

**Zeroth-Order Optimization:  
How to optimize without first-order gradient?**

- by finite-difference:  $\hat{\nabla}_{\theta} \ell(\theta) = \frac{1}{q} \sum_{i=1}^q \left[ \frac{\ell(\theta + \mu \mathbf{u}_i) - \ell(\theta)}{\mu} \mathbf{u}_i \right]$
- to tackle with black-box involved optimization, such as non-differentiable physical simulators[1].
- to unleash the potential of special hardware, such as ONNs[2] (optical neural networks).

**Gradient Estimator: CGE over RGE**

- CGE uses basis vectors while RGE uses random vectors.
- (RGE)  $\hat{\nabla}_{\theta} \ell(\theta) = \frac{1}{q} \sum_{i=1}^q \left[ \frac{\ell(\theta + \mu \mathbf{u}_i) - \ell(\theta)}{\mu} \mathbf{u}_i \right]$ ; (CGE)  $\hat{\nabla}_{\theta} \ell(\theta) = \sum_{i=1}^d \left[ \frac{\ell(\theta + \mu \mathbf{e}_i) - \ell(\theta)}{\mu} \mathbf{e}_i \right]$
- Experiments show that CGE outperforms RGE in terms of accuracy given the same query budget
- Runtime profiling exhibits CGE's efficiency merit over RGE at the same number of queries

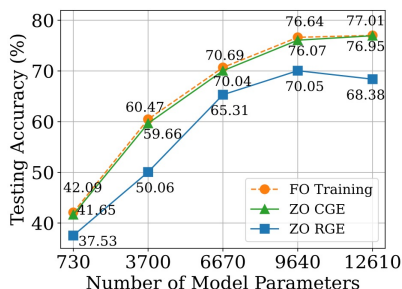


Figure 1. Performance comparison on a simple CNN with varying numbers of parameters on CIFAR-10.

	CGE	RGE
Accuracy (even $q = d$ )	😊	😞
Computation efficiency ( $q = d$ )	😊	😞
Query efficiency ( $q < d$ )	😊	😞

Table 1. Comparison between CGE and RGE from the perspective of accuracy, computation, and query efficiency.

Reference [1] Kiwon Um et al., Solver-in-the-loop: Learning from differentiable physics to interact with iterative pde-solvers. [2] Jiaqi Gu, et al., L2ight: Enabling on-chip learning for optical neural networks via efficient in-situ subspace optimization. [3] Chaoqi Wang et al, Picking winning tickets before training by preserving gradient flow.

**DeepZero: Sparse Gradients Guided by ZO Pruning**

- The disentanglement of weights within CGE is inherently pruning-friendly.
- We extend a pruning-at-initialization method, GraSP [3], to its ZO version:
 
$$\hat{\mathbf{S}} := -\theta \odot \frac{\hat{\nabla}_{\theta} \ell(\theta + \mu \hat{\mathbf{g}}) - \hat{\nabla}_{\theta} \ell(\theta)}{\mu}$$
- Guided by ZO-GraSP, we then introduce **dynamic sparse pattern**, leading to sparse ZO gradient estimates

**Algorithm 1 ZO-GraSP-Guided ZO Training**

- Get  $\mathcal{S}_{\text{ZO-GraSP}}$  through ZO-GraSP (3)
- Obtain layer-wise pruning ratio  $\mathcal{S}_{\text{layer}}$  based on  $\mathcal{S}_{\text{ZO-GraSP}}$
- for Epoch  $t = 0, 1, 2, \dots, T - 1$  do
- Randomly generate a sparse coordinate set  $\mathcal{S}_t$  according to  $\mathcal{S}_{\text{layer}}$
- for Iterations per epoch do
- Obtain (Sparse-CGE)  $\hat{\nabla}_{\theta} \ell(\theta)$  based on  $\mathcal{S}_t$
- Update model weights:  $\theta \leftarrow \theta - \alpha \hat{\nabla}_{\theta} \ell(\theta)$
- end for
- end for

**Acceleration: Feature Reuse & Forward Parallel**

**Feature Reuse:**  $f_{\theta}(\mathbf{x}) = f_{\theta_{>1}}(\mathbf{z}_1) = \underbrace{f_{\theta_{L_L}} \circ f_{\theta_{L-1}} \circ \dots \circ f_{\theta_{i+1}}}_{f_{\theta_{>1}(\cdot)}} \circ \underbrace{f_{\theta_i} \circ \dots \circ f_{\theta_1}}_{\mathbf{z}_1 = f_{\theta_{1:L}}(\mathbf{x})}(\mathbf{x})$

**Forward Parallel:**  $\hat{\nabla}_{\theta} \ell(\theta) = \sum_{i=1}^M \hat{\mathbf{g}}_i$ ;  $\hat{\mathbf{g}}_i := \sum_{j \in \mathcal{S}_i} \left[ \frac{\ell(\theta + \mu \mathbf{e}_j) - \ell(\theta)}{\mu} \mathbf{e}_j \right]$

