Variance-Reduced Zeroth-Order Optimization using SVRG-based ZO-AdaMM

Jincy P Janardhanan
Department of
Electrical Engineering
IIT Bombay
Email: 24m1075@iitb.ac.in

Abhinav P
Department of
Electrical Engineering
IIT Bombay
Email: 24d0547@iitb.ac.in

Arnab Mukherjee
Centre for
Machine Intelligence and Data Science
IIT Bombay
Email: 24d1597@iitb.ac.in

***Abstract*—This paper presents a variance-reduced zeroth-order optimization method, VR-ZO-AdaMM, which incorporates SVRG-style control variates into the ZO-AdaMM framework. By reducing the variance in gradient estimates, the method generates perturbations that more closely approximate the true gradient of the black-box loss. Compared to standard ZO-SGD and ZO-AdaMM, VR-ZO-AdaMM achieves the largest drop in model confidence while requiring far fewer queries, making it an efficient and effective choice for black-box attacks where high impact with minimal queries is desired.**

I. Introduction

Zeroth-order (ZO) optimization is a central framework for black-box optimization (BBO), where the objective function $f(\mathbf{x})$ is accessible only through function evaluations. Such settings appear in adversarial attacks on vision models, hyper-parameter tuning, scientific simulations, and optimization of non-differentiable or implicitly defined systems. Without analytic gradients, ZO methods rely on stochastic finite-difference estimators based on random directional queries.

A persistent challenge in ZO optimization is the high variance of these gradient estimators, which scales as $O(d)$ with problem dimension d. This variance substantially slows convergence and inflates query complexity. High-dimensional black-box learning problems—especially non-convex vision tasks—further amplify instability due to noisy gradients, heterogeneous coordinate curvature, and tight query budgets, making both robustness and variance control essential.

A. Existing Methods and Their Limitations

1) Basic and Deterministic Methods: Classical ZO-SGD and ZO-GD use simple randomized gradient estimators with fixed learning rates. Although foundational, they inherit full $O(d)$ variance and converge slowly in practice. Deterministic coordinate-descent variants reduce noise but require $O(d)$ queries per iteration, becoming impractical for modern high-dimensional problems such as image-based adversarial attacks.

2) Adaptive ZO Methods: Adaptive optimizers such as ZO-Adam and ZO-AdaMM significantly improve empirical stability by incorporating momentum and per-coordinate learning-rate scaling. These methods are widely used in practical black-box attacks because they navigate non-convex, ill-conditioned landscapes more effectively than basic ZO schemes. However, they operate directly on noisy ZO gradient estimates and do not fundamentally reduce their variance, leaving convergence limited by the $O(d)$ estimator noise.

3) Variance-Reduction Methods: Classical variance-reduction (VR) algorithms such as SVRG, SARAH, and SPIDER offer strong guarantees in first-order optimization. Their ZO counterparts (e.g., ZO-SVRG, ZO-SARAH) mimic these principles, but have seen limited adoption in high-dimensional vision-based black-box attacks. Recursive VR methods such as SARAH and SPIDER accumulate ZO noise across iterations, becoming unstable under high-dimensional stochastic estimators. Existing ZO-VR approaches also rely on fixed global learning rates, making them sensitive to ill-conditioning and incompatible with the adaptive scaling that practical ZO attacks depend on.

Consequently, despite their theoretical appeal, VR techniques have not become viable optimizers for large-scale black-box adversarial problems.

B. Proposed Approach: ZO-AdaMM-SVRG

To unify robustness and variance efficiency, we introduce *ZO-AdaMM-SVRG*, a hybrid algorithm that embeds an SVRG-style variance-reduction mechanism within the adaptive ZO-AdaMM optimizer.

AdaMM provides per-coordinate scaling and momentum, stabilizing updates under heterogeneous curvature and high estimator noise. SVRG introduces periodic reference-gradient surrogates that prevent noise accumulation and alleviate the $O(d)$ variance barrier. Unlike recursive VR methods, SVRG does not propagate estimator noise across inner iterations, making it well-suited for integration into adaptive ZO schemes.

The resulting method:

- reduces gradient-estimator variance without recursive noise amplification,
- retains the stability benefits of adaptive per-coordinate scaling,
- improves convergence speed under realistic query budgets,
- and remains robust in high-dimensional, non-convex black-box settings.

C. Motivation and Evaluation Setting

The absence of effective VR methods in practical black-box adversarial attacks highlights a gap between theoretical

ZO-VR algorithms and the optimizers used in modern applications. High-dimensional image attacks rely on NES-style estimators, bandit-based priors, and adaptive ZO optimizers, while recursive VR formulations remain unstable under noisy ZO gradients and tight query constraints.

SVRG offers a variance-reduction framework that avoids recursive error accumulation and is structurally compatible with adaptive optimizers. By integrating SVRG with AdaMM, we obtain a ZO method that is both variance-efficient and empirically stable—properties that prior ZO-VR algorithms could not simultaneously achieve.

