LEARNING SAMPLING POLICY FOR FASTER DERIVATIVE FREE OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Zeroth-order (ZO, also known as derivative-free) methods, which estimate a noisy gradient based on the finite difference with two function evaluations, have attracted much attention recently because of its broad applications in machine learning community. The function evaluations are normally requested on a point plus a random perturbations drawn from a (standard Gaussian) distribution. The accurateness of noisy gradient highly depends on how many perturbations randomly sampled from the distribution, which intrinsically conflicts to the efficiency of ZO algorithms. Although there have been much effort made to improve the efficiency of ZO algorithms, however, we explore a new direction, i.e., learn an optimal sampling policy based on reinforcement learning (RL) to generate perturbation instead of using totally random strategy, which make it possible to calculate a ZO gradient with only 2 function evaluations. Specifically, we first formulate the problem of learning a sampling policy as a Markov decision process. Then, we propose our ZO-RL algorithm, i.e., using deep deterministic policy gradient, an actor-critic RL algorithm to learn a sampling policy which can guide the generation of perturbed vectors in getting ZO gradients as accurate as possible. Since our method only affects the generation of perturbed vectors which is parallel to existing efforts of accelerating ZO methods such as learning a data driven Gaussian distribution, we show how to combine our method with other acceleration techniques to further improve the efficiency of ZO algorithms. Experimental results with different ZO estimators show that our ZO-RL algorithm can effectively reduce the query complexity of ZO algorithms especially in the later stage of the optimization process, and converge faster than existing ZO algorithms.

1 INTRODUCTION

Gradient based optimization is the dominant method in machine learning. However, in many fields of science and engineering, explicit gradient information $\nabla f(x)$ is difficult or even infeasible to obtain. Zeroth-order (ZO, also known as derivative-free) optimization, where the optimizer is provided with only function values $f(x)$ (zeroth-order information) instead of explicit (first-order) gradients $\nabla f(x)$, has attracted an increasing amount of attention. ZO optimization (ZOO) can address a wide range of problems and has been studied in a large number of fields such as optimization, online learning and bioinformatics (Koch et al., 2018; Mania et al., 2018; Vemula et al., 2019). One of the famous applications of ZOO is to generate prediction-evasive adversarial examples in the black-box setting (Liu et al., 2018; Papernot et al., 2017), e.g., crafted images with imperceptible perturbations to deceive a well-trained image classifier into misclassification.

The *ZO gradient* which is also referred to as evolutionary strategies (Huning, 1976) is calculated based on the finite difference approach (Strikwerda, 2004) and normally with the following formula.

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

where $\mu > 0$ is the smoothing parameter, $\{u_i\}_{i=1}^{q}$ are the random perturbed vectors (directions) usually drawn from a distribution $p(u)$ which could be standard Gaussian or uniform distribution on a unit sphere (Nesterov & Spokoiny, 2017; Duchi et al., 2012). Essentially, if the smoothing parameter μ approaches infinitesimal, $\lim_{\mu \to 0} f(x + \mu u) - f(x)$ is exactly calculating

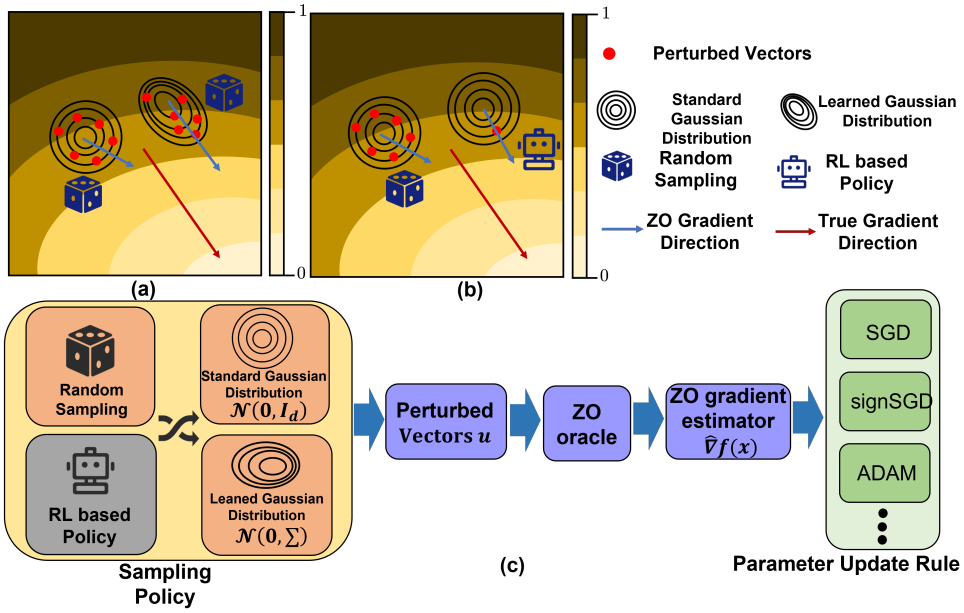


Figure 1: (a) Comparison of the ZO gradient directions obtained by sampling perturbed vectors from the standard Gaussian distribution and the learned Gaussian distribution. (b) Comparison of the ZO gradient directions obtained by a random sampling policy and a RL based policy. (c) The architecture of ZO optimizer.

