Natural Perturbations for Black-box Training of Neural Networks by Zeroth-Order Optimization

Hiroshi Sawada [1] **Kazuo Aoyama** [1] **Yuya Hikima** [1]

Abstract

This paper proposes a novel concept of natural perturbations for black-box training of neural networks by zeroth-order optimization. When a neural network is implemented directly in hardware, training its parameters by backpropagation ends up with an inaccurate result due to the lack of detailed internal information. We instead employ zeroth-order optimization, where the sampling of parameter perturbations is of great importance. The sampling strategy we propose maximizes the entropy of perturbations with a regularization that the probability distribution conditioned by the neural network does not change drastically, by inheriting the concept of natural gradient. Experimental results show the superiority of our proposal on diverse datasets, tasks, and architectures.

1. Introduction

The backpropagation algorithm (Rumelhart et al., 1986; Amari, 1993; LeCun et al., 2015) is a standard technique for training the parameters of a neural network on CPU or GPU-based ordinary computers. It computes the gradient vectors with respect to the parameters by backpropagating the gradient information from the loss function defined at the output. For the computation, it is necessary to obtain detailed internal information, such as the derivatives of component-wise activation functions and layer-wise inputs to neurons generated during the forward computation. On the other hand, when neural networks are implemented directly in hardware (Figure 1), such as optical neural networks (Shen et al., 2017; Ashtiani et al., 2022) and analog-based in-memory computing (Li et al., 2018; Aguirre et al., 2024), training the parameters by backpropagation ends up with an inaccurate result. This is because detailed internal information cannot be accurately obtained even when the hardware is simulated by an ordinary computer, for example, due to hardware-specific manufacturing variations (Fang et al., 2019; Banerjee et al., 2023).

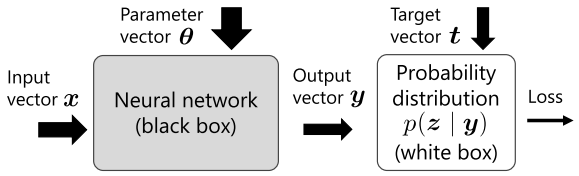

Figure 1. Training neural network implemented in hardware

For such above cases, several black-box optimization methods have been used to train the parameters of neural networks on hardware, including genetic algorithm (Zhang et al., 2019; 2021), bacterial foraging optimization (Cong et al., 2022), covariance matrix adaptation evolution strategy (CMA-ES) (Chen et al., 2022; Lupo et al., 2023), and zeroth-order (ZO) optimization (Shen et al., 2017; Zhou et al., 2020; Gu et al., 2020; 2021; Bandyopadhyay et al., 2022). Among these methods, this paper focuses on ZO optimization (Liu et al., 2020) because it is a local search based on (approximate) gradient vectors like backpropagation and is expected to scale to training neural networks with a moderately large number N of parameters.

The core operation of the ZO optimization for computing approximate gradient vectors is to slightly perturb the parameter vector $\boldsymbol{\theta} \in \mathbb{R}^N$ to $\boldsymbol{\theta} + \mu \boldsymbol{u}$ with a smoothing hyperparameter $\mu > 0$ and a perturbation vector $\boldsymbol{u} \in \mathbb{R}^N$, and perform a query to evaluate how the loss function changes from $\ell(\boldsymbol{\theta})$ to $\ell(\boldsymbol{\theta} + \mu \boldsymbol{u})$. Here, the perturbation vector $\boldsymbol{u}$ is sampled from some distribution, typically from a multivariate standard normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$ with an identity matrix $\mathbf{I}$ (Duchi et al., 2015; Nesterov & Spokoiny, 2017) or similarly from a uniform distribution on a unit sphere (Fazel et al., 2018). While these sampling strategies are plausible for a shallow neural network, they are not the best way for a deep neural network because the parameters are correlated in the optimization landscape. The reason for the correlation is that many parameters are involved in a path from an input element to an output element of the neural network (Sawada et al., 2025). To mitigate the correlation of parameters, the use of coordinate-wise perturbations (Lian

[1]NTT Communication Science Laboratories, NTT Corporation, Kyoto, Japan. Correspondence to: Hiroshi Sawada <sawada.hiroshi@ieee.org>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

What to maximize \ Regularization by	Parameter Space Discrepancy (PSD)	+ Function Space Discrepancy (FSD)
Loss decrease (steepest descent)	Gradient descent	Natural gradient descent
Distribution entropy (exploration)	Perturbations from $\mathcal{N}(\mathbf{0}, \mathbf{I})$	**Ours:** *Natural perturbations*

Figure 2. Criteria for methods computing gradient or perturbations

et al., 2016; Berahas et al., 2022; Chen et al., 2024) is effective, where a perturbation vector $\boldsymbol{u}$ has only one non-zero element. However, the disadvantage is that it requires as many queries as the number of parameters to change the entire parameters.

Returning to the situation where we can use backpropagation, the natural gradient method (Amari, 1998; Pascanu & Bengio, 2013; Martens, 2020) is known to be effective especially when the parameters are correlated. It modifies the gradient vector computed by backpropagation based on the Fisher information matrix (FIM), which describes how the parameters are correlated. Regarding ZO optimization, some methods (Zhao et al., 2020; Sawada et al., 2024) have been proposed to introduce the idea of natural gradient. Although these methods modify the ZO gradient vector based on the natural gradient criterion, they do not suggest an appropriate distribution from which to sample perturbations.

This paper proposes a novel concept of *natural perturbations* and a way to design the distribution from which we sample them. The term "natural" refers to the same meaning as natural gradient, but the criterion of natural perturbations is different and newly introduced in this paper. As summarized in Figure 2, natural gradient descent aims at the steepest descent direction by taking care not only of the parameter space discrepancy (PSD), which is the only regularization in gradient descent, but also of the function space discrepancy (FSD) defined with the FIM, as will be detailed with (5). Analogously, natural perturbations are sampled by considering not only the PSDs but also the FSDs. What to maximize in this sampling strategy is the entropy of the distribution, making the sampled perturbations explore as widely as possible. As will be shown in Section 3.1, the existing sampling strategy from $\mathcal{N}(\mathbf{0}, \mathbf{I})$ can be considered to maximize the entropy only under the PSD regularization.

Our proposed method and natural evolution strategies (NES) (Wierstra et al., 2014) are both black-box optimization methods that use the FIM. However, they are fundamentally different as described in Appendix B.

For neural networks with a large number N of parameters, computing and inverting the FIM introduced above is practically prohibited because the matrix size is N^2. For natural gradient, sophisticated approximation methods (Martens & Grosse, 2015; Benzing, 2022) have been proposed, which require detailed internal information and are therefore not applicable to our black-box ZO optimization method. Instead, we propose to simply partition a large parameter set into small blocks. This makes perturbation vectors block coordinate and thus the FIM that we need block diagonal, whose inversion can be computed practically.

This work builds on (Sawada et al., 2025), which points out that the parameters of an optical neural network are layered and thus correlated, and proposes a method to perturb the parameters of a linear module. The advancement of this work over the previous work is substantial as follows.

- The concept of natural perturbations is newly proposed as Figure 2 shows clear relations to the existing methods (gradient or perturbation and natural or not).
- A black-box method to compute FIMs is newly proposed in Section 4.1, whereas the previous work needs to simulate the internal information, which may be inaccurate.
- The new ZO optimization method can efficiently handle convolutional neural networks (CNNs) and recurrent neural networks (RNNs), which the previous work cannot do because FIMs are computed in a module-wise manner.

The rest of this paper is organized as follows. Section 2 formulates a neural network, and explains natural gradient and ZO optimization. We then newly propose *natural perturbations* in Section 3, and describe a practical method for ZO optimization using them in Section 4. Section 5 shows the experimental results to demonstrate the effectiveness of our proposal. Section 6 concludes the paper.