We validate ZO-AdaMM-SVRG on black-box adversarial attacks for MNIST, CIFAR-10, and CIFAR-100 classifiers. These tasks exemplify high-dimensional, noisy, and strictly zeroth-order optimization, providing a rigorous testbed for assessing both query efficiency and stability.

II. Related Work

Black-box optimization has a long history in derivative-free methods and evolutionary strategies. Early approaches such as Nelder–Mead and CMA-ES [1], [2] laid foundational principles for adapting search distributions, though their query complexity scales poorly in high-dimensional settings such as image-based adversarial attacks. Randomized smoothing–based estimators, including Gaussian smoothing [3] and its NES-style variants [4], popularized stochastic directional derivatives as scalable tools for zeroth-order learning.

In adversarial machine learning, black-box attacks sparked renewed interest in efficient ZO optimization. The seminal boundary-based and score-based attacks of Chen et al. [5] and Ilyas et al. [6] used finite-difference estimators and NES sampling, showing that high-dimensional gradients can be approximated through a small number of random query directions. Follow-up work refined estimator structure using priors, temporal smoothing, and structured perturbations. Notable examples include AutoZOOM [7], which uses dimensionality reduction for query efficiency; P-RGF [8], which injects gradient priors from surrogate models; and BANDITS [9], which models gradient coordinates as reward distributions. These techniques improve empirical query complexity but do not fundamentally reduce the variance of stochastic gradient estimates.

Adaptive first-order optimizers such as Adam [10] inspired their zeroth-order counterparts. ZO-Adam and ZO-AdaGrad [11], [12] incorporate momentum, bias correction, and per-coordinate scaling, yielding significantly more stable trajectories in non-convex black-box problems. These adaptive ZO methods quickly became standard in practical adversarial attacks due to their robustness under noisy directions. Yet they operate directly on raw finite-difference gradients, leaving estimator variance—scaling as $O(d)$—unchanged.

Variance reduction is another major line of work. Classical VR algorithms such as SVRG [13], SAGA [14], SARAH [15], and SPIDER [16] achieve near-optimal gradient complexities in first-order stochastic optimization. Zeroth-order analogues were later proposed in works such as ZO-SVRG [17], ZO-SARAH [18], and ZO-SPIDER [19]. Although these algorithms inherit VR guarantees in expectation, their recursive formulations introduce instability when gradients are approximated via high-variance finite differences. This limitation has been documented both theoretically—where variance grows across inner loops—and empirically, where recursive ZO-VR often diverges in high-dimensional black-box adversarial settings.

Recent advances in ZO estimation strategies have attempted to address variance more directly. Two-point estimators [3], orthogonal or structured direction sampling [20], and antithetic sampling [21] all provide improvements in constant factors but not asymptotic dimension dependence. Hybrid ZO approaches using learned priors or model-based corrections [22] reduce noise empirically but remain brittle under tight query budgets.

Overall, the literature reveals two converging threads: adaptive ZO optimizers that are empirically strong but variance-limited, and VR-based ZO methods that are theoretically appealing yet empirically unstable in high-dimensional black-box environments. The lack of a method that simultaneously achieves adaptive stability and variance reduction motivates the integration of SVRG with an adaptive optimizer such as AdaMM, offering a path toward practical ZO variance reduction in adversarial settings.

A. Existing Methods

1) ZOO SGD: Zeroth-order SGD works by optimizing a function using only its output values, without ever needing explicit gradients. At each step, the method picks a random direction and checks how the function changes when the current point is nudged slightly forward and backward along that direction. The difference between these two values gives a rough sense of the slope, which is then scaled to form an estimate of the smoothed gradient. The algorithm updates the parameters by moving against this estimated slope and repeats the process over many iterations. With sensible choices of smoothing and step size, ZO-SGD ends up behaving very much like standard first-order SGD, even though it relies purely on black-box evaluations [23]. This method is present as Algorithm 1.

2) ZOO AdaMM: ZO-AdaMM (Zeroth-Order Adaptive Momentum Method) is basically Adam adapted for situations where you can't get real gradients. Instead of relying on derivatives, it feels out the landscape by sampling random directions and checking how the function changes when you nudge the current point slightly forward and backward. Those two evaluations give a rough idea of the slope, which then gets fed into Adam's usual moving-average updates. AMSGrad keeps the variance under control, and the final step scales the update according to how noisy each coordinate seems. The end result is an optimizer that behaves a lot like Adam, but works entirely from black-box queries—making it useful when gradients are unavailable or too expensive to [24]. This method is present as Algorithm 2.