the directional derivative $\nabla_u f(x)$ along a direction u, which means we use q directional derivatives to approximate the ground truth gradient $\nabla f(x)$ due to $\nabla f(x) = \int_u (\nabla_u f(x) u) p(u) \mathrm{d}u$. From this perspective, the accurateness of ZO gradient highly depends on how many perturbed vectors randomly sampled from the distribution. For a practical ZOO algorithm, e.g., vanilla ZOO Algorithm 1, we should balance the number q of perturbed vectors and the accurateness of ZO gradient to make the overall query complexity[1] of ZOO algorithm as optimal as possible.

Algorithm 1 Vanilla ZOO Algorithm

Input: Smoothing parameter μ, learning rate η and q.
Output: $x \in \mathbb{R}^d$
1: **for** $k = 0$ to $K - 1$ **do**
2: Sampling q perturbed vectors from the standard Gaussian distribution $u_i \sim \mathcal{N}(0, I_d)$.
3: Calculating the ZO gradient $\hat{\nabla} f(x_k)$.
4: Obtain the next update $x_{k+1} = x_k - \eta \hat{\nabla} f(x_k)$.
5: **end for**

To improve the efficiency of ZO algorithms, there have been much effort made in the communities of machine learning and optimization. Specifically, different from sampling perturbed vectors using a standard Gaussian distribution $\mathcal{N}(0, I)$, a few ZOO algorithms (Maheswaranathan et al., 2019; Ruan et al., 2019) use a learned non-isotropic Gaussian distribution $\mathcal{N}(0, \Sigma)$ to generate perturbed vectors. The co-variance matrix Σ of these Gaussian may not be a scale of the identity matrix (illustrated in Fig. 1.a). The rationale of these methods is easier to illustrate using the black-box adversarial attack example. For this type of task, there is usually a well-defined significant subspace that is more prone to attack, and perturbed directions through this subspace naturally leads to faster learning convergence. Similar to this, Evolutionary Strategies (ES) such as Natural ES (Wierstra et al., 2008), CMA-ES (Hansen, 2006), and Guided ES (Maheswaranathan et al., 2019) were also proposed to guide the sampling direction in ZOO.

As discussed above, the learned sampling distribution is beneficial to calculate a more accurate ZO gradient. Parallel to learn a sampling distribution, a natural question is whether it is feasible to learn a sampling policy to generate one gradient oriented perturbed vectors instead of using multiple

[1]The overall number of queries of function evaluations is called query complexity.

randomly sampled perturbed vectors to approximate the ground truth gradient. The quick answer is yes because it can be easily proved that there must exist one perturbation vector u' such that $\nabla_{u'} f(x)u' = \int_u (\nabla_u f(x)u)p(u)\mathrm{d}u = \nabla f(x)$. If we could find such perturbation vector u', the ZO gradient can be directly approximated with only two queries as follows.

$$\hat{\nabla} f(x) = \frac{1}{\mu}[f(x + \mu u') - f(x)]u' \tag{2}$$

The subsequent issue is how to find this kind of perturbed vector u' to approximate the gradient as accurate as possible, which is still an open problem and even not be well noticed in the community as far as we know. In this paper, we will take the approach of Reinforcement Learning (RL) to learn a good sampling policy to solve this issue.

As it regains its popularity recently because of star projects like AlphaGo (Wang et al., 2016), DQN (Mnih et al., 2015) and AlphaStar (Arulkumaran et al., 2019), RL (Mnih et al., 2015) is a formal framework in which a learning agent can continuously optimize its policy to obtain higher cumulative rewards while interacting with an uncertain environment. If we take the ZOO algorithm as the RL agent whose goal is to optimize a ZOO problem as fast as possible. The target function f, current parameter x_t and the calculation mechanism of noisy gradient (i.e., Eq. (1)) are all parts of the environment. Each step, the agent picks a perturbation vector and pass it over to the environment for execution. The environment takes its step to update x_k and output x_{k+1} as the new observation for the RL agent. The RL agent then receives a reward of $f(x_{k+1}) - f(x_k)$. This is a typical RL problem, which can be solved by any model-free reinforcement learning method. Ideally, with sufficient learning, the agent will become smart enough in selecting a good perturbation point upon every move it make, which is essentially a good sampling policy. We provide Fig 1.(a) to intuitively show the benefit of using the learned sampling distribution, and also provide Fig 1.(b) to intuitively demonstrate the benefit of using the RL based policy. The effective prediction of gradient oriented perturbed vector by RL can reduce the query complexity and speed up the convergence of the ZOO algorithm.