2. Preliminaries

2.1. Neural Network and Loss Function

As shown in Figure 1, we consider a neural network $\boldsymbol{f}$ with a parameter vector $\boldsymbol{\theta} \in \mathbb{R}^N$ that transforms an input vector $\boldsymbol{x} \in \mathbb{R}^K$ to an output vector $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta}) \in \mathbb{R}^M$. A probabilistic interpretation can be given that the neural network expresses a conditional distribution $p_{\boldsymbol{\theta}}(\boldsymbol{z} \mid \boldsymbol{x}) = p(\boldsymbol{z} \mid \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta}))$ with a probability distribution $p(\boldsymbol{z} \mid \boldsymbol{y})$ defined at the output. Given a training dataset $\mathcal{D}_{\mathrm{tr}} = \{(\boldsymbol{x}, \boldsymbol{t})\}$, each element of which is a pair of an input vector $\boldsymbol{x}$ and a target vector $\boldsymbol{t}$, the loss function for mini-batch $\mathcal{D} \subseteq \mathcal{D}_{\mathrm{tr}}$ can be defined as

$$\ell_{\mathcal{D}}(\boldsymbol{\theta}) := -\mathbb{E}_{(\boldsymbol{x},\boldsymbol{t})\sim\mathcal{D}}[\ln p(\boldsymbol{t} \mid \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta}))] \ , \tag{1}$$

where $\ln$ represents the natural logarithm, i.e., $\ln = \log_e$.

2.2. Natural Gradient

For a small change $\delta\boldsymbol{\theta} \in \mathbb{R}^N$ of $\boldsymbol{\theta}$, we measure how the conditional distribution changes using the Kullback–Leibler (KL) divergence $d_{\mathrm{KL}}(p_{\boldsymbol{\theta}}(\boldsymbol{z} \mid \boldsymbol{x}), p_{\boldsymbol{\theta}+\delta\boldsymbol{\theta}}(\boldsymbol{z} \mid \boldsymbol{x}))$. The function space discrepancy (FSD) is the expected value of the

measurements over a subset $\mathcal{D}$ of the training dataset

$$\xi_{\mathcal{D}}(\delta\boldsymbol{\theta}) := \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[d_{\mathrm{KL}}(p_{\boldsymbol{\theta}}(\boldsymbol{z}\,|\,\boldsymbol{x}), p_{\boldsymbol{\theta}+\delta\boldsymbol{\theta}}(\boldsymbol{z}\,|\,\boldsymbol{x}))] \,. \quad (2)$$

Note that the target vectors $\boldsymbol{t}$ are not used here (Kunstner et al., 2019). Let us approximate the above by the Taylor series expansion of d_{KL} up to the second order (Pascanu & Bengio, 2013; Martens, 2016). Then it is given as

$$\xi_{\mathcal{D}}(\delta\boldsymbol{\theta}) \approx \frac{1}{2}\,\delta\boldsymbol{\theta}^{\mathsf{T}}\mathbf{F}_{\boldsymbol{\theta}}\,\delta\boldsymbol{\theta} = \frac{1}{2}\mathrm{tr}(\mathbf{F}_{\boldsymbol{\theta}}\delta\boldsymbol{\theta}\delta\boldsymbol{\theta}^{\mathsf{T}}) \quad (3)$$

with the Fisher information matrix (FIM)

$$\mathbf{F}_{\boldsymbol{\theta}} := \mathbb{E}_{\boldsymbol{x}}\left[\mathbb{E}_{\boldsymbol{z}}\left[\nabla_{\boldsymbol{\theta}} \ln p_{\boldsymbol{\theta}}(\boldsymbol{z}\,|\,\boldsymbol{x})\, \nabla_{\boldsymbol{\theta}} \ln p_{\boldsymbol{\theta}}(\boldsymbol{z}\,|\,\boldsymbol{x})^{\mathsf{T}}\right]\right], \quad (4)$$

where the expectation notations are in concise forms as $\mathbb{E}_{\boldsymbol{x}}[\cdot] = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[\cdot]$ and $\mathbb{E}_{\boldsymbol{z}}[\cdot] = \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})}[\cdot]$.

With a parameter update $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \delta\boldsymbol{\theta}$ in mind, natural gradient descent as well as ordinary gradient descent can be formulated as the minimization problem (Bae et al., 2022)

$$\delta\boldsymbol{\theta} \leftarrow \arg\min_{\delta\boldsymbol{\theta}} \{\ell_{\mathcal{D}}(\boldsymbol{\theta}+\delta\boldsymbol{\theta}) + r\} \;\text{ with} \quad (5)$$

$$r := \frac{\lambda_P^{(\mathrm{NG})}}{2\eta}\|\delta\boldsymbol{\theta}\|_2^2 + \frac{\lambda_F^{(\mathrm{NG})}}{\eta}\xi_{\mathcal{D}}(\delta\boldsymbol{\theta}) \,, \quad (6)$$

where $\|\delta\boldsymbol{\theta}\|_2^2/2$ is the parameter space discrepancy (PSD) defined by the squared l_2-norm, and η serves as a learning rate hyperparameter. The PSD and FSD are weighted by hyperparameters $\lambda_P^{(\mathrm{NG})} > 0$ and $\lambda_F^{(\mathrm{NG})} \geq 0$. By solving (5) with the first-order approximation of the loss term $\ell_{\mathcal{D}}(\boldsymbol{\theta} + \delta\boldsymbol{\theta}) \approx \ell_{\mathcal{D}}(\boldsymbol{\theta}) + \delta\boldsymbol{\theta}^{\mathsf{T}}\nabla_{\boldsymbol{\theta}}\ell_{\mathcal{D}}(\boldsymbol{\theta})$ and (3), we have

$$\delta\boldsymbol{\theta} = -\eta \cdot \left(\lambda_P^{(\mathrm{NG})} \cdot \mathbf{I} + \lambda_F^{(\mathrm{NG})} \cdot \mathbf{F}_{\boldsymbol{\theta}}\right)^{-1} \nabla_{\boldsymbol{\theta}}\ell_{\mathcal{D}}(\boldsymbol{\theta}) \,. \quad (7)$$

If $\lambda_F^{(\mathrm{NG})} = 0$, (7) reduces to ordinary gradient descent. Natural gradient corresponds to when $\lambda_F^{(\mathrm{NG})} > 0$. A sufficiently large value of $\lambda_P^{(\mathrm{NG})}$, e.g., $\lambda_P^{(\mathrm{NG})} = 1$, ensures that the matrix to be inverted is full rank, even if the computed FIM is not full rank.

The term $\nabla_{\boldsymbol{\theta}}\ell_{\mathcal{D}}(\boldsymbol{\theta})$ in (7) is the gradient vector for the loss function with respect to the parameters, and typically computed by backpropagation. However, when neural networks are implemented directly in hardware, we cannot use backpropagation and need to rely on a black-box method. In this paper, we compute approximate gradient vectors by ZO optimization explained in the next subsection.

2.3. Zeroth-Order (ZO) Optimization

ZO optimization is a black-box optimization method for function $\ell_{\mathcal{D}}(\boldsymbol{\theta})$ that we do not have full access to. Specifically, we can query a function value $\ell_{\mathcal{D}}(\boldsymbol{\theta})$, but we cannot compute the gradient $\nabla_{\boldsymbol{\theta}}\ell_{\mathcal{D}}(\boldsymbol{\theta})$ precisely. Still, we can approximate the gradient by

$$\hat{\nabla}_{\boldsymbol{\theta}}^{(\mathbf{I})}\ell_{\mathcal{D}}(\boldsymbol{\theta}) := \mathbb{E}_{\boldsymbol{u}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left[\frac{\ell_{\mathcal{D}}(\boldsymbol{\theta}+\mu\boldsymbol{u}) - \ell_{\mathcal{D}}(\boldsymbol{\theta})}{\mu}\boldsymbol{u}\right] , \quad (8)$$

which is derived from Gaussian smoothing of $\ell_{\mathcal{D}}(\boldsymbol{\theta})$ with a multivariate standard normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$ and a smoothing hyperparameter $\mu > 0$ (Nesterov & Spokoiny, 2017; Berahas et al., 2022).