Algorithm 1 Zeroth-Order Stochastic Gradient Descent (ZO-SGD)

```
Require: Initial point $x_0 \in \mathbb{R}^d$, smoothing parameter $\mu > 0$, step-size schedule $\{\eta_t\}$, number of iterations $T$
1: for $t = 0$ to $T - 1$ do
2:     Sample a random direction $u_t \sim \mathcal{N}(0, I_d)$
3:     Compute the two-point function evaluations:
           $f_+ = f(x_t + \mu u_t), \qquad f_- = f(x_t - \mu u_t)$
4:     Estimate gradient using the two-point ZO estimator:
           $g_t = \frac{f_+ - f_-}{2\mu} u_t$
5:     Update:
           $x_{t+1} = x_t - \eta_t g_t$
6: end for
7: return $x_T$
```

Algorithm 2 ZO-AdaMM

```
Require: initial point $x_0$, smoothing parameter $\mu$, number of directions $q$, step sizes $\{\alpha_t\}$, Adam parameters $\beta_1, \beta_2$, small $\varepsilon > 0$, total iterations $T$
1: Initialize $m_0 = \mathbf{0}$,  $v_0 = v_{\text{init}}\mathbf{1}$,  $\hat{v}_0 = v_0$
2: for $t = 0, 1, \ldots, T - 1$ do
3:     Sample $q$ random directions $u_{t,1}, \ldots, u_{t,q}$ (unit normalized)
4:     Compute two-point ZO gradient estimator:
           $\hat{g}_t = \frac{d}{q} \sum_{i=1}^{q} \frac{f(x_t + \mu u_{t,i}) - f(x_t - \mu u_{t,i})}{2\mu} u_{t,i}$
5:     First moment update:
           $m_{t+1} = \beta_1 m_t + (1 - \beta_1) \hat{g}_t$
6:     Second moment update:
           $v_{t+1} = \beta_2 v_t + (1 - \beta_2) \hat{g}_t^2$
7:     AMSGrad correction:
           $\hat{v}_{t+1} = \max(\hat{v}_t,\, v_{t+1})$
8:     Parameter update:
           $x_{t+1} = x_t - \alpha_t \frac{m_{t+1}}{\sqrt{\hat{v}_{t+1}} + \varepsilon}$
9: end for
10: return $x_T$
```

III. Proposed Method

In this section, we describe the proposed Zeroth-Order Adaptive Moment Method with Stochastic Variance Reduction and AMSGrad correction (ZO-AdaMM-SVRG). The method aims to optimize a smooth, potentially nonconvex objective function $f : \mathbb{R}^d \to \mathbb{R}$ using only function evaluations. The algorithm pseudocode is presented in Algorithm 3.

A. Gaussian Smoothing Gradient Estimation

Given a smoothing parameter $\mu > 0$, the algorithm approximates the gradient of f using two-point Gaussian smoothing. For an input x and random direction $u \sim \mathcal{N}(0, I_d)$ normalized to unit length, the zeroth-order (ZO) gradient estimator is

$$g_\mu(x; u) = \frac{d}{2\mu} \big(f(x + \mu u) - f(x - \mu u) \big) u. \tag{1}$$

To reduce variance, the method averages q such estimators at each inner-loop step. At the beginning of every outer loop (epoch), the method computes a high-accuracy reference gradient using a larger batch of Q directions:

$$\tilde{g}_s = \frac{d}{Q} \sum_{i=1}^{Q} \frac{f(x_{s,0} + \mu u_i) - f(x_{s,0} - \mu u_i)}{2\mu} u_i. \tag{2}$$

This reference gradient acts as a control variate for the variance-reduced estimator.

B. Variance Reduction

Within each epoch, for the current iterate $x_{s,t}$ and using the same mini-batch directions $u_{t,1}, \ldots, u_{t,q}$, the algorithm computes:

$$\hat{g}_t = \frac{d}{q} \sum_{i=1}^{q} \frac{f(x_{s,t} + \mu u_{t,i}) - f(x_{s,t} - \mu u_{t,i})}{2\mu} u_{t,i}, \tag{3}$$

$$\hat{g}_t^{\text{ref}} = \frac{d}{q} \sum_{i=1}^{q} \frac{f(x_{s,0} + \mu u_{t,i}) - f(x_{s,0} - \mu u_{t,i})}{2\mu} u_{t,i}. \tag{4}$$

The resulting variance-reduced gradient estimator is

$$g_{s,t}^{\text{VR}} = \hat{g}_t - \hat{g}_t^{\text{ref}} + \tilde{g}_s. \tag{5}$$

This estimator preserves unbiasedness while significantly reducing stochastic fluctuations, improving the stability of the adaptive updates.