In this paper, we propose a zero-order algorithm based on reinforcement learning (ZO-RL) to learn the sampling policy in ZOO using the policy gradient algorithm. Specifically, we use an actor-critic algorithm called deep deterministic policy gradient (DDPG) (Lillicrap et al. (2015), with two neural network function approximators. Compared with the stochastic policy gradient algorithm, deterministic policies have the advantages of requiring less data to be sampled, and stable performance in a series of tasks with continuous action spaces. The RL based policy guide the optimizer to estimate more accurate ZO gradients in the parameter space to reduce the variance. Especially, we can combine our ZO-RL with existing the ZOO algorithms that utilize the improved parameter update rule and learned sampling distribution. Experimental results for different ZOO problems show that our ZO-RL algorithm can effectively reduce the variances of ZO gradient by learning the sampling policy, and converge faster than existing ZOO algorithms in different scenarios.

Contributions. The main contributions of this paper are summarized as follows.

1. We propose to learn the sampling policy by reinforcement learning instead of using random sampling as in the standard ZOO algorithms to generate perturbed vectors.
2. We conduct extensive experiments to show that our ZO-RL algorithm can effectively reduce the variances of ZO gradients by learning a sampling policy, and converge faster than existing ZOO algorithms in different scenarios.

2 Related Work

In order to construct ZO gradients that are closer to the true gradient direction and enable the ZOO algorithm to obtain convergence with fewer queries, existing works focus on learning an adaptive Gaussian distribution $\mathcal{N}(0, \Sigma)$ as discussed previously. Thus, they sample the perturbed vectors by $u_i \sim \mathcal{N}(0, \Sigma)$ that the co-variance matrix Σ may not be a scale of the identity matrix. Specifically, Maheswaranathan et al. (2019) utilized evolution strategies to let co-variance matrix Σ track a low-dimensional subspace, which is related with recent history of ZO gradients during optimization. Ruan et al. (2019) utilized RNN to learn an adaptive co-variance matrix Σ and dynamically guide the sampling distribution. By learning the significant sampling distribution, more accurate ZO gradient

can be obtained for a fixed query budget, which can improve the convergence of ZOO algorithms. In this paper, from different angle, we apply RL to learn a smarter sampling policy to replace the plain random sampling used in existing methods. It also should be noted that, our proposed ZO-RL algorithm is paralleled to the existing ZOO algorithms of learning a sampling distribution. Thus, it is possible to combine them together to generate a better ZO gradient estimation with only two queries.

As discussed above, our paper considers using RL to accelerate the ZO algorithms. Interestingly, there is one opposite research direction compared to our paper, i.e., utilizing ZO algorithms to optimize RL model (Mania et al., 2018; Vemula et al., 2019). Specifically, policy gradient methods (Mnih et al., 2016) is a popular type of RL approach used in complex and uncertain environments (Arulkumaran et al., 2019), which relies on random exploration in the sampling distribution to learn sampling policy for guiding sampling direction, and directly updates a RL agent in the policy space using stochastic gradient descent. In particular, Mania et al. (2018); Vemula et al. (2019) used ZO gradients instead of explicit gradients to train static, linear policies for continuous control problems to achieve higher sample efficiency.

3 Learning Sampling Policy in Zeroth-Order Optimization

We consider the problem of finding a sampling policy that encourages the sampled perturbed vectors to be more efficient in calculating the ZO gradients and enable the ZOO algorithm to obtain convergence with fewer queries. In the this paper, we use RL to learn a smarter sampling policy compared to plain random sampling policy. RL provides a framework in which the agent can learn the best action to take by subsequently receiving rewards from the environment with which it interacts. We view the each query of ZOO algorithm as the execution of a fixed policy in a MDP as a tuple $(\mathcal{S}, \mathcal{A}, \mathbb{P}_{sa}, R)$:

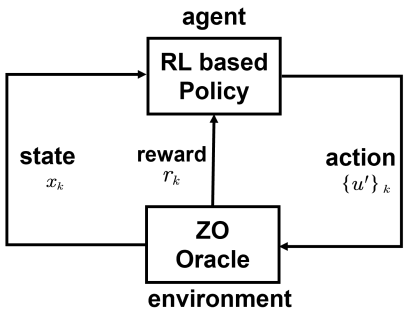


Figure 2: Illustration of learning sampling policy for ZOO base on RL.

1. State space $\mathcal{S} \subset \mathbb{R}^q$: We choose $s_k = x_k \in \mathcal{S}$ to describe the current state of ZO algorithm, where x_k is the point location for the kth iteration.

2. Action space $\mathcal{A} \subset \mathbb{R}^p$: We choose $a_k = \{u'\}_k \in \mathcal{A}$ as the action, where $\{u'\}_k$ is the perturbed vector.

3. Transition probability $\mathbb{P}_{sa} = \mathbb{P}(\cdot|s, a)$: Unknown in the model-free RL.

4. Reward function $R(s, a) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$: We consider the reward function $R(s_k, a_k) = r_k = f(x_{k+1}) - f(x_k)$ as the difference between the function values at the current point location x_k and the immediately preceding point location x_{k+1} after the action a_k is performed, which encourages the learned policy to reach the minimum of the function value as quickly as possible.

