# Stochastic zeroth-order and first order optimization: Sensitivity to learning rate and model architecture

Yihan Wang, Xiaocheng Zhang, Louis-Alexandre Leger

*How well do zero-order optimization methods do for ML applications, compared to standard first-order methods?*

***Abstract*—The expensive or in some cases even unavailable derivatives of the objective function set limits to the application of gradient-based optimization methods, like first-order (FO) methods. Zeroth-order (ZO) methods on the other hand, being capable of solving the optimization problems similarly to gradient-based methods without actual computation of gradient are thus in demand. This paper compares the performance of widely-used FO optimizers: SGD and signSGD, and their ZO versions: ZO-SGD and ZO-signSGD with different hyperparameters and neural network configurations. Different sensitivity to hyperparameters, model architecture and scaling was found for ZO and FO optimizers. Hybrid optimization was also highlighted as a promising approach.**

## I. Introduction

Applications in signal processing, machine learning, and deep learning can involve dealing with complex optimization problems that are difficult to solve analytically with objective functions' gradients generally inaccessible. The optimization corresponding to these problems belongs to ZO optimization regarding black-box models. As gradient-free counterparts of FO methods, ZO optimization methods conduct function value-based gradient approximations. A typical ZO optimization iteratively performs three major steps: gradient estimation, descent direction computation, and solution update [1].

This report proposes to compare the performance of the most widely-used FO optimization algorithms, i.e., Stochastic Gradient Descent (SGD) [2] and signSGD [3] with their ZO counterparts, ZO-SGD [4] and ZO-signSGD [5]. Specifically, convergence speed, stability, and overall performance of the 4 optimizers will be compared given different neural network configurations and hyperparameters in deep learning tasks.

## II. Zeroth-order optimization algorithm

In this section, we provide a brief overview of the zeroth-order optimization algorithms used in this paper. The generic form of ZO optimization is shown as Algorithm 1:

Step 1) is the essential idea of ZO optimization method. For objective function $f(x) : \mathbb{R}^d \to \mathbb{R}$ on a $d$-dimension variable $\mathrm{x} \in \mathbb{R}^d$, the one-point gradient estimation uses the forward differences:

$$\hat{\nabla} f(\mathbf{x}) := \frac{\phi(d)}{\epsilon} f(\mathbf{x} + \epsilon \mathbf{u})\mathbf{u} \tag{4}$$

where $\mathbf{u} \sim p$ is a random direction vector drawn from a certain distribution $p$, and in this report we have chosen a uniform distribution on the unit sphere of $\mathbb{R}^d, p \sim \mathcal{U}(\mathcal{S}(0,1))$.

**Algorithm 1** Generic Form of ZO Optimization

**Initialize** $x_0 \in \chi$, gradient estimation operation $\phi(\cdot)$, number of iterations $T$, and learning rate $\eta_t > 0$ at iteration $t$,

**for** $t = 1, 2, ..., T$ **do**

**1) Gradient estimation:**

$$\widehat{g}_t = \phi(\{f(x_t; \xi_i)\}_{i \in \Omega_t}^t), \tag{1}$$

where $\Omega_t$ denotes a set of minibatch stochastic samples used at iteration $t$,

**2) Descent direction computation:**

$$m_t = \psi(\{\widehat{g}_i\}_{i=1}^t), \tag{2}$$

**3) Point Updating:**

$$x_t = \Pi_\chi(x_{t-1}, m_t, \eta_t), \tag{3}$$

where $\Pi_\chi$ denotes a point-updating operation subject to the constraint $x \in \chi$

**end for**

$\epsilon > 0$ is a perturbation radius and $\phi(d)$ denotes a dimension-dependent factor related to the distribution $p$, and in our case $\phi(d) = d$. It is common in practice to use minibatch sampling to reduce the variance of ZO gradient estimates, where the average of $b$ i.i.d samples $\{\mathbf{u}_i\}_{i=1}^b$ drawn from $p$ are used for gradient estimation, leading to the multipoint estimate:

$$\hat{\nabla} f(\mathbf{x}) := \frac{\phi(d)}{\epsilon} \sum_{i=1}^{b} \left[ \left( f\left(\mathbf{x} + \epsilon \mathbf{u}_i\right) - f(\mathbf{x}) \right) \mathbf{u}_i \right] \tag{5}$$