C. Adaptive Moment Estimation

The method incorporates first- and second-moment estimates following the Adam structure. Given hyperparameters $\beta_1, \beta_2 \in (0, 1)$, the moment updates are

$$m_{s,t+1} = \beta_1 m_{s,t} + (1 - \beta_1)\, g_{s,t}^{\text{VR}}, \tag{6}$$

$$v_{s,t+1} = \beta_2 v_{s,t} + (1 - \beta_2) \left(g_{s,t}^{\text{VR}} \right)^2. \tag{7}$$

To ensure stable convergence, we employ the AMSGrad correction:

$$\hat{v}_{s,t+1} = \max\{\hat{v}_{s,t},\, v_{s,t+1}\}. \tag{8}$$

The adaptive parameter update is then computed as

$$x_{s,t+1} = x_{s,t} - \alpha_{s,t} \frac{m_{s,t+1}}{\sqrt{\hat{v}_{s,t+1}} + \varepsilon}, \tag{9}$$

where $\alpha_{s,t}$ is a prescribed learning-rate schedule and $\varepsilon > 0$ ensures numerical stability.

D. Outer and Inner Loop Structure

The algorithm proceeds in epochs indexed by $s = 0, \ldots, S-1$. Each epoch begins at $x_{s,0}$, computes the reference gradient $\tilde{g}_s$, and resets the moment estimates. The inner loop performs m adaptive updates using the variance-reduced ZO gradient. At the end of the epoch, the next epoch's starting point is set to $x_{s+1} = x_{s,m}$. If the optimization is constrained to a set $\mathcal{X}$, each update is followed by a projection $x_{s,t+1} \leftarrow \Pi_{\mathcal{X}}(x_{s,t+1})$.

The final output of the method is the last outer-loop iterate x_S.

IV. Experimental Setup

The experiments compares three algorithm setups by measuring their average query counts, average drop in model confidence, and overall attack success rates.

Updating the Perturbation Using the Estimated Gradient

Let the perturbation be defined as

$$\delta = x - x_0.$$

Once the estimated gradient $\hat{g}(x)$ is obtained, the perturbation is updated via

$$\delta_{t+1} = \delta_t - \alpha\, \hat{g}(x).$$

The updated perturbation is then projected onto the valid perturbation set (e.g., an L_2 ball of radius ϵ):

$$\delta_{t+1} = \mathrm{Proj}_{\|\delta\| \le \epsilon}(\delta_{t+1}).$$

Finally, the adversarial input is reconstructed as

$$x_{t+1} = x_0 + \delta_{t+1}.$$

This update rule is identical for all three zeroth-order attack methods; only the gradient estimation differs.

A. Experiment 1

We trained a classification model that uses the Wisconsin Breast Cancer dataset, which contains 569 samples described by 30 continuous diagnostic features such as radius, texture, perimeter, and smoothness. Each sample is labeled as either malignant or benign. After standardizing all features, a simple fully connected neural network is trained as the target classifier. The model consists of an input layer matching the 30-dimensional feature vector, followed by two hidden layers with 64 and 32 units and ReLU activations, plus dropout for regularization, and a final sigmoid output predicting the probability of a benign tumor. This network serves as the black-box model that the optimization methods attempt to attack. Then done the black box attack on the above model.

Algorithm 3 ZO-AdaMM-SVRG

```
Require: Initial point $x_0$, smoothing parameter $\mu$, inner loop length $m$, number of random directions $q$, learning rate schedule $\{\alpha_t\}$, Adam parameters $\beta_1, \beta_2$, small $\varepsilon > 0$.
 1: Initialize $m_0 = 0,\ v_0 = v_{\text{init}}\mathbf{1},\ \hat{v}_0 = v_0$.
 2: for $s = 0, 1, 2, \ldots, S-1$ do
 3:     $x_{s,0} = x_s$
 4:     Compute a reference gradient (large batch):
            $$\tilde{g}_s = \frac{d}{Q}\sum_{i=1}^{Q} \frac{f(x_{s,0}+\mu u_i) - f(x_{s,0}-\mu u_i)}{2\mu} u_i,$$
        where $u_i \sim \mathcal{N}(0, I_d)$ normalized.
 5:     Set $m_{s,0} = 0,\ v_{s,0} = v_{\text{init}}\mathbf{1},\ \hat{v}_{s,0} = v_{s,0}$.
 6:     for $t = 0, 1, \ldots, m-1$ do
 7:         Sample $q$ random directions $u_{t,1}, \ldots, u_{t,q}$.
 8:         Compute ZO gradient at current point:
                $$\hat{g}_t = \frac{d}{q}\sum_{i=1}^{q} \frac{f(x_{s,t}+\mu u_{t,i}) - f(x_{s,t}-\mu u_{t,i})}{2\mu} u_{t,i}.$$
 9:         Compute ZO gradient at reference point:
                $$\hat{g}_t^{\text{ref}} = \frac{d}{q}\sum_{i=1}^{q} \frac{f(x_{s,0}+\mu u_{t,i}) - f(x_{s,0}-\mu u_{t,i})}{2\mu} u_{t,i}.$$
10:         Variance–Reduced ZO gradient:
                $$g_{s,t}^{\text{VR}} = \hat{g}_t - \hat{g}_t^{\text{ref}} + \tilde{g}_s.$$
11:         Adam first moment:
                $$m_{s,t+1} = \beta_1 m_{s,t} + (1-\beta_1) g_{s,t}^{\text{VR}}.$$
12:         Adam second moment:
                $$v_{s,t+1} = \beta_2 v_{s,t} + (1-\beta_2)\big(g_{s,t}^{\text{VR}}\big)^2.$$
13:         AMSGrad correction:
                $$\hat{v}_{s,t+1} = \max(\hat{v}_{s,t}, v_{s,t+1}).$$
14:         Adaptive update:
                $$x_{s,t+1} = x_{s,t} - \alpha_{s,t}\frac{m_{s,t+1}}{\sqrt{\hat{v}_{s,t+1}} + \varepsilon}.$$
15:         Projection (if constrained):
                $$x_{s,t+1} \leftarrow \Pi_{\mathcal{X}}(x_{s,t+1}).$$
16:     end for
17:     Set $x_{s+1} = x_{s,m}$.
18: end for
19: return Final iterate $x_S$.
```