Through the framework of MDP, we propose a ZOO algorithm based on reinforcement learning (ZO-RL). At each query of ZO-RL algorithm, the agent outputs the perturbed vectors $\{u_i\}_k$ according to the current state x_k. Then, the agent receives rewards and next state x_{k+1} by interacting with the environment, and learns the sampling policy to maximize rewards. We show the illustration of ZOO base RL in Figure 2.

In the following, we first introduce the principle of our ZO-RL algorithm. Then, we introduce the network structure and the batch normalization technique which are used in our ZO-RL algorithm. Finally, we discuss to combine our ZO-RL with existing ZOO algorithms to further accelerate the existing ZOO algorithms.

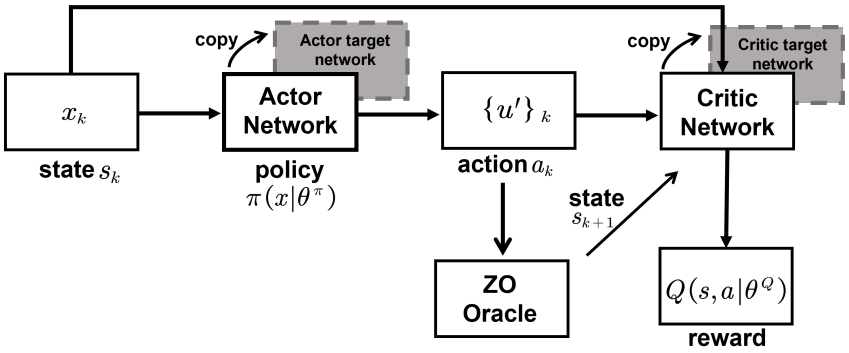


Figure 3: Illustration of of ZO-RL algorithm.

3.1 PRINCIPLE OF OUR ZO-RL ALGORITHM

Since the action space is continuous in ZOO, we use the deterministic sampling policy. Compared with the stochastic policy gradient algorithm, the deterministic policy has the advantages of requiring less data to be sampled so that it achieve higher efficiency for the algorithm, and performing stably in a series of tasks with continuous action space. Thus, to find the optimal policy to approach the true gradient direction, we use deep deterministic policy gradient (DDPG) to learn sampling policy π. DDPG is an actor-critic and model-free algorithm (Konda & Tsitsiklis, 2000) for RL over continuous action spaces and output deterministic actions in a stochastic environment to maximize cumulative rewards.

The DDPG has two neural network function approximators. One is called the actor network which learns a deterministic sampling policy $\pi(x|\theta^\pi)$ with neural network weights θ^π. The other is called the critic network, which outputs a state-action value function $Q(s, a|\theta^Q)$ with neural network weights θ^Q to evaluate the value of the action performed. In addition, DDPG creates a copy of the actor and critic networks, $Q'(s, a|\theta^{Q'})$ and $\pi'(x|\theta^{\pi'})$ respectively, that are used for calculating the target values. The weights of these target networks are updated by making them slowly track the learned networks: $\theta' \rightarrow \tau\theta' + (1-\tau)\theta'$ with $\tau \ll 1$. This means that the target values are constrained to change slowly, greatly improving the stability of learning. This simple change moves the relatively unstable problem of learning the action-value function closer to the case of supervised learning.

At each iteration of ZO-RL, we use actor network to output action $\{u_i\}$ according to current state x_k. Then, we transfer the action $\{u_i\}$ to ZO Oracle, calculate the ZO gradient $\tilde{g}$ and output the next state x_{k+1}, and use the critic network to output the reward of this action. We call $\{\{u_i\}_k, x_k, x_{k+1}, r_k\}$ a transition. Good transitions can accelerate the learning speed of the agent. Thus, in order to better initialize actor network, we alternately use the random sampling policy to sampling perturbed vectors and interact with the current environment to obtain ZO gradient estimator $\hat{g}$. We store these transitions generated by random sampling policy into relay memory buffer, and use them to update the actor network. We use a cosine similarity ρ to calculate the similarity of gradient directions obtained by plain random sampling and our sampling policy $\pi(x|\theta^\pi)$:

$$\rho = \frac{\hat{g} \cdot \tilde{g}}{||\hat{g}|| \cdot ||\tilde{g}||} \tag{3}$$

where $\hat{g}$ is the ZO gradient estimator defined in (1) with the perturbed vectors generated by random sampling policy, and $\tilde{g}$ is the ZO gradient estimator defined in (2) with the perturbed vector generated by our RL based policy. Set a threshold ϵ, if $\rho < \epsilon$, we use the random sampling policy to generate perturbed vectors, otherwise we use the learned policy $\pi(x|\theta^\pi)$ to generate perturbed vector.