Here for implementation convenience and computation saving, we carried out the experiment using a single random direction as Equation 4, while implementing the stochastic gradient descent (SGD) algorithm, the gradient was estimated on minibatches using this single direction. More specifically,

$$\hat{\nabla} f\left(\{\mathbf{x}_i\}_{i=1}^n\right) := \frac{\phi(d)}{\epsilon b} \left[ \sum_{i=1}^{b} \left( f\left(\mathbf{x}_i + \epsilon \mathbf{u}\right) - f\left(\mathbf{x}_i\right) \right) \right] \mathbf{u} \tag{6}$$

For ZO-SGD algorithm, the descent direction $\mathbf{m}_t$ is set as the current gradient estimate $\mathbf{m}_t = \widehat{g}_t$. In ZO-signSGD, $\mathbf{m}_t$ is given by the element-wise sign of the current gradient estimate, $\mathbf{m}_t = sign(\widehat{g}_t)$, which helps scaling down the (coordinate-wise) estimation errors [5].

## III. Experiments

We empirically compared the performance of FO-SGD, FO-signSGD, ZO-SGD, and ZO-signSGD, based on real-world handwritten digits classification dataset, MNIST.

### A. Setup

Two neural network configurations were implemented with architectures shown below, aiming to test the compatibility of the different optimizers with different complexity of the network. For computation efficiency, we will focus our performance analysis with respect to different hyperparameters on the simpler model, MyNet.

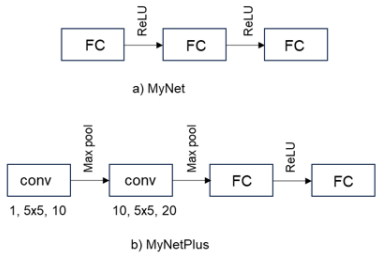


Fig. 1: Neural network architecture.

To search for the optimal parameter configuration for each optimizer with the MyNet model, we performed a grid search on learning rate, momentum for FO-SGD, and forward difference estimate parameter $\epsilon$ for ZO-SGD and ZO-signSGD. TABLE I lists the optimal configurations. The model was trained 4 times with each optimizer for 100 epochs, using a training batch size of 64. All optimizers were further compared at different learning rates aiming at understanding their training trajectories.

| Optimizer | Learning rate | Other |
|---|---|---|
| FO-SGD | 1e-2 | momentum=0.9 |
| FO-signSGD | 1e-4 | |
| ZO-SGD | 1e-2 | $\epsilon$=1e-3 |
| ZO-signSGD | 1e-3 | $\epsilon$=1e-3 |

TABLE I: Optimal configuration for ZO and FO optimizers

### B. Results

*1) Optimal configuration comparison:* Using the optimal hyperparameters found as TABLE I through grid search, the training loss and training accuracy of different optimizers over epoch are as shown in Fig. 2 (on model MyNet).

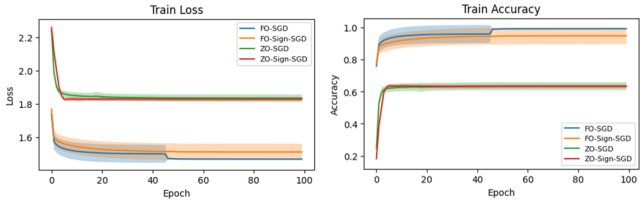


Fig. 2: Comparison of ZO and FO optimizers with optimal configurations.

Given the nature of the ZO optimizers which merely use estimates of the gradients, it is not rational to expect close accuracy to their FO counterparts. The maximum accuracy of FO optimizers reached around 0.98, while those for ZO optimizers were only around 0.63. Theoretically, the convergence rate of derivative-free methods can be quite competitive compared to first-order methods up to a slow-down factor of $\sqrt{d}$ [5]. This factor did not make a significant difference in our simple network configuration as we observed.

It is also notable that ZO optimizers seem to be giving more stable performance compared to their FO counterparts given the 4 optimal runs, which probably results from the landscape of the objective function. Since FO methods are using the actual gradient values, they tend to be more sensitive to the variance in the gradient, compared to ZO optimizers using only value-based evaluation.

*2) Same learning rate comparison:* Performance comparison was also conducted for different optimizers with same learning rates as shown in Fig. 3

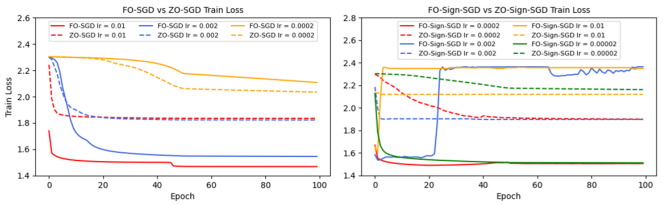


Fig. 3: Comparison of ZO and FO optimizers using same learning rates.