Algorithm 1 ZO optimization for training a neural network

Input: training dataset $\mathcal{D}_{\mathrm{tr}} = \{(\boldsymbol{x}, \boldsymbol{t})\}$, hyperparameters Q, μ
Initialize the parameter vector $\boldsymbol{\theta}$
while not converged **do**
  Sample mini-batch $\mathcal{D}$ from $\mathcal{D}_{\mathrm{tr}}$
  Evaluate loss $\ell_{\mathcal{D}}(\boldsymbol{\theta})$ by (1)
  Sample perturbations $\boldsymbol{u}_q$, $q = 1, \ldots, Q$ from $\mathcal{N}(\mathbf{0}, \mathbf{I})$
  Compute the approximate gradient $\hat{\nabla}_{\boldsymbol{\theta}}\ell_{\mathcal{D}}(\boldsymbol{\theta})$ by (10)
  Update the parameter vector $\boldsymbol{\theta}$ by (9)
end while

Algorithm 1 shows a procedure for training a neural network with ZO optimization. For each mini-batch $\mathcal{D}$, we update the parameter vector $\boldsymbol{\theta}$ by

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \cdot \hat{\nabla}_{\boldsymbol{\theta}}\ell_{\mathcal{D}}(\boldsymbol{\theta}) \quad (9)$$

with a realization of the approximate gradient (8)

$$\hat{\nabla}_{\boldsymbol{\theta}}\ell_{\mathcal{D}}(\boldsymbol{\theta}) := \frac{1}{Q}\sum_{q=1}^{Q}\frac{\ell_{\mathcal{D}}(\boldsymbol{\theta}+\mu\boldsymbol{u}_q) - \ell_{\mathcal{D}}(\boldsymbol{\theta})}{\mu}\boldsymbol{u}_q \,, \quad (10)$$

where Q perturbation vectors $\boldsymbol{u}_q \in \mathbb{R}^N$, $q = 1, \ldots, Q$ are sampled from $\mathcal{N}(\mathbf{0}, \mathbf{I})$. The query consumption cost for (10) is $Q + 1$, one for $\ell_{\mathcal{D}}(\boldsymbol{\theta})$ and Q for $\ell_{\mathcal{D}}(\boldsymbol{\theta} + \mu\boldsymbol{u}_q)$.

As discussed in Section 1, if the neural network has a deep structure, the sampling strategy for perturbation vectors $\boldsymbol{u}_q$ from $\mathcal{N}(\mathbf{0}, \mathbf{I})$ is not the best way because it ignores the parameter correlation represented by the FIM. One way to mitigate the parameter correlation is to sample coordinate-wise perturbations with one-hot vectors

$$\boldsymbol{u}_q = \boldsymbol{e}_{n_q} := [0, \ldots, 1\,(n_q\text{-th entry}), \ldots, 0]^{\mathsf{T}} \in \mathbb{R}^N \,, \quad (11)$$

where indices n_q, $q = 1, \ldots, Q$ are sampled from the set $\{1, \ldots, N\}$ without replacement. Indeed, the coordinate-wise sampling strategy is often more effective than that from $\mathcal{N}(\mathbf{0}, \mathbf{I})$, as shown by the experimental results in Section 5. However, it consumes as many queries as N to change the entire parameter elements because there is only one non-zero element in the perturbation vector.

3. Natural Perturbations

Aiming to solve the two issues just mentioned, namely mitigating the parameter correlation and having multiple non-zero elements in a perturbation vector, we propose a novel strategy that samples *natural perturbations*.

3.1. Criterion and Main Result

We consider that a good sampling strategy for ZO optimization is to somehow maximize the entropy of the distribution so that the sampled perturbations $\boldsymbol{u}$ explore as widely as possible (Figure 2). To simplify the discussion, we restrict the distributions to N-dimensional multivariate normal distributions $\mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$ with an arbitrary positive definite covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{N \times N}$. Then, the entropy of random vectors $\boldsymbol{u}$ drawn from $\mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$ is given as

$$h(\boldsymbol{\Sigma}) := \mathbb{E}_{\boldsymbol{u} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})}[-\ln p(\boldsymbol{u})] = \frac{1}{2}\left[N \ln(2\pi) + \ln\det \boldsymbol{\Sigma} + N\right] . \quad (12)$$

Analogously to (5), the sampling strategy for *natural perturbations* $\boldsymbol{u} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$ is formulated as minimizing the criterion

$$\mathcal{C}(\boldsymbol{\Sigma}) := -h(\boldsymbol{\Sigma}) + \frac{\lambda_P}{2}\mathbb{E}_{\boldsymbol{u}}\left[\|\boldsymbol{u}\|_2^2\right] + \lambda_F \, \mathbb{E}_{\boldsymbol{u}}[\xi_{\mathcal{D}}(\boldsymbol{u})] \ , \quad (13)$$

where the expectation is written concisely as $\mathbb{E}_{\boldsymbol{u}}[\cdot] = \mathbb{E}_{\boldsymbol{u} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})}[\cdot]$. The first term is the negative entropy. The second term with a weight hyperparameter $\lambda_P > 0$ is the expected PSD, and the third term with a weight hyperparameter $\lambda_F \geq 0$ is the expected FSD.

Let us derive the covariance matrix that minimizes the criterion (13). The last two terms, without the weight hyperparameters, can be written as

$$\frac{1}{2}\mathbb{E}_{\boldsymbol{u}}\left[\|\boldsymbol{u}\|_2^2\right] = \frac{1}{2}\mathrm{tr}(\mathbb{E}_{\boldsymbol{u}}\left[\boldsymbol{u}\boldsymbol{u}^\mathsf{T}\right]) = \frac{1}{2}\mathrm{tr}(\boldsymbol{\Sigma}) , \quad (14)$$

$$\mathbb{E}_{\boldsymbol{u}}[\xi_{\mathcal{D}}(\boldsymbol{u})] \overset{(3)}{\approx} \frac{1}{2}\mathrm{tr}(\mathbf{F}_{\boldsymbol{\theta}}\mathbb{E}_{\boldsymbol{u}}\left[\boldsymbol{u}\boldsymbol{u}^\mathsf{T}\right]) = \frac{1}{2}\mathrm{tr}(\mathbf{F}_{\boldsymbol{\theta}}\boldsymbol{\Sigma}) , \quad (15)$$

because $\mathbb{E}_{\boldsymbol{u}}\left[\boldsymbol{u}\boldsymbol{u}^\mathsf{T}\right] = \boldsymbol{\Sigma}$ by definition. By ignoring constant terms in (12) that do not depend on $\boldsymbol{\Sigma}$, (13) becomes

$$\mathcal{C}(\boldsymbol{\Sigma}) \overset{\mathrm{c}}{=} -\frac{1}{2}\ln\det\boldsymbol{\Sigma} + \frac{\lambda_P}{2}\mathrm{tr}(\boldsymbol{\Sigma}) + \frac{\lambda_F}{2}\mathrm{tr}(\mathbf{F}_{\boldsymbol{\theta}}\boldsymbol{\Sigma}) . \quad (16)$$

The gradient matrix with respect to $\boldsymbol{\Sigma}$ is given by

$$\frac{\partial \mathcal{C}(\boldsymbol{\Sigma})}{\partial \boldsymbol{\Sigma}} = -\frac{1}{2}\boldsymbol{\Sigma}^{-1} + \frac{\lambda_P}{2}\mathbf{I} + \frac{\lambda_F}{2}\mathbf{F}_{\boldsymbol{\theta}} . \quad (17)$$

Setting this to a zero matrix gives

$$\boldsymbol{\Sigma} = \left(\lambda_P \cdot \mathbf{I} + \lambda_F \cdot \mathbf{F}_{\boldsymbol{\theta}}\right)^{-1} \quad (18)$$

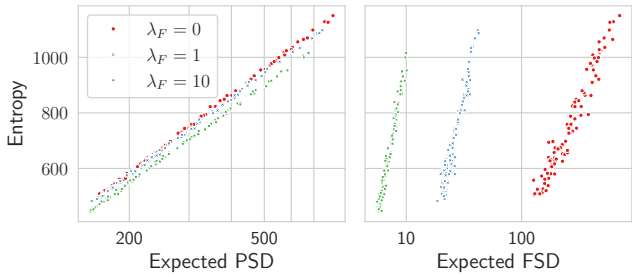

Figure 3. Experimental examples on the relations among the entropy, the expected PSD, and the expected FSD.

as the optimal $\boldsymbol{\Sigma}$ that minimizes $\mathcal{C}(\boldsymbol{\Sigma})$.