B. Experiment 2

We use a standard CNN classifier for MNIST, following architectures commonly used in zeroth-order adversarial attack literature. The model contains two convolutional layers with 32 and 64 filters (kernel size 3×3), each followed by ReLU and 2×2 max-pooling, and two fully connected layers with

128 hidden units and 10 output logits. Images are normalized to $[0, 1]$. The network is trained using Adam with learning rate 10^{-3}, batch size 128, for 10 epochs using cross-entropy loss. No data augmentation is applied. The model achieves over 99% test accuracy and serves as a standard baseline for black-box attacks.

We use a modified ResNet-18 adapted for CIFAR-10 by replacing the first (7×7) convolution with a (3×3) layer, removing the max-pooling block, and using a 10-class fully connected output. Training uses SGD with momentum 0.9, learning rate 0.1, weight decay (5×10^{-4}), and a `MultiStepLR` decay at epochs 60, 120, and 160. Standard CIFAR-10 augmentation (random crop and horizontal flip) and channel-wise normalization are applied. The model is trained for up to 200 epochs with early stopping, batch size 128, and cross-entropy loss. The final test accuracy is approximately **93–95%**.

C. *Experiment 3*

We use a pretrained WideResNet-28 model, which is a widened variant of ResNet designed to improve representation capacity by increasing the number of channels instead of depth. The "28" denotes the network's depth, and the widened residual blocks allow more expressive feature learning. This model is fine-tuned on the CIFAR-100 dataset using standard preprocessing. After fine-tuning, correctly classified test samples are selected, and the black-box adversarial evaluation is performed

D. *Evaluation Metrics*

The following metrics used for evaluation,

- **Average Queries:** The mean number of model evaluations required by the attack to achieve misclassification.
- **Average Confidence Drop:** The average reduction in the model's predicted confidence for the true class after the perturbation is applied.
- **Perturbation Norm:** The magnitude of the adversarial modification (e.g., L_2), indicating how much the input was altered.
- **Attack Success Rate:** The fraction of test samples for which the attack successfully caused misclassification.

V. Results

A. *Experiment 1*

Trained neural networks details given below.

Metric	Value
Accuracy	0.9708
Precision	0.9811
Recall	0.9720
F1-Score	0.9765

TABLE I
Neural Network Performance Metrics for Experiment 1

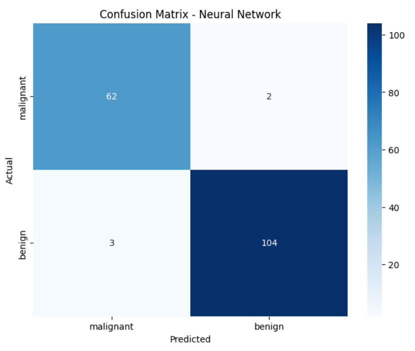

Fig. 1. Confusion matrix of the trained neural network - Experiment 1.

Black box attack results are given below.

Method	Success	Avg Queries	Avg Conf Drop
ZO-AdaMM	75	114	-0.3497
ZO-SGD	79	2796	-0.4046
VR-ZO-AdaMM	78	510	-0.3681

TABLE II
Attack performance comparison across methods Experiment 1.

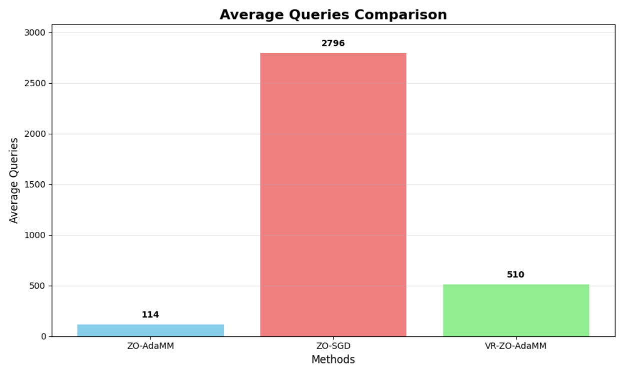

Fig. 2. Average query count comparison across attack methods in Experiment 1.