It can be observed that smaller learning rates were accompanied by slower convergence, which is in line with expectations. Further, the performance of the FO optimizers appeared to be more sensitive to changes in the learning rate. Excessive learning rates could result in overshooting for FO optimizers, which was not observed for ZO optimizers in the scope of our experiments. This could also be explained by FO's reliance on gradient information. Learning rate, or rather, step size plays a crucial role in FO optimizers convergence analysis given objective function assumptions. ZO optimizers on the other hand utilize perturbation techniques to estimate the gradients which inherently introduce a level of noise. This noise can act as a regularizer, making ZO optimizers more robust to learning rate changes. Another point to note here is that to achieve similar performance, sign-optimizers generally require lower learning rates, which may be caused by the loss of information on gradient magnitude.

*3) Model comparison:* The performance of ZO and FO optimizers were further evaluated on the MyNetPlus model with a learning rate of 2e-4. MyNetPlus has additional convolutional and max pooling layers compared with MyNet, resulting in increased complexity. The train loss of ZO optimizers showed no improvement over 100 epochs (see Fig. 6). This implies a strong performance dependence of ZO methods on the model architecture. They may struggle to escape a local

minima or find favorable ones when initiated far from the minima given their random search nature.

## IV. Further Analysis

As part of further analysis, we evaluated the performance of our optimizers in 2 different scenarios: During the scaling of our MyNet model complexity and the performance of a hybrid optimizer that combines first order and zero order optimization during different splits of training epochs.

### *A. FO vs ZO optimization dynamics in Neural Network up and down scaling of hidden layer complexity*

In this part, we ran the training of our MyNet neural network with ZO and FO optimization (with optimal configurations) while scaling the hidden layer neuron complexity by the following scaling factors of $[0.25, 0.5, 2]$. In terms of training accuracy, we observed relative values consistent with previous analysis where FO outperformed ZO optimization in all model complexity scales. We also observed that scaling the model by a factor of 2 resulted in much higher accuracy which is to be expected (Results in Appendix Fig. 7). In terms of epoch time efficiency, we also observed coherent values where ZO optimization was much slower than FO optimization and also that per optimization method, the time spent per epoch $t_e$ was proportional to the model complexity so $t_e \sim s$ (where $s$ is the scaling factor used for the NN hidden layers).

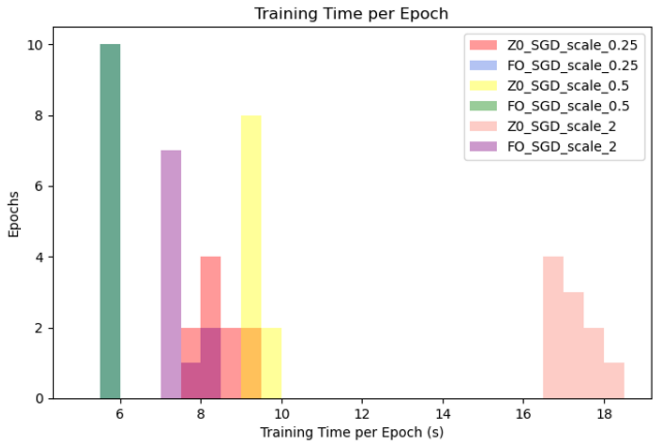


Fig. 4: Time efficency of ZO vs FO optimization with NN model scaling.

The interesting results we found were that the ratio for epoch time at different scales was different for different optimizers. In particular by taking the ratio of epoch time at $s = 2$ by epoch time at $s = 0.5$, we found that ZO optimization had a much higher ratio ($r = 1.83$) than FO optimization ($r = 1.27$) which we confirmed with a Student's $t$-test ($p = 1.86e - 13$). In other words for scaling factors $s_2 > s_1$:

$$\frac{t^z_{e,s_2}}{t^z_{e,s_1}} > \frac{t^f_{e,s_2}}{t^f_{e,s_1}}$$

With this, we can conclude that in our ML application ZO-SGD is much more sensitive in terms of time efficiency to model scaling than FO-SGD, and ZO-SGD struggles more with high dimensionality than FO-SGD.