From (13) and (18), we understand that the existing sampling strategy from $\mathcal{N}(\mathbf{0}, \mathbf{I})$ is a special case with $\lambda_P = 1$ and $\lambda_F = 0$, and takes into account the expected PSD (14) but not the expected FSD (15). In contrast, our proposed sampling strategy takes into account the expected FSD as well by setting $\lambda_F > 0$.

3.2. Metrics

To see how well the term $\lambda_F \cdot \mathbf{F}_{\boldsymbol{\theta}}$ in (18) works with $\lambda_F > 0$, let us experimentally investigate the relations among the three terms in the criterion (13), namely the entropy (12), the expected PSD (14), and the expected FSD (15).

We trained a small-size CNN having $N = 816$ parameters with the MNIST handwritten digits (LeCun & Cortes, 2010). We stopped the training at the 5-th epoch to see a situation on the way (the test accuracy was still 0.662). We examined three values for λ_F as shown by colors/styles in Figure 3. For λ_P, we examined various values ranging from 1 to 5 so that the tendencies can be observed visually with many dots. Note that the values of λ_P are not visible in Figure 3. For each pair of λ_F and λ_P values, we computed $\boldsymbol{\Sigma}$ as (18) and sampled $Q = 20$ perturbations $\boldsymbol{u}_q$ from $\mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$. The expected PSD (14) and FSD (15) were actually calculated by the averages over Q perturbations:

$$\frac{1}{2}\mathbb{E}_{\boldsymbol{u}}\left[\|\boldsymbol{u}\|_2^2\right] \approx \frac{1}{2}\frac{1}{Q}\sum_{q=1}^{Q}\|\boldsymbol{u}_q\|_2^2 , \quad (19)$$

$$\mathbb{E}_{\boldsymbol{u}}[\xi_{\mathcal{D}}(\boldsymbol{u})] \approx \frac{1}{2}\frac{1}{Q}\sum_{q=1}^{Q}\boldsymbol{u}_q^\mathsf{T}\mathbf{F}_{\boldsymbol{\theta}}\boldsymbol{u}_q . \quad (20)$$

We observe the followings in Figure 3. The existing sampling strategy with $\lambda_F = 0$ attained the largest entropy under the same expected PSD (see left hand plot vertically). However, this was at a cost with the largest expected FSD for attaining the same entropy among the three λ_F values (see the right hand plot horizontally). Our sampling strategy for natural perturbations with $\lambda_F > 0$ (blue and green symbols),

on the other hand, reduced the expected FSD for attaining the same entropy (see the right hand plot horizontally) by paying a little cost of reducing the entropy (see left hand plot vertically) under the same expected PSD.

Too large an FSD leads to unstable training because the probability distribution conditioned by the neural network changes drastically. Natural perturbations as well as natural gradient prevent such situations.

3.3. Theoretical Property

This subsection shows the approximation error bound of the ZO gradient approximation

$$\hat{\nabla}^{(\mathrm{NP})}_{\boldsymbol{\theta}}\ell_{\mathcal{D}}(\boldsymbol{\theta}) := \mathbb{E}_{\boldsymbol{u}\sim\mathcal{N}(\mathbf{0},\boldsymbol{\Sigma})}\left[\frac{\ell_{\mathcal{D}}(\boldsymbol{\theta}+\mu\boldsymbol{u})-\ell_{\mathcal{D}}(\boldsymbol{\theta})}{\mu}\boldsymbol{u}\right] \tag{21}$$

by natural perturbations $\boldsymbol{u}\sim\mathcal{N}(\mathbf{0},\boldsymbol{\Sigma})$ with $\boldsymbol{\Sigma}$ being (18), to the natural gradient

$$\nabla^{(\mathrm{NG})}_{\boldsymbol{\theta}}\ell_{\mathcal{D}}(\boldsymbol{\theta}) := \left(\lambda_P^{(\mathrm{NG})}\cdot\mathbf{I}+\lambda_F^{(\mathrm{NG})}\cdot\mathbf{F}_{\boldsymbol{\theta}}\right)^{-1}\nabla_{\boldsymbol{\theta}}\ell_{\mathcal{D}}(\boldsymbol{\theta}) \tag{22}$$

extracted from (7).

Assumption 3.1. $\ell_{\mathcal{D}}(\boldsymbol{\theta})$ is L-smooth, i.e.,

$$\ell_{\mathcal{D}}(\boldsymbol{\theta}+\mu\boldsymbol{u}) \le \ell_{\mathcal{D}}(\boldsymbol{\theta})+\nabla\ell_{\mathcal{D}}(\boldsymbol{\theta})^{\mathsf{T}}(\mu\boldsymbol{u})+\frac{L}{2}\|\mu\boldsymbol{u}\|_2^2\,, \tag{23}$$

for any $\boldsymbol{\theta}\in\mathbb{R}^N$, $\boldsymbol{u}\in\mathbb{R}^N$, and $\mu>0$.

Theorem 3.2. *When we assume $\lambda_P^{(\mathrm{NG})}=\lambda_P$, $\lambda_F^{(\mathrm{NG})}=\lambda_F$ and Assumption 3.1, the difference between the ZO gradient approximation by natural perturbations and the natural gradient is bounded by*

$$\left\|\hat{\nabla}^{(\mathrm{NP})}_{\boldsymbol{\theta}}\ell_{\mathcal{D}}(\boldsymbol{\theta})-\nabla^{(\mathrm{NG})}_{\boldsymbol{\theta}}\ell_{\mathcal{D}}(\boldsymbol{\theta})\right\|_2 \le \frac{\mu L}{2}\left(\frac{3+N}{\lambda_P}\right)^{3/2}. \tag{24}$$

The proof is shown in Appendix A.

Therefore, the ZO gradient approximation by natural perturbations well approximates the natural gradient as long as the smoothing hyperparameter μ is small enough and the PSD weight λ_P is not very small. We actually set $\mu=0.001$ and $\lambda_P=1$ in the experiments reported in Section 5.2.

4. Method for ZO Optimization

To practically use *natural perturbations* for ZO optimization of neural networks with many parameters, we need to overcome two related issues. The first is how to compute the FIM $\mathbf{F}_{\boldsymbol{\theta}}$ in a black-box manner. The second is then how to avoid the inverse (18) of a big matrix including $\mathbf{F}_{\boldsymbol{\theta}}$. The second issue is managed by partitioning the parameters into small blocks, as explained in Section 4.2. But let us start this section with the FIM computation, assuming that the number N of parameters is not so large.

4.1. FIM Computation

The derived result (18) for sampling natural perturbations requires to compute the FIM $\mathbf{F}_{\boldsymbol{\theta}}$ as in the case of natural gradient. If we are allowed to perform backpropagation, there are ways (Dangel et al., 2020; Martens, 2020) to approximately compute the FIM efficiently. However, in our black-box context, we need to develop another practical way to compute the FIM, as this subsection proposes.