At each iteration, we minimize a squared-error loss L to update the critic network parameter:

$$\min_{\theta^Q} L = \frac{1}{N}\sum_{k=1}^{N}(y_k - Q(s, a|\theta^Q))^2 \tag{4}$$

where y_k represents the TD target denoted as

$$y_k = r_k + \gamma Q'\Big(s_{k+1}, \pi'(x_{k+1}|\theta^{\pi'})|\theta^{Q'}\Big) \tag{5}$$

We maximize the cumulative reward using a sampled policy gradient to the actor network parameter:

$$\nabla_{\theta^\pi} J \approx \frac{1}{N}\sum_{k=1}^{N} \nabla_{\{u_i\}} Q(s, a|\theta^Q)|_{s=x_i, a=\pi(x_i)} \cdot \nabla_{\theta^\pi}\pi(x|\theta^\pi)|_{x_i} \tag{6}$$

where $J = \mathbb{E}_{s\sim\beta, a\sim\pi}[R(s,a)]$ represents the expected cumulative reward, β is the distribution of state space.

We show the illustration of our ZO-RL algorithm in Fig. 3 and summarize our ZO-RL algorithm in Algorithm 2. Note that lines 4-6 of Algorithm 2 corresponds to the vanilla ZO algorithm (i.e., Algorithm 1). This is because the ZO estimator with RL is not accurate in the early stage. However, if $\rho > \epsilon$ which means that the accurateness of ZO estimator with RL is acceptable, the updating rules of ZOO immediately switch to the ZO estimator with RL.

Algorithm 2 Zeroth-Order Optimization for Reinforcement Learning

Input: Hyper-parameter ϵ, smoothing parameter μ, the number of sampled perturbed vectors q and learning rate η, mini-batch size N.
Output: Sampling policy $\pi(x|\theta^\pi)$.

1: Initialize $\rho = 0$.
2: **for** $k = 1$ to K **do**
3: **if** $\rho < \epsilon$ **then**
4: Randomly sample q perturbed vectors $\{u_i\}_{i=1}^q$ from the standard Gaussian distribution $\mathcal{N}(0, I_d)$.
5: Calculate the ZO gradient estimator $\hat{g}_k$ according to (1) with the q perturbed vectors.
6: Obtain the next update $x_{k+1} = x_k - \eta \cdot \hat{g}_k$.
7: **end if**
8: Sample perturbed vectors u' according to the sampling policy $\pi(x_k|\theta^\pi)$.
9: Calculate the ZO gradient estimator $\tilde{g}_k$ according to (2) with the perturbed vectors u'.
10: **if** $\rho < \epsilon$ **then**
11: Update ρ based on $\hat{g}_k$ and $\tilde{g}_k$ according to (3).
12: **else**
13: Obtain the next update $x_{k+1} = x_k - \eta \cdot \tilde{g}_k$.
14: **end if**
15: Store transition $\{\{u_i\}_k/\{u'\}_k, x_k, x_{k+1}, r_k\}$ in a replay memory buffer.
16: Observe N transitions from replay memory buffer to update the actor network and critic network.
17: **end for**

3.2 Network Structure and Batch Normalization

The choice of the structure of the critic and actor nets is important because they are used not only to evaluate sampling policies, but also to learn sampling policies. We choose the convolutional neural network (CNN) (Sezer & Ozbayoglu, 2018) both for the critic net and the actor net.

The difficulty of policy gradient method is due to the lack of information gradient of policy performance. Specifically, gradients may not exist due to non-smoothness of the environment or policy, or may only be available as high-variance estimates because the environment usually can only be accessed via sampling. In order to make the problem smooth and have a way of to estimate its gradients, we add noise in action space, which is done by sampling the actions from an appropriate distribution. In our ZO-RL, we select action $\{u_i\}_k = \pi(x_k|\theta^\pi) + \mathcal{N}_k$ according to to the current policy and exploration noise, where $\mathcal{N}_k(0, I)$ is a standard Gaussian distribution.

The parameters of the ZO optimizer have different descent rates in different dimensions and the range may be different in different environments. This may make it difficult for the network to learn efficiently and find hyper-parameters that generalize the scale of state values in different environments. One way to address this issue is to manually scale features so that they are in a similar range

across environments and units. We address this problem by adapting one of the latest techniques in deep learning, called batch normalization (Santurkar et al., 2018). This technique normalizes each dimension of a sample in a mini-batch to have unit mean and variance. In addition, batch normalization maintains a running average of the mean and variance to be used for normalization during testing. In deep networks, batch normalization is used to minimize the co-variance bias during training, by ensuring that each layer receives whitened inputs.