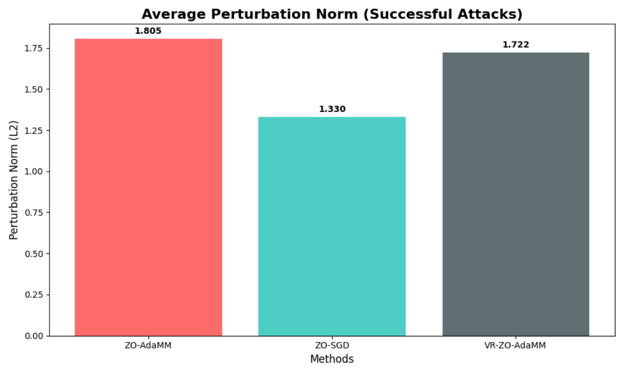

Fig. 3. Average perturbation magnitude across attack methods in Experiment 1.

B. *Experiment 2*

The CNN model for MNIST achieves over 99% test accuracy, while the modified ResNet-18 for CIFAR-10 achieves approximately 93–95% test accuracy. The results of Black box attack on MNIST and CIFAR 10 Model are given below.

Method	Success	Perturbation	Avg Queries
ZO-AdaMM SVRG (MNIST)	0.999	6.5	1177
ZO-AdaMM (MNIST)	0.999	7.2	220
ZO-AdaMM SVRG (CIFAR-10)	0.967	32.0	1524
ZO-AdaMM (CIFAR-10)	0.705	28.9	587

TABLE III
PERFORMANCE OF ZO-ADAMM VARIANTS ON MNIST AND CIFAR-10.

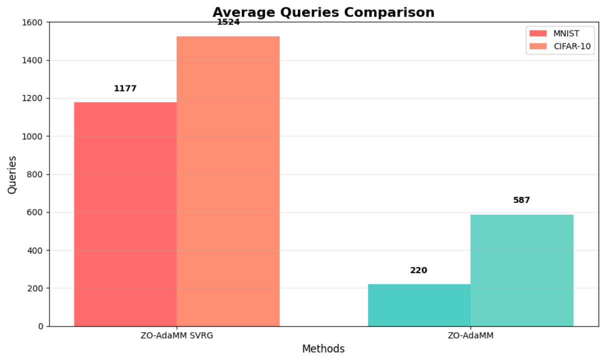

Fig. 4. Average queries for Experiment 2.

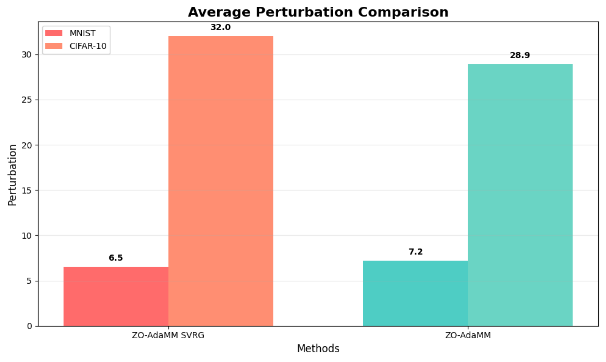

Fig. 5. Average perturbations for Experiment 2.

C. Experiment 3

The original SOTA WRN-28-10 achieved 80.75% accuracy on CIFAR-100[25], and our fine-tuned model reached a comparable level, confirming correct replication. CIFAR-100 was chosen over CIFAR-10 because its higher class diversity offers a more challenging and informative setting for evaluating black-box adversarial attacks.

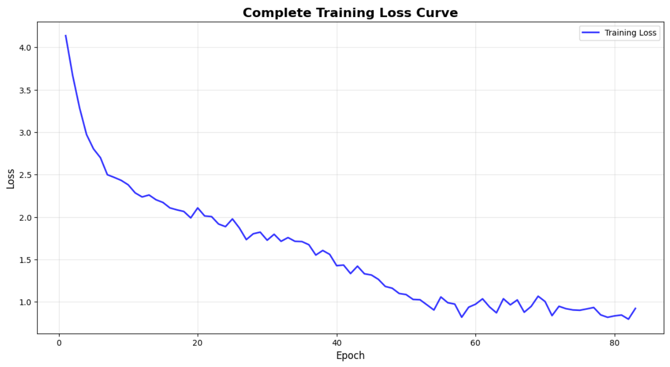

Fig. 6. Loss curve during fine-tuning of the model.

Black box attack results are given below. we considered a subsample data of few due to limited resources.