The FIM $\mathbf{F}_{\boldsymbol{\theta}}$ defined in (4) can be expressed (Park et al., 2000; Martens, 2020) as

$$\mathbf{F}_{\boldsymbol{\theta}} = \mathbb{E}_{\boldsymbol{x}}\left[\mathbf{J}_{\boldsymbol{y}\boldsymbol{\theta}}^{\mathsf{T}}\mathbf{F}_{\boldsymbol{y}}\mathbf{J}_{\boldsymbol{y}\boldsymbol{\theta}}\big|_{\boldsymbol{y}=\boldsymbol{f}(\boldsymbol{x},\boldsymbol{\theta})}\right]\in\mathbb{R}^{N\times N} \tag{25}$$

with a collection of Jacobian matrices

$$\mathbf{J}_{\boldsymbol{y}\boldsymbol{\theta}} := \frac{\partial\boldsymbol{y}}{\partial\boldsymbol{\theta}} = \frac{\partial\boldsymbol{f}(\boldsymbol{x},\boldsymbol{\theta})}{\partial\boldsymbol{\theta}}\in\mathbb{R}^{M\times N} \tag{26}$$

and a collection of the FIMs with respect to the output

$$\mathbf{F}_{\boldsymbol{y}} := \mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z}|\boldsymbol{y})}\left[\nabla_{\boldsymbol{y}}\ln p(\boldsymbol{z}\,|\,\boldsymbol{y})\,\nabla_{\boldsymbol{y}}\ln p(\boldsymbol{z}\,|\,\boldsymbol{y})^{\mathsf{T}}\right]\in\mathbb{R}^{M\times M}. \tag{27}$$

4.1.1. White-Box Output FIM Computation

The computation of $\mathbf{F}_{\boldsymbol{y}}$ can be performed in a white-box manner because we have complete information on the probability distribution $p(\boldsymbol{z}\,|\,\boldsymbol{y})$ used at the output (see Figure 1).

When we use a multivariate standard normal distribution as $p(\boldsymbol{z}\,|\,\boldsymbol{y})$ for a regression task, the output FIM is simply given as $\mathbf{F}_{\boldsymbol{y}}=\mathbf{I}$ with an identity matrix (Malagò & Pistone, 2015). When we use a categorical distribution (a multinomial distribution for a single observation) as $p(\boldsymbol{z}\,|\,\boldsymbol{y})$ with logits $[\boldsymbol{y}]_m$, $m=1,\dots,M$, for a classification task, the output FIM can be computed (Sawada et al., 2024) by

$$\mathbf{F}_{\boldsymbol{y}} = \sum_{\boldsymbol{z}\in\mathcal{T}} p(\boldsymbol{z}\,|\,\boldsymbol{y})\nabla_{\boldsymbol{y}}\ln p(\boldsymbol{z}\,|\,\boldsymbol{y})\,\nabla_{\boldsymbol{y}}\ln p(\boldsymbol{z}\,|\,\boldsymbol{y})^{\mathsf{T}}, \tag{28}$$

where $\mathcal{T}=\{[1,0,\dots,0]^{\mathsf{T}},\dots,[0,\dots,0,1]^{\mathsf{T}}\}$ is a set of all possible outcomes, and

$$\nabla_{\boldsymbol{y}}\ln p(\boldsymbol{z}\,|\,\boldsymbol{y}) = \boldsymbol{z}-softmax(\boldsymbol{y}), \tag{29}$$

$$[softmax(\boldsymbol{y})]_m = \frac{\exp([\boldsymbol{y}]_m)}{\sum_{k=1}^M\exp([\boldsymbol{y}]_k)}. \tag{30}$$

4.1.2. Black-Box Jacobian Computation

The computation of the Jacobian matrices $\mathbf{J}_{\boldsymbol{y}\boldsymbol{\theta}}$ should be performed in a black-box manner. We adopt a column-wise

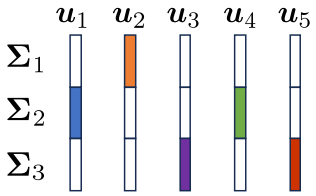

Figure 4. Block coordinate perturbations when $B = 3$ and $Q = 5$. Colored partitioned vectors are sampled using $\boldsymbol{\Sigma}_b$ and white partitioned vectors are filled with zero.

(parameter-wise) computation

$$\mathbf{J}_{\boldsymbol{y}\boldsymbol{\theta}} = [\delta \boldsymbol{y}_1, \dots, \delta \boldsymbol{y}_N] \,, \tag{31}$$

and each column is computed by evaluating function values for multivariate difference quotients

$$\delta \boldsymbol{y}_n := \frac{\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta} + \mu \boldsymbol{e}_n) - \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta})}{\mu} \in \mathbb{R}^M \tag{32}$$

with coordinate-wise perturbations using one-hot vectors $\boldsymbol{e}_n := [0, \dots, 1\,(n\text{-th entry}), \dots, 0]^\mathsf{T} \in \mathbb{R}^N$, $n = 1, \dots, N$.

Assuming that $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta})$ is already evaluated, the query consumption cost is N for the computation of $\mathbf{J}_{\boldsymbol{y}\boldsymbol{\theta}}$. This cost is large for a large number N of parameters. We will explain how to reduce the query cost in the rest of Section 4.

4.2. Partitioning Parameters into Small Blocks

When the number of parameters N is large, the inverse computation (18) for the covariance matrix $\boldsymbol{\Sigma}$ is practically infeasible. To overcome this issue, we adopt a block coordinate approach (Cai et al., 2021; Zhang et al., 2024). Figure 4 shows examples of block coordinate perturbations, where the block coordinate nature is adopted not only for sampling perturbations but also for computing the covariance matrix.

4.2.1. Block Coordinate Perturbations

Let $N_{\max}$ be the maximum number of parameters allowed for each block. We partition a parameter vector $\boldsymbol{\theta} \in \mathbb{R}^N$ into small blocks $\boldsymbol{\theta}^{(b)}$, $b = 1, \dots, B$, so that

$$\boldsymbol{\theta} := \begin{bmatrix} \boldsymbol{\theta}^{(1)} \\ \vdots \\ \boldsymbol{\theta}^{(B)} \end{bmatrix}, \; \boldsymbol{\theta}^{(b)} \in \mathbb{R}^{N_b} \text{ and } N_b \le N_{\max} \,. \tag{33}$$

For a parameter vector $\boldsymbol{\theta}$, perturbations $\boldsymbol{u}_q \in \mathbb{R}^N$, $q = 1, \dots, Q$, are made in a block coordinate manner, where only one block $\boldsymbol{u}^{(b)} \in \mathbb{R}^{N_b}$ corresponding to $\boldsymbol{\theta}^{(b)}$ is allowed to have non-zero elements. The block indices b are sampled with replacement if $B < Q$, and without replacement if $B \ge Q$ to totally perturb as many parameters as possible.

4.2.2. Block Diagonal Covariance Matrix

As a consequence of block coordinate perturbations, we are allowed to make the covariance matrix $\boldsymbol{\Sigma}$ block diagonal, whose blocks are covariance matrices $\boldsymbol{\Sigma}_b$, $b = 1, \dots, B$ of small size $N_b \times N_b$ defined as

$$\boldsymbol{\Sigma}_b := \left(\lambda_P \cdot \mathbf{I} + \lambda_F \cdot \mathbf{F}_{\boldsymbol{\theta}^{(b)}}\right)^{-1} . \tag{34}$$

Here, the FIM also becomes block diagonal with blocks $\mathbf{F}_{\boldsymbol{\theta}^{(b)}}$, whose definition differs slightly from (4):

$$\mathbf{F}_{\boldsymbol{\theta}^{(b)}} := \mathbb{E}_{\boldsymbol{x}}\left[\mathbb{E}_{\boldsymbol{z}}\left[\nabla_{\boldsymbol{\theta}^{(b)}} \ln p_{\boldsymbol{\theta}}(\boldsymbol{z}\,|\,\boldsymbol{x})\, \nabla_{\boldsymbol{\theta}^{(b)}} \ln p_{\boldsymbol{\theta}}(\boldsymbol{z}\,|\,\boldsymbol{x})^\mathsf{T}\right]\right] . \tag{35}$$