3.3 COMBINING OUR ZO-RL WITH EXISTING ZOO ALGORITHMS

In this subsection, we discuss how to combine our ZO-RL with an existing ZOO algorithm based on the parameter update rule or the learned sampling distribution. For example, (Ruan et al., 2019) proposed a ZO optimization algorithm called ZO-LSTM, which replaces parameter update rule as well as guided sampling rule to sample the perturbed vectors with learned recurrent neural networks (RNN). Especially, they updated the parameter through a Long Short-Term Memory (LSTM) network called UpdateRNN:

$$x_t = x_{t-1} + \text{UpdateRNN}(\hat{\nabla} f(x_t)) \tag{7}$$

where x_t is the optimizer parameter at iteration t. UpdateRNN can reduce the negative impact of high variances of ZO gradient due to long-term dependence, in addition to learning to compute parameter updates adaptively by exploring the loss landscape. They use another LSTM network called QueryRNN to learn the sampling distributions. They dynamically predict the converiance matrix Σ_k:

$$\Sigma_t = \text{QueryRNN}([\hat{\nabla} f(x_t), \Delta x_{t-1}]) \tag{8}$$

QueryRNN can increase the sampling probability in the direction of the bias of the estimated gradient or the parameter update of the previous iteration.

Although they considered both the sampling distribution and the parameter update rule, they still used random sampling for the perturbed vectors. Using the RL based policy on the learned sampling distribution can further speed up the convergence of ZOO algorithms. Thus, we can combine our ZO-RL algorithm with ZO-LSTM algorithm. Especially, we first train the UpdateRNN using standard Gaussian random vectors as query directions. Then we freeze the parameters of the UpdateRNN and train the QueryRNN. Finally, we use the previous work as a warm start and use our ZO-RL in the pre-learning distribution to learn the sampling policy. In addition, other ZOO algorithms based on parameter update rules such as (Lian et al., 2016; Chen et al., 2017), can be directly combined with our ZO-RL algorithm.

4 EXPERIMENTS

In this section, we empirically demonstrate the superiority of our proposed ZO optimizer on a practical application (black-box adversarial attack on MNIST dataset) and a synthetic problem (nonconvex binary classification problems on benchmark datasets). To show the effectiveness of the learned sampling policy, we compare the convergence behavior of our proposed ZO optimizer with existing ZOO algorithms under a same query number. To show that our algorithm can estimate the ground-truth gradient direction more accurately, we count the cosine similarity between the ground-truth gradient direction and the ZO gradient direction computed by existing ZO algorithms and our proposed ZO optimizer.

Specifically, we obtain ZO gradient estimator along sampled directions via ZO Oracle. Since our algorithm is the first one to learning the sampling policy, we compare the performance of the ZO gradient estimators sampled form the standard Gaussian distribution and two learned Gaussian distributions, *i.e.* using different covariance matrix Σ. In addition, we compare the algorithm of synchronously learning sampling and distribution policy by combining our algorithm with other algorithms. The five algorithms for calculating ZO gradient estimators are summarized as follows:

1. ZO-GS (Wang et al., 2019): Randomly Sampling the perturbed vectors u_i from a standard Gaussian distribution.
2. ZO-LSTM (Ruan et al., 2019): They learned the Gaussian sampling rule and dynamically predicted the covariance matrix Σ for query directions with recurrent neural networks.
3. Guided ES (Maheswaranathan et al., 2019): They let the covariance matrix Σ be related with the recent history of surrogate gradients during optimization.

4. ZO-RL: Our proposed ZO algorithm learns the sampling policy through reinforcement learning.
5. ZO-RL-LSTM: Our proposed ZO algorithm combined with ZO-LSTM to learn sampling policy on a learned Gaussian distribution.

4.1 IMPLEMENTATION

For each task, we tune the hyper-parameters of baseline algorithms to report the best performance. We coarsely tune the constant δ on a logarithmic range $\{0.01; 0.1; 1; 10; 100; 1000\}$ and set the learning rate of baseline algorithms to $\eta = \delta/d$, where d is the dimension of dataset. We set the smoothing parameter $\mu = 0.01$ in all experiments. To ensure fair comparison, all optimizers use the same number of query directions in each iteration to obtain the ZO gradient.

4.2 ADVERSARIAL ATTACK TO BLACK-BOX MODELS

We consider generating adversarial examples to attack black-box DNN image classifier and formulate it as a zeroth-order optimization problem. The targeted DNN image classifier $F(x) = [F_1, F_2, \cdots, F_K]$ takes as input an image $x \in [0, 1]^d$ and outputs the prediction scores of K classes. Given an image $x_0 \in [0, 1]^d$ and its corresponding true label $t_0 \in [1, 2, \cdots, K]$, an adversarial sample x is visually similar to the original image x_0 but leads the targeted model F to make wrong prediction other than t_0. The black-box attack problem is normally formulated as follows.

$$\max_x \{F_{t_0}(x) - \max_{j \neq t_0} F_j(x), 0\} + c\|x - x_0\|_p \tag{9}$$

where the first term is the attack loss which measures how successful the adversarial attack is and penalizes correct prediction by the targeted model. The second term is the distortion loss (p-norm of added perturbation) which enforces the perturbation added to be small and c is the regularization coefficient. In our experiment, we use ℓ_1 norm (i.e., $p = 1$), and set $c = 0.1$ for MNIST attack task. Due to the black-box setting, one can only compute the function values of the above objective, which leads to ZOO problems (Chen et al., 2017). Note that attacking each sample x_0 in the dataset corresponds to a particular ZOO problem instance, which motivates us to train a ZO optimizer offline with a small subset, and apply it to online attack to other samples with faster convergence (which means lower query complexity) and lower final loss (which means less distortion). We randomly select 50 images that are correctly classified by the targeted model in each test set to train the optimizer and select another 50 images to test the learned optimizer. The number of sampled query directions is set to $q = 20$ for MNIST.