TABLE IV
COMPARISON OF ATTACK PERFORMANCE ACROSS METHODS

Method	**Success Rate**	**Avg Queries**	**Avg Conf Drop**
ZO-AdaMM	100.0%	106	0.2768
ZO-SGD	60.0%	1710	0.1984
VR-ZO-AdaMM	100.0%	201	0.3612

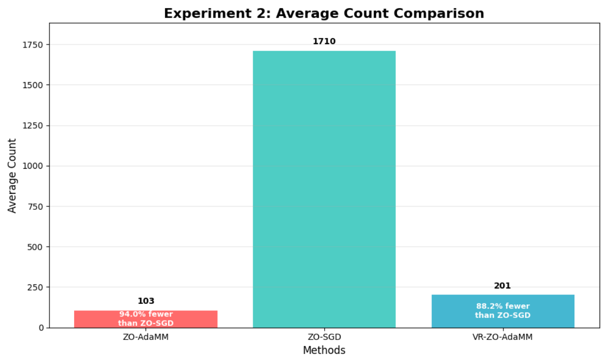

Fig. 7. Average number of queries required by Experiment 3.

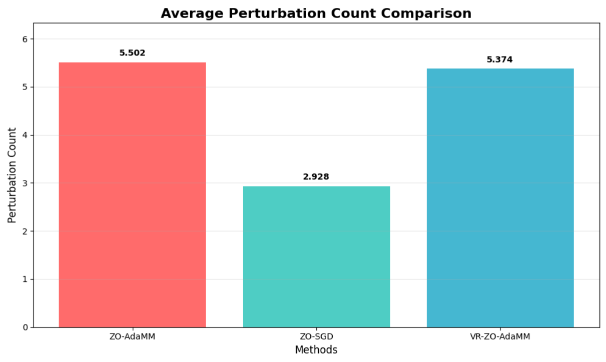

Fig. 8. Average perturbation for Experiment 3.

VI. CONCLUSION

Among the three black-box attack methods we studied, ZO-SGD tends to require a very large number of queries to succeed. This happens because its gradient estimates are noisy, so it needs to probe the model many times to reliably figure out which way to perturb the input.

ZO-AdaMM improves on this by using adaptive learning rates and momentum, which stabilizes the updates and significantly reduces the number of queries compared to ZO-SGD.

The SVRG variant of ZO-AdaMM takes this a step further. By reducing the variance in the gradient estimates, it produces updates that are much closer to the true gradient of the black-box loss. This not only achieves the largest drop in model confidence among all methods but also keeps the query count relatively low.

In short, VR-ZO-AdaMM is an ideal choice for situations where you want a strong attack—maximizing confidence drop—while keeping the number of queries to a minimum. It hits the sweet spot between effectiveness and efficiency.

References

[1] N. Hansen and A. Ostermeier, "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es)," *Evolutionary Computation*, vol. 11, no. 1, pp. 1–18, 2003.

[2] N. Hansen, "The cma evolution strategy: A tutorial," in *arXiv*, 2016.

[3] Y. Nesterov and V. Spokoiny, "Random gradient-free minimization of convex functions," *Foundations of Computational Mathematics*, vol. 17, no. 2, pp. 527–566, 2017.

[4] T. Salimans, J. Ho, X. Chen, and I. Sutskever, "Evolution strategies as a scalable alternative to reinforcement learning," in *ICLR*, 2017.

[5] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh, "Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models," in *ACM AsiaCCS*, 2017, pp. 15–26.

[6] A. Ilyas, L. Engstrom, A. Athalye, and J. Lin, "Black-box adversarial attacks with limited queries and information," in *ICML*, 2018, pp. 2152–2161.

[7] C.-C. Tu, P.-H. Ting, P.-Y. Liu, H. Zhang, J. Ye, and C.-J. Hsieh, "Autozoom: Autoencoder-based zeroth order optimization method for attacking black-box neural networks," in *AAAI*, 2019, pp. 742–749.

[8] M. Cheng, T. Le, P.-Y. Chen, J. Yi, H. Zhang, and C.-J. Hsieh, "Improving black-box adversarial attacks with a transfer-based prior," in *NeurIPS*, 2019.

[9] A. Ilyas, L. Engstrom, and A. Madry, "Prior convictions: Black-box adversarial attacks with bandits and priors," in *ICLR*, 2019.

[10] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv*, 2014.

[11] S. Liu, T. Xu, C. Jin, and L. Yang, "Zeroth-order online convex optimization: Frank-wolfe and gradient descent with variance reduction," in *NeurIPS*, 2018.

[12] S. Liu, M. Cheng, Y. Han, H. Zhang, C. Gan, and C.-J. Hsieh, "A practical zeroth-order optimization method for deep model training," in *ICML*, 2020, pp. 6326–6335.

[13] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," in *NeurIPS*, 2013.

[14] A. Defazio, F. Bach, and S. Lacoste-Julien, "Saga: A fast incremental gradient method with support for non-strongly convex composite objectives," in *NeurIPS*, 2014.

[15] L. M. Nguyen, J. Liu, K. Scheinberg, and M. Takác, "Sarah: A novel method for machine learning problems using stochastic recursive gradient," in *ICML*, 2017, pp. 2613–2621.