### *B. Hybrid Optimizer approach*

Lastly, we can clearly see that in MNIST classification in terms of accuracy and training time efficiency FO-SGD clearly outperforms ZO-SGD. However in cases where gradients are computationally more expensive than gradient approximation, we can imagine that hybrid optimization could be a powerful tool for overcoming this hurdle. In this part, we observed the performance of a hybrid optimizer where for 20 epochs and different epoch splits ($[25\%, 75\%], [50\%, 50\%], [75\%, 25\%]$) we first performed ZO-SGD and then FO-SGD (named ZOFO hybrid optimizers and we also ran the experiments for FOZO optimizers in different epoch splits). The results we observed

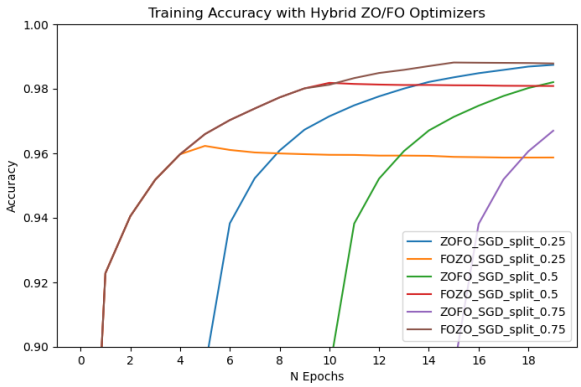


Fig. 5: Hybrid optimization for different epoch splits where "ZOFO SGD split 0.25" used ZO-SGD in 25% of training epochs and FO-SGD in the remaining 75%.

were that all of the hybrid optimizers reached very high accuracies of at least 0.95 at the end of training and also that by small margins ZOFO optimizers out performed FOZO optimizers. This validated the use of hybrid optimizers as an efficient alternative even in the case of 25% FO-SGD use. This also confirmed common theory that hybrid optimization works best by using zero order optimization first and then first order optimization.

## V. Conclusion

This paper compared the performance of ZO-SGD and ZO-signSGD with their FO counterparts on a fully connected neural network. ZO optimizers were outperformed by the FO optimizers under the optimal configurations. The performance of FO methods appeared more sensitive to learning rates, while ZO methods showed a strong reliance on model architecture and, in terms of time efficiency, to model scaling. Hybrid FO and ZO approach could be a promising tool for balancing gradient computation cost and performance.

## REFERENCES


[1] S. Liu, P.-Y. Chen, B. Kailkhura, G. Zhang, A. O. Hero, and P. K. Varshney, "A primer on zeroth-order optimization in signal processing and machine learning: Principals, recent advances, and applications," *IEEE Signal Processing Magazine*, vol. 37, no. 5, p. 43–54, 2020.

[2] L. Bottou and O. Bousquet, "The tradeoffs of large-scale learning," *Optimization for Machine Learning*, p. 351–368, 2011.

[3] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The rprop algorithm," *IEEE International Conference on Neural Networks*.

[4] S. Ghadimi and G. Lan, "Stochastic first- and zeroth-order methods for nonconvex stochastic programming," *SIAM Journal on Optimization*, vol. 23, no. 4, p. 2341–2368, 2013.

[5] S. Liu, P.-Y. Chen, X. Chen, and M. Hong, "signSGD via zeroth-order oracle," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=BJe-DsC5Fm


## APPENDIX

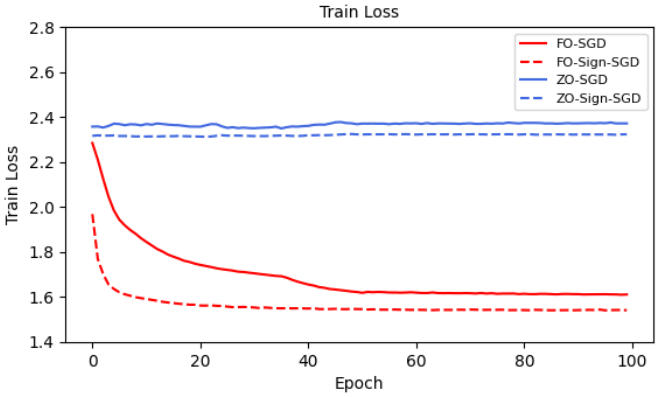


Fig. 6: Train loss of ZO and FO optimizers on MyNetPlus.

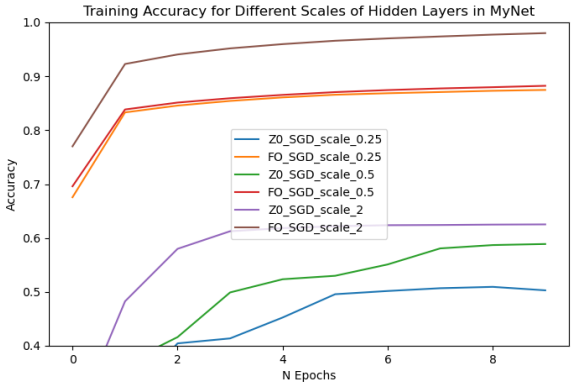


Fig. 7: Training Accuracy for model scaling.