Its computational procedure is similar to that based on (25):

$$\mathbf{F}_{\boldsymbol{\theta}^{(b)}} = \mathbb{E}_{\boldsymbol{x}}\left[\mathbf{J}_{\boldsymbol{y}\boldsymbol{\theta}^{(b)}}^\mathsf{T} \mathbf{F}_{\boldsymbol{y}} \mathbf{J}_{\boldsymbol{y}\boldsymbol{\theta}^{(b)}}\Big|_{\boldsymbol{y}=\boldsymbol{f}(\boldsymbol{x},\boldsymbol{\theta})}\right] . \tag{36}$$

The computation of $\mathbf{F}_{\boldsymbol{y}}$ is the same as those described in Section 4.1.1. The computation of $\mathbf{J}_{\boldsymbol{y}\boldsymbol{\theta}^{(b)}}$ is similar to (31):

$$\mathbf{J}_{\boldsymbol{y}\boldsymbol{\theta}^{(b)}} = \left[\delta \boldsymbol{y}_1^{(b)}, \dots, \delta \boldsymbol{y}_{N_b}^{(b)}\right] \in \mathbb{R}^{M \times N_b} , \tag{37}$$

$$\delta \boldsymbol{y}_n^{(b)} := \frac{\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta} + \mu \boldsymbol{e}_n^{(b)}) - \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta})}{\mu} \in \mathbb{R}^M , \tag{38}$$

$$\boldsymbol{e}_n^{(b)} := [0, \dots, 1\,(n'\text{th entry}), \dots, 0]^\mathsf{T} \in \mathbb{R}^N \tag{39}$$

with $n' = n + \sum_{i=1}^{b-1} N_i$.

4.3. Overall Procedure

Let us summarize aforementioned techniques as Algorithm 2, which is derived from Algorithm 1. The key operations are in lines 13 and 17, which compute the covariance matrices $\boldsymbol{\Sigma}_b$ and sample natural perturbations $\boldsymbol{u}^{(b)}$ from $\mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_b)$. The matrices $\boldsymbol{\Sigma}_b$ are initialized as identity matrices in line 4, and eventually updated in line 13, depending on which index b is sampled in line 9.

What we need to take care of is the cost of ZO queries. In line 10, the query cost to compute the Jacobian $\mathbf{J}_{\boldsymbol{y}\boldsymbol{\theta}^{(b)}}$, and consequently $\boldsymbol{\Sigma}_b$, is N_b. In line 20, the additional query cost to compute the approximate gradient is Q. To achieve effective black-box local search with a fixed budget for total queries, we should consume as many queries as possible for the approximate gradient. Therefore, we reduce the update frequency of $\boldsymbol{\Sigma}_b$ by introducing a hyperparameter $T_{\text{ud}} > 1$ in line 8. We have experimentally confirmed in Section 5.3.5 that $T_{\text{ud}} = 100$ is appropriate.

5. Experiments

This section reports the experimental results to show the superiority of the proposed method over the existing ones. We coded our programs with PyTorch (Paszke et al., 2019), and ran them on an NVIDIA RTX A6000 (48 GB).

Algorithm 2 ZO optimization for training a neural network with block coordinate *natural perturbations*

Input: training dataset $\mathcal{D}_{\rm tr}=\{(\boldsymbol{x},\boldsymbol{t})\}$, hyperparameters Q, μ, $N_{\max}$, $T_{\rm ud}$, λ_P, λ_F
Initialize the parameter vector $\boldsymbol{\theta}$
Partition $\boldsymbol{\theta}$ into blocks $\{\boldsymbol{\theta}^{(1)},\ldots,\boldsymbol{\theta}^{(B)}\}$ according to (33)
Initialize $\boldsymbol{\Sigma}_b=\mathbf{I}$ for $b=1,\ldots,B$
while not converged, in τ-th iteration **do**
  Sample mini-batch $\mathcal{D}$ from $\mathcal{D}_{\rm tr}$
  Evaluate loss $\ell_{\mathcal{D}}(\boldsymbol{\theta})$ by (1)
  if $(\tau \bmod T_{\rm ud})=0$ **then**
    Sample $b\sim\{1,\ldots,B\}$
    Compute $\mathbf{J}_{\boldsymbol{y}\boldsymbol{\theta}^{(b)}}$ by (37)
    Compute $\mathbf{F}_{\boldsymbol{y}}$ by (28) or Set $\mathbf{F}_{\boldsymbol{y}}=\mathbf{I}$
    Compute $\mathbf{F}_{\boldsymbol{\theta}^{(b)}}$ by (36)
    Compute $\boldsymbol{\Sigma}_b$ by (34)
  end if
  for $q=1,\ldots,Q$ **do**
    Sample $b\sim\{1,\ldots,B\}$
    Sample a perturbation $\boldsymbol{u}^{(b)}$ from $\mathcal{N}(\mathbf{0},\boldsymbol{\Sigma}_b)$
    Set $\boldsymbol{u}_q=[\mathbf{0},\ldots,\boldsymbol{u}^{(b)},\ldots,\mathbf{0}]^{\mathsf{T}}$
  end for
  Compute the approximate gradient $\hat{\nabla}_{\boldsymbol{\theta}}\ell_{\mathcal{D}}(\boldsymbol{\theta})$ by (10) with $\boldsymbol{u}_q$, $q=1,\ldots,Q$ computed above.
  Update the parameter vector $\boldsymbol{\theta}$ by (9)
end while

5.1. Datasets/Tasks and Architectures

The first two columns of Table 1 list the five datasets/tasks we examined and the corresponding neural network architectures with N parameters. We used a CNN for **MNIST** (LeCun & Cortes, 2010) and **FashionMNIST** (Xiao et al., 2017), and MLP-Mixer (Tolstikhin et al., 2021) for **CIFAR10** (Krizhevsky, 2009). For these datasets, the computing elements are parameterized by matrices as in the most general cases. For **Equalization** and **Copying memory** tasks, on the other hand, we used a hardware specific architecture based on Mach-Zehnder interferometers (MZIs) (Reck & Zeilinger, 1994; Clements et al., 2016) assuming optical neural networks (ONNs). The MZI parameterization is completely different from that of a matrix: a linear module realizes a unitary matrix and has a deep structure (see Appendix C for the details), which leads to a situation where many parameters are involved in a path from an input element to an output element even in a single linear module.

The **Equalization** task is a kind of regression task. As Figure 5 shows, the ONN on the upper path tries to realize a given fixed matrix on the lower path. The ONN was in a form of the singular value decomposition $\mathbf{UDV}^{\mathsf{H}}$, where $\mathbf{U}$ and $\mathbf{V}$ are unitary matrices and $\mathbf{D}$ is a diagonal matrix. The absolute values of the complex signals were obtained at the output to convert the optical signals to electrical signals by photodetectors. The loss function was the mean squared error (MSE) loss. In the experiment, the dimensionality of the matrix was set to 16.

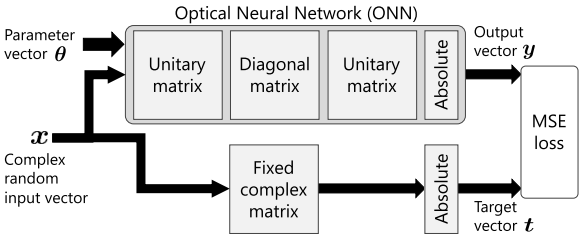

Figure 5. Equalization task

The **Copying memory** task tests how well an RNN remembers data seen many time steps before. An example of input and target output sequences is

Input: `38612----------:----`
Target: `----------38612`

where `38612` is the data to remember, with possible alphabet symbols from `1` to `8`. And `-` and `:` are special symbols meaning 'blank' and 'start', respectively. In total, the RNN has a 10-dimensional output. The details are described in (Arjovsky et al., 2016; Lezcano-Casado & Martınez-Rubio, 2019). The task is particularly well tested when the RNN's hidden unit is realized by a unitary matrix, where the absolute values of all eigenvalues are one. In our experiment, the data length and the time steps to 'start' after the data were 10 and 100, respectively. The size of the unitary matrix used in the hidden unit was 32×32.