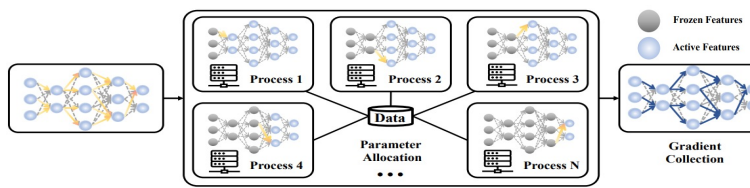


Figure 2. **Forward Parallel:** Queries are totally independent of each other and thus can be easily parallelized without any performance loss. **Feature Reuse:** one can reuse the feature immediately preceding the perturbed layer thanks to the minimum coordinate-wise perturbation by CGE.

**Experiments and Applications**

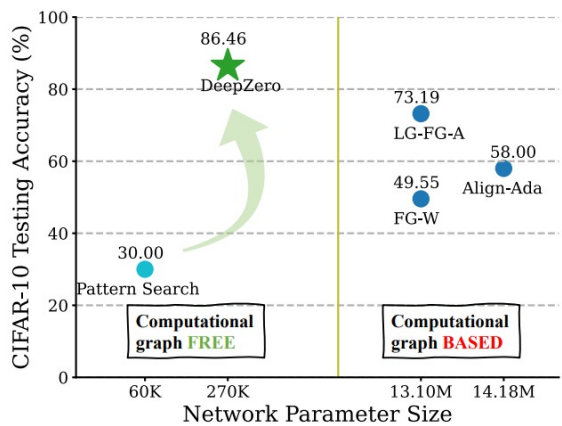


Figure 3. DeepZero outperforms the computational-graph-free baseline Pattern Search and computational-graph-dependent non-BP methods (ResNet-20, CIFAR10).

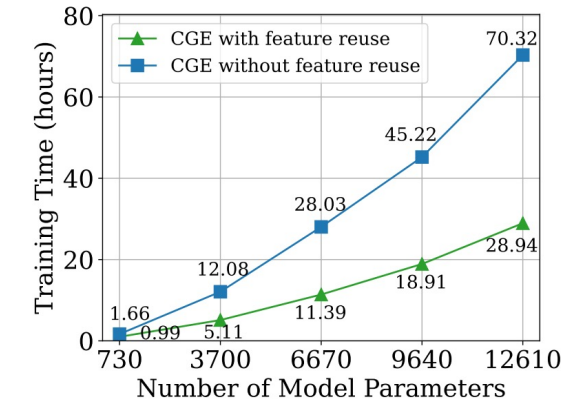


Figure 5. Computation cost of CGE-based ZO training w/ Feature Reuse vs. w/o Feature Reuse. Empirically, Feature Reuse can half the running time.

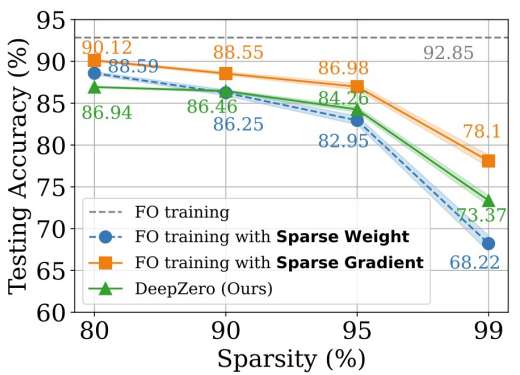


Figure 4. Comparison between DeepZero and FO training baselines on a ResNet-20 for CIFAR-10.

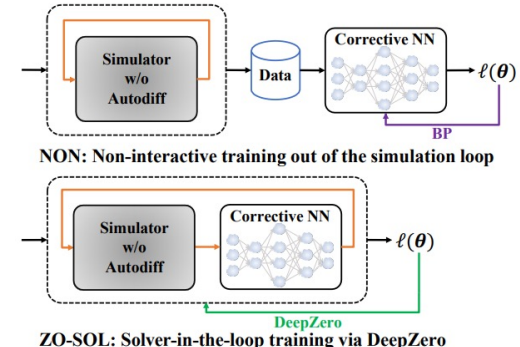


Figure 6. In physical simulation-involved tasks, DeepZero enables ‘solver-in-the-loop’ training for non-differentiable simulators.

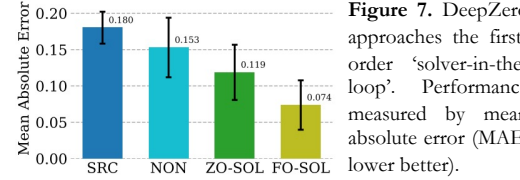


Figure 7. DeepZero approaches the first-order ‘solver-in-the-loop’. Performance measured by mean absolute error (MAE, lower better).