4.3 NON-CONVEX BINARY CLASSIFICATION PROBLEMS

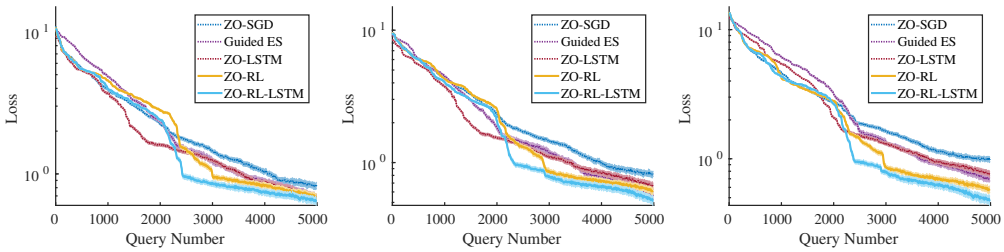
We consider a binary classification problem with a non-convex least squared loss function $\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n (y_i - 1/(1 + e^{-w^T x_i}))^2$. Here (x_i, y_i) is the ith data sample containing feature $x_i \in \mathbb{R}^d$ and label $y_i \in \{-1, 1\}$. We compare the algorithms on benchmark datasets (heat scale, german and a9a[2]). All the algorithms only access to the ZO oracle of function value evaluations. We use the same set of hyper-parameters for different datasets and repeated runs in the experiments. The number of query directions are set to $q = 20$. For each dataset, we repeat the experiment 10 times and report the average and the standard deviation.

4.4 DISCUSSION AND ANALYSIS

Fig. 4 shows the black-box attack loss versus query number using different ZOO algorithms. Fig. 5 shows the non-convex least squared loss versus query number using different ZOO algorithms. The loss curves are averaged over 10 independent random trails and the shaded areas indicate the standard deviation. The results clearly show that our ZO-RL algorithm can effectively reduce the query complexity of ZOO algorithms especially in the later stage of the optimization process, and our ZO-RL-LSTM can always obtain the best results by combining learned sampling policy and sampling distribution. This is due to the fact that our ZO-RL algorithm learn a smarter sampling policy though RL instead of random sampling.

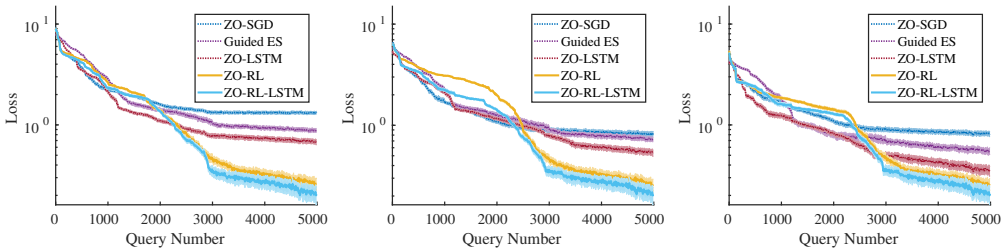
Fig. 6 plots the cosine similarities between ZO gradient estimator and ground-truth gradient for non-convex binary classification problems. The cosine similarities curves are averaged over 10

[2] http://archive.ics.uci.edu/ml/datasets.html



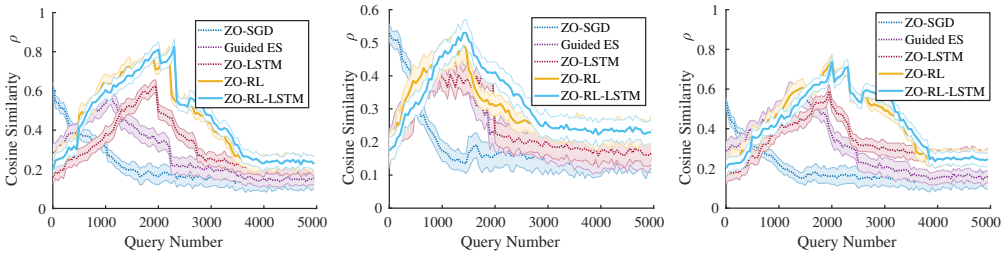
(a) MNIST TEST 1 (b) MNIST TEST 2 (c) MNIST TEST 3

Figure 4: Adversarial attack to black-box models.



(a) a9a (b) german (c) heat scale

Figure 5: Non-convex binary classification problems.