We generated the training and test datasets for the Equalization and Copying memory tasks with our own programs. As in the case of MNIST and FashionMNIST, the number of samples in the training and test datasets were 60000 and 10000, respectively.

5.2. Methods and Overall Results

We compared three ZO optimization methods: ZO-**I** and ZO-co are the existing methods, and ZO-NP is our proposed method using natural perturbations. We treat that ZO-**I** is a special case of ZO-NP with $\lambda_P=1$ and $\lambda_F=0$ in (34), and lines from 8 to 14 are skipped in Algorithm 2. In ZO-co, perturbation vectors are sampled as **co**ordinate-wise from a set of one-hot vectors. The number Q of perturbation vectors were set for each task as the sixth column shows. The smoothing hyperparameter was set as $\mu=0.001$.

Instead of the stochastic gradient descent form (9) with approximate gradients, we used the Adam optimizer (Kingma & Ba, 2014) with approximate gradients. The size of mini-batch $\mathcal{D}$ was 100 for all tasks. The learning rate (lr) was appropriately set for each task as the seventh column of Table 1 shows. The results with varying learning rates are summarized in Appendix D.1.

Regarding ZO-**I** and ZO-NP using the block coordinate perturbations explained in Section. 4.2.1, we set the maximum

Table 1. Overview of our experiments and the test accuracies (mean ± standard deviation of five runs). For Equalization, the test R^2's (coefficients of determination) are shown instead, whose best possible value is 1.0 as the test accuracy. The bold values indicate that they are the best and statistically different from the others according to the one-sided Mann-Whitney U test at a significance level of $p = 0.008$.

			Hyperparameters					Test accuracy (↑)		
dataset/task	architecture	N	$N_{\max}$	B	Q	lr	λ_F	ZO-**I**	ZO-co	ZO-NP
MNIST	CNN (matrix)	2586	431	6	20	0.0005	100	0.961 ±0.004	0.966 ±0.001	**0.978** ±0.002
FashionMNIST	CNN (matrix)	2586	431	6	20	0.0005	100	0.796 ±0.013	0.814 ±0.005	**0.858** ±0.004
Equalization	FeedForward (MZI)	560	280	2	20	0.005	10	0.944 ±0.007	0.944 ±0.008	**0.993** ±0.003
Copying memory	RNN (MZI)	3168	453	7	20	0.0001	10	0.965 ±0.026	0.985 ±0.002	**0.993** ±0.003
CIFAR10	MLP-mixer (matrix)	33642	510	66	200	0.001	100	0.595 ±0.002	0.601 ±0.004	**0.625** ±0.004

number $N_{\max}$ of parameters allowed for each block, and consequently had the number B of blocks as shown in the fourth and fifth columns of Table 1. For ZO-NP, the FSD weight hyperparameter λ_F was set appropriately for each task as shown in the eighth column. The other hyperparameters were set as $\lambda_P = 1$ and $T_{\text{ud}} = 100$ for all tasks.

The existing methods ZO-**I** and ZO-co completed 1000 epochs for CIFAR10 and 100 epochs for the other four tasks. However, ZO-NP finished with fewer epochs than these numbers under the same query budget. This was due to the fact that ZO-NP consumed extra queries for the black-box Jacobian computation.

Table 1 shows the overall results. The proposed method ZO-NP outperformed the existing ones with statistical significance in all cases. Also, ZO-NP outperformed CMA-ES (Hansen, 2016; Hansen et al., 2019), which is another black-box optimization method, as shown in Appendix D.2.

5.3. Discussion with Detailed Results

5.3.1. Convergence Behavior

Figure 6 shows the convergence behavior for CIFAR10 (see Appendix D.3 for the other tasks). The left plot shows how the training loss decreased as the number of epochs increased. ZO-NP completed fewer epochs (975) than the other two (1000) with the same query budget of 1.005×10^8. The right plot shows how the test accuracies improved along the elapsed time. Although ZO-NP had a computational overhead over the other two, it was worth paying because ZO-NP outperformed the other two even with the same elapsed time.

5.3.2. Effect of Block Size

Figure 7 compares ZO-co with ZO-**I** and ZO-NP with varying the maximum block size $N_{\max}$. To make the comparison, we slightly modified Algorithm 2 as inserting

$$\boldsymbol{u}^{(b)} \leftarrow (\sqrt{N_b}/\|\boldsymbol{u}^{(b)}\|_2) \cdot \boldsymbol{u}^{(b)}\,, \tag{40}$$

which made the norm of $\boldsymbol{u}^{(b)}$ be $\sqrt{N_b}$, after line 17. Then the behavior of ZO-co and the other two became the same

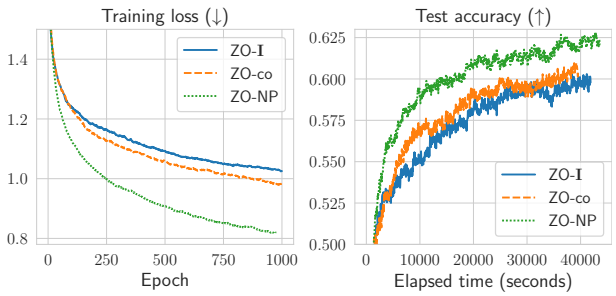

Figure 6. Convergence behavior for CIFAR10

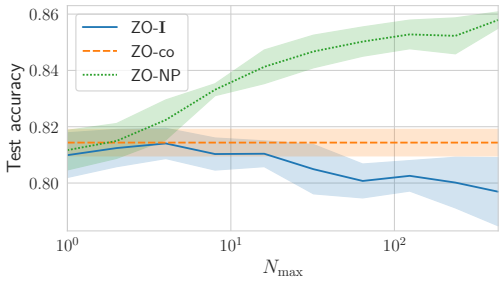

Figure 7. FashionMNIST test accuracies by varying the maximum block size $N_{\max}$ for ZO-**I** and ZO-NP. The size for ZO-co is only valid at $N_{\max} = 10^0$, but is stretched horizontally for easy comparison. Shaded areas show one standard deviation over five independent runs.

under $N_{\max} = N_b = 1$ because the perturbation vectors were one-hot vectors (with ±1 non-zero element) in all methods. The plot shows this empirically at $N_{\max} = 10^0 = 1$. As we increased $N_{\max}$, the performance of ZO-NP steadily increased. This was due to two effects. The first was to increase the number of changing parameters per iteration. The second was to mitigate the parameter correlation represented by the FIM $\mathbf{F}_{\boldsymbol{\theta}^{(b)}}$ in (35). While the proposed method ZO-NP did both, ZO-co did not do the first, and ZO-**I** did the first but not the second.

5.3.3. Computational Overhead of ZO-NP

Table 2 shows the average elapsed times per epoch and the memory footprints for the three methods. Looking at the left numbers, we observe that the elapsed time overhead

Table 2. The average elapsed times in seconds per epoch (left) and the memory footprints in gigabytes (right) for the three methods, under the same experimental conditions shown in Table 1.

dataset/task	ZO-**I**		ZO-co		ZO-NP	
MNIST	4.74	0.57	4.52	0.57	5.04	2.74
FashionMNIST	4.66	0.57	4.43	0.57	5.04	2.74
Equalization	3.25	0.04	3.20	0.04	3.36	0.05
Copying memory	84.89	0.36	84.22	0.36	90.89	2.57
CIFAR10	43.10	1.92	40.31	1.92	45.79	3.83

Table 3. FashionMNIST memory footprints in gigabytes for the ZO-NP settings of Figure 7.

