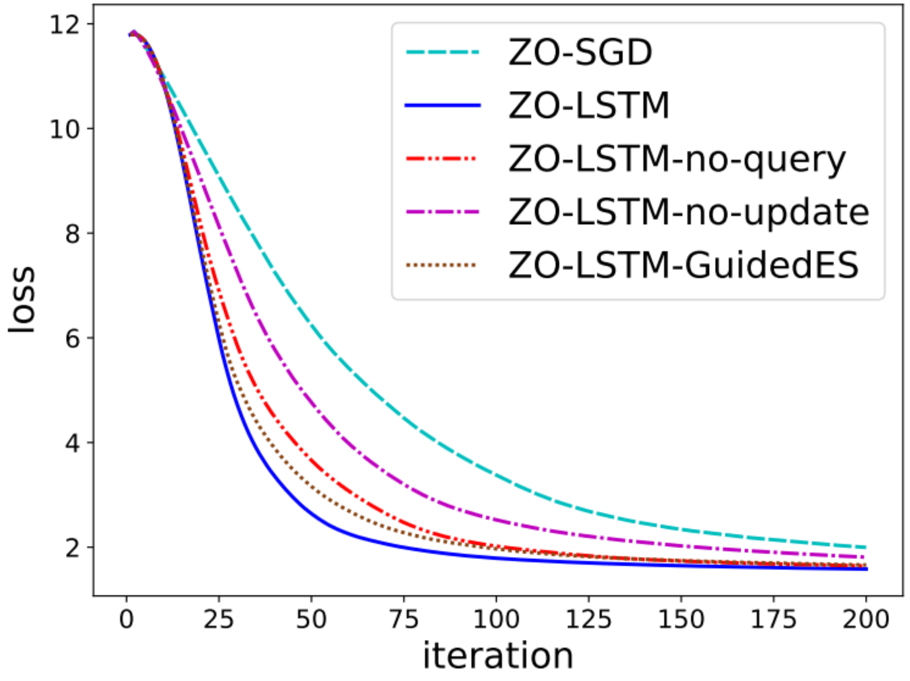


(f) CIFAR - Average

- Promising Application: automatically learned efficient “attacker”

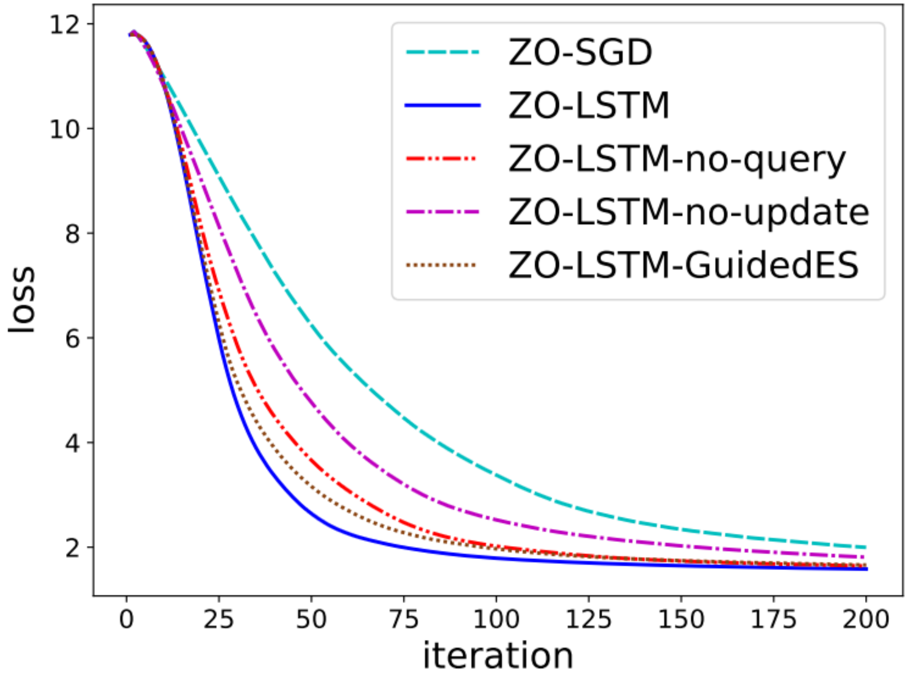
Analytical experiments

- Ablation study
  - ✓ Effectiveness of both modules

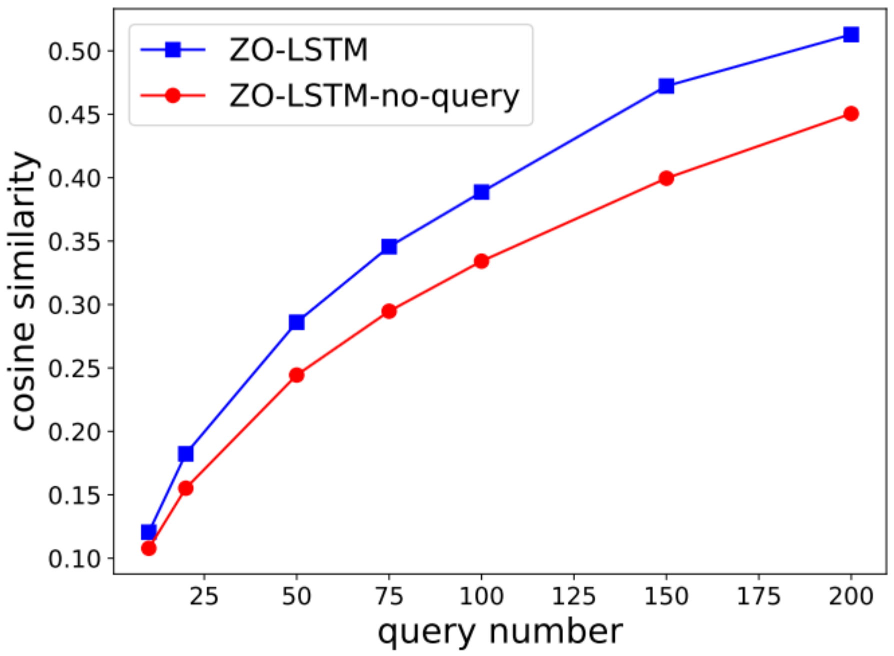


Analytical experiments

- Ablation study
  - ✓ Effectiveness of both modules



- Estimated gradient evaluation
  - ✓ QueryRNN leads to more accurate gradient estimators



Paper link:
https://openreview.net/forum?id=ryxz8CVYDH

Thank you!