(a) a9a (b) german (c) heat scale

Figure 6: Cosine similarities between ZO gradient estimator and ground-truth gradient for non-convex binary classification problems.

independent random trails and the shaded areas indicate the standard deviation. The results show that the direction of the ZO gradient estimator generated by our ZO-RL algorithm is closer to the direction of the ground-truth gradient compared to other ZOO algorithms. In the later stage of the optimization process, the convergence of ZO gradient leads to the reduction of the reward obtained by our ZO-RL algorithm in exploring action space. Thus, the cosine similarity decreases after the convergence of the ZOO algorithm.

5 Conclusion

We proposed a new reinforcement learning based sampling policy for generating the perturbations in ZOO instead of using the existing random sampling. The learned sampling policy guides the perturbation (direction) in the parameter space to estimate a ZO gradient as accurate as possible. Since our method only affects the generation of perturbed vectors, it can be used with other acceleration techniques to further improve the efficiency of ZO optimization. Especially, our ZO-RL can be combined with the existing ZO algorithms that could further accelerate them. Experimental results on different ZOO algorithms show that our ZO-RL algorithm can effectively reduce the query complexity of ZO algorithms especially in the later stage of the optimization process, and converge faster than existing ZO algorithms in different scenarios.

REFERENCES

Kai Arulkumaran, Antoine Cully, and Julian Togelius. Alphastar: An evolutionary computation perspective. In *Proceedings of the genetic and evolutionary computation conference companion*, pp. 314–315, 2019.

Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pp. 15–26, 2017.

John C Duchi, Peter L Bartlett, and Martin J Wainwright. Randomized smoothing for stochastic optimization. *SIAM Journal on Optimization*, 22(2):674–701, 2012.

Nikolaus Hansen. The cma evolution strategy: a comparing review. In *Towards a new evolutionary computation*, pp. 75–102. Springer, 2006.

Alois Huning. Evolutionsstrategie. optimierung technischer systeme nach prinzipien der biologischen evolution, 1976.

Patrick Koch, Oleg Golovidov, Steven Gardner, Brett Wujek, Joshua Griffin, and Yan Xu. Autotune: A derivative-free optimization framework for hyperparameter tuning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 443–452, 2018.

Vijay R Konda and John N Tsitsiklis. Actor-critic algorithms. In *Advances in neural information processing systems*, pp. 1008–1014, 2000.

Xiangru Lian, Huan Zhang, Cho-Jui Hsieh, Yijun Huang, and Ji Liu. A comprehensive linear speedup analysis for asynchronous stochastic parallel optimization from zeroth-order to first-order. *Advances in Neural Information Processing Systems*, 29:3054–3062, 2016.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Liu Liu, Minhao Cheng, Cho-Jui Hsieh, and Dacheng Tao. Stochastic zeroth-order optimization via variance reduction method. *arXiv preprint arXiv:1805.11811*, 2018.

Niru Maheswaranathan, Luke Metz, George Tucker, Dami Choi, and Jascha Sohl-Dickstein. Guided evolutionary strategies: Augmenting random search with surrogate gradients. In *International Conference on Machine Learning*, pp. 4264–4273. PMLR, 2019.

Horia Mania, Aurelia Guy, and Benjamin Recht. Simple random search of static linear policies is competitive for reinforcement learning. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 1805–1814, 2018.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937. PMLR, 2016.

Yurii Nesterov and Vladimir Spokoiny. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17(2):527–566, 2017.

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pp. 506–519, 2017.

Yangjun Ruan, Yuanhao Xiong, Sashank Reddi, Sanjiv Kumar, and Cho-Jui Hsieh. Learning to learn by zeroth-order oracle. *arXiv preprint arXiv:1910.09464*, 2019.

Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? *arXiv preprint arXiv:1805.11604*, 2018.

Omer Berat Sezer and Ahmet Murat Ozbayoglu. Algorithmic financial trading with deep convolutional neural networks: Time series to image conversion approach. *Applied Soft Computing*, 70: 525–538, 2018.

John C Strikwerda. *Finite difference schemes and partial differential equations*. SIAM, 2004.

Anirudh Vemula, Wen Sun, and J Bagnell. Contrasting exploration in parameter and action space: A zeroth-order optimization perspective. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 2926–2935. PMLR, 2019.

Fei-Yue Wang, Jun Jason Zhang, Xinhu Zheng, Xiao Wang, Yong Yuan, Xiaoxiao Dai, Jie Zhang, and Liuqing Yang. Where does alphago go: From church-turing thesis to alphago thesis and beyond. *IEEE/CAA Journal of Automatica Sinica*, 3(2):113–120, 2016.

Jun-Kun Wang, Xiaoyun Li, and Ping Li. Zeroth order optimization by a mixture of evolution strategies. 2019.

Daan Wierstra, Tom Schaul, Jan Peters, and Juergen Schmidhuber. Natural evolution strategies. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pp. 3381–3387. IEEE, 2008.