[16] C. Fang, C. J. Li, and Z. Lin, "Spider: Near-optimal non-convex optimization via stochastic path-integrated differential estimator," in *NeurIPS*, 2018.

[17] J. Liu and J. Huang, "Zeroth-order stochastic variance reduction for nonconvex optimization," in *ICML*, 2017, pp. 2152–2161.

[18] K. Ji, Y. Wang, and Y. Zhou, "Improved zeroth-order variance reduced methods with fast convergence," in *AAAI*, 2019.

[19] C. Fang, C. J. Li, Z. Lin, and T. Zhang, "Spider-boost and zeroth-order spider," in *arXiv*, 2018.

[20] K. Choromanski, A. Pacchiano, J. Parker-Holder, Y. Tang, A. Pacchiano, T. Sarlos, and M. I. Jordan, "From complexity to simplicity: Adaptive es-active subspaces for blackbox optimization," in *ICML*, 2019.

[21] Y. Feng, K. Balasubramanian, and P. Zhang, "Antithetic sampling for efficient zeroth-order optimization," *Journal of Machine Learning Research*, vol. 23, no. 89, pp. 1–34, 2022.

[22] L. Lee, N. Awad, A. Rajeswaran, O. Nachum, and S. Levine, "Model-based gradient-free reinforcement learning with exploration," in *NeurIPS*, 2020.

[23] X. Liu, L. Chen, Z. Wang, and Q. Wang, "General stability analysis for zeroth-order optimization algorithms," in *Proceedings of the 12th International Conference on Learning Representations*, 2024.

[24] X. Chen, S. Liu, K. Xu, X. Li, X. Lin, M. Hong, and D. Cox, "Zo-adamm: Zeroth-order adaptive momentum method for black-box optimization," in *Advances in Neural Information Processing Systems (NeurIPS 2019)*, 2019.

[25] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146*, 2016.

Appendix A
Black-Box Adversarial Attack: Margin-Based Objective

This section provides a brief description of the margin-based black-box adversarial attack used in our experiments. The attack assumes no access to model gradients and relies solely on model queries via a prediction function $\mathrm{Predict}(\cdot)$.

Given a base input x_0 with true label y, the goal is to find a bounded perturbation δ such that the perturbed input $\tilde{x} = x_0 + \delta$ causes the classifier to reduce confidence in the true label relative to the most likely incorrect label. The attack is driven by the following objective:

$$f(\delta) = M(\delta) + \lambda D(\delta),$$

where $M(\delta)$ is the classification margin and $D(\delta)$ is the distortion penalty.

The margin term encourages misclassification:

$$M(\delta) = \log(p_{\text{true}} + 10^{-12}) - \log(p_{\text{other}} + 10^{-12}),$$

with p_{true} denoting the model's predicted probability for the true class and p_{other} the maximum probability among all incorrect classes. The small constant prevents numerical instability.

The distortion term is defined as

$$D(\delta) = \|\delta\|^2,$$

ensuring that the perturbation remains small unless needed for successful misclassification. A projection operator $\Pi_{[l,u]}$ enforces element-wise bounds on δ.

The attack incorporates several early stopping conditions, including reaching a desired margin threshold, exceeding a distortion limit, lack of improvement over a patience window, plateauing of a moving average, or exhausting a preset query budget. These mechanisms improve efficiency and prevent unnecessary model queries.

Algorithm 4 Black-Box Untargeted Attack: Margin-Based Objective

```
Require: Base image $x_0 \in \mathbb{R}^d$, true label $y$, perturbation $\delta$,
    model query function $\mathrm{Predict}(\cdot)$, distortion weight $\lambda$.
 1: function Loss($\delta$)
 2:     $\tilde{x} \leftarrow x_0 + \delta$
 3:     $p \leftarrow \mathrm{softmax}(\mathrm{Predict}(\tilde{x}))$
 4:     $p_{\text{true}} \leftarrow p_y$
 5:     $p_{\text{other}} \leftarrow \max_{j \neq y} p_j$
 6:     margin:
            $M(\delta) = \log(p_{\text{true}} + 10^{-12}) - \log(p_{\text{other}} + 10^{-12})$
 7:     distortion:
            $D(\delta) = \|\delta\|^2$
 8:     return
            $f(\delta) = M(\delta) + \lambda D(\delta)$
 9: end function
10: function Project($\delta$)
11:     return
            $\Pi_{[l,\, u]}(\delta)$
12: end function
    Early Stopping Conditions:
13: function CheckStop($f(\delta)$)
14:     if $f(\delta) <$ margin_threshold then
15:         return "margin reached"
16:     else if $\|\delta\| >$ distortion_limit then
17:         return "distortion exceeded"
18:     else if no improvement for $T_{\mathrm{patience}}$ steps then
19:         return "patience exhausted"
20:     else if moving average plateaus then
21:         return "flat region"
22:     else if queries exceed budget then
23:         return "query budget exhausted"
24:     else
25:         return False
26:     end if
27: end function
```