$N_{\max}$	4	8	16	32	64	124	236	431
memory	0.57	0.57	0.57	0.57	0.59	0.94	1.60	2.74

of ZO-NP over the two existing methods was up to around 10 %, which was offset by the efficient training as already shown by the right plot of Figure 6. Looking at the right numbers, we observe that the memory footprint overhead of ZO-NP over the two existing methods was substantial for the tasks except Equalization where the maximum block size $N_{\max}$ was not so large. We can reduce the overhead to a negligible amount by decreasing $N_{\max}$ as shown in Table 3. Together with Figure 7, we understand that ZO-NP trades off the test accuracy and the memory overhead.

Appendix D.4 reports the computational overhead of ZO-NP for an MLP-mixer with $N = 1,706,762$ parameters, which is much larger than the MLP-mixer reported in the last row of Table 1. We observe that the proposed method scales to some extent, but a network with one million parameters might be a practical limitation.

5.3.4. Various numbers Q of perturbations

Table 4 shows FashionMNIST test accuracies with various numbers Q of perturbations. The bold values indicate that they are the best and statistically different from the others according to the one-sided Mann-Whitney U test at a significance level of $p = 0.004$. We observe that the proposed method ZO-NP consistently outperformed the other two methods, except in the extreme case ($Q = 1$), where the number of epochs allowed for ZO-NP was small since the relative query consumption cost for the black-box Jacobian computation was large.

5.3.5. Robustness to Hyperparameter Settings

Figure 8 shows the test accuracies for three tasks by ZO-NP with varying hyperparameters, namely the FSD weight λ_F on the left and the update frequency $T_{\rm ud}$ of $\mathbf{\Sigma}_b$ on the right. The leftmost results with $\lambda_F = 0$ correspond to those of ZO-**I** with fewer epochs (82, 88, and 82 for the three tasks).

Table 4. Second to fourth rows: FashionMNIST test accuracies (mean of five runs) by varying the number Q of perturbation vectors. *Fifth row*: query budget determined by the number of queries that ZO-**I** and ZO-co consumed for 100 epochs. *Sixth row*: the number of epochs allowed for ZO-NP with the same budget.

Q	1	2	5	10	20	50	100
ZO-**I**	0.717	0.736	0.763	0.782	0.796	0.816	0.830
ZO-co	0.714	0.746	0.773	0.798	0.814	0.833	0.845
ZO-NP	0.714	**0.776**	**0.824**	**0.841**	**0.858**	**0.865**	**0.873**
budget ($\times 10^6$)	0.12	0.18	0.36	0.66	1.26	3.06	6.06
ZO-NP epochs	31	41	58	71	82	92	95

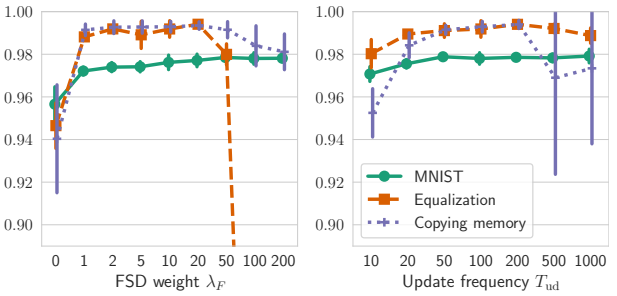

Figure 8. Left: test accuracies obtained by ZO-**I** ($\lambda_F = 0$) and ZO-NP with varying the FSD hyperparameter. Right: those by ZO-NP with varying the update frequency hyperparameter. Vertical bars show one standard deviation over three independent runs.

Both hyperparameters had wide ranges of appropriate settings that outperformed the ZO-**I** results, although the appropriate ranges for λ_F were different depending on the task and we actually customized λ_F as Table 1 shows. Regarding the update frequency, we consider that $T_{\rm ud} = 100$ is an appropriate setting working generally.

6. Conclusion

We have newly proposed an efficient sampling strategy for ZO optimization, where perturbations $\boldsymbol{u}$ are sampled from $\mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$ with a covariance matrix $\mathbf{\Sigma}$ designed by regularizing not only the expected PSD (14) but also the expected FSD (15). We call such sampled $\boldsymbol{u}$ *natural perturbations* since the regularization inherits from the concept of natural gradient. We then have proposed a new ZO optimization method based on the sampling strategy, where the block coordinate approach allows us to compute the covariance matrix efficiently. Experimental results show that the proposed method clearly outperformed the existing ones for a variety of datasets, tasks, and neural network architectures.

Future work includes the development of a much stronger approximation of the FIM than making it block diagonal. This would enable the method to scale to networks with more than millions of parameters, and to be applied to ZO-based large language model (LLM) fine-tuning (Zhang et al., 2024).

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Aguirre, F., Sebastian, A., Le Gallo, M., Song, W., Wang, T., Yang, J. J., Lu, W., Chang, M.-F., Ielmini, D., Yang, Y., et al. Hardware implementation of memristor-based artificial neural networks. *Nature Communications*, 15 (1):1974, 2024.

Amari, S. Backpropagation and stochastic gradient descent method. *Neurocomputing*, 5(4-5):185–196, 1993.

Amari, S. Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276, 1998.

Arjovsky, M., Shah, A., and Bengio, Y. Unitary evolution recurrent neural networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 1120–1128, 2016.

Ashtiani, F., Geers, A. J., and Aflatouni, F. An on-chip photonic deep neural network for image classification. *Nature*, 606:501–506, 2022.

Bae, J., Vicol, P., HaoChen, J. Z., and Grosse, R. B. Amortized proximal optimization. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:8982–8997, 2022.

Bandyopadhyay, S., Sludds, A., Krastanov, S., Hamerly, R., Harris, N., Bunandar, D., Streshinsky, M., Hochberg, M., and Englund, D. Single chip photonic deep neural network with accelerated training. *arXiv preprint arXiv:2208.01623*, 2022.

Banerjee, S., Nikdast, M., and Chakrabarty, K. Characterizing coherent integrated photonic neural networks under imperfections. *Journal of Lightwave Technology*, 41(5): 1464–1479, 2023. doi: 10.1109/JLT.2022.3193658.

Benzing, F. Gradient descent on neurons and its link to approximate second-order optimization. In *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 1817–1853, 2022.

Berahas, A. S., Cao, L., Choromanski, K., and Scheinberg, K. A theoretical and empirical comparison of gradient approximations in derivative-free optimization. *Foundations of Computational Mathematics*, 22(2):507–560, 2022.

Cai, H., Lou, Y., McKenzie, D., and Yin, W. A zeroth-order block coordinate descent algorithm for huge-scale black-box optimization. In *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 1193–1203, 2021.

Chen, A., Zhang, Y., Jia, J., Diffenderfer, J., Parasyris, K., Liu, J., Zhang, Y., Zhang, Z., Kailkhura, B., and Liu, S. DeepZero: Scaling up zeroth-order optimization for deep model training. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

Chen, M. K., Liu, X., Sun, Y., and Tsai, D. P. Artificial intelligence in meta-optics. *Chemical Reviews*, 122(19): 15356–15413, 2022.

Clements, W. R., Humphreys, P. C., Metcalf, B. J., Kolthammer, W. S., and Walmsley, I. A. Optimal design for universal multiport interferometers. *Optica*, 3(12):1460–1465, 2016.

Cong, G., Yamamoto, N., Inoue, T., Maegami, Y., Ohno, M., Kita, S., Namiki, S., and Yamada, K. On-chip bacterial foraging training in silicon photonic circuits for projection-enabled nonlinear classification. *Nature Communications*, 13(1):3261, 2022.

Dangel, F., Kunstner, F., and Hennig, P. BackPACK: Packing more into backprop. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

Duchi, J. C., Jordan, M. I., Wainwright, M. J., and Wibisono, A. Optimal rates for zero-order convex optimization: The power of two function evaluations. *IEEE Transactions on Information Theory*, 61(5):2788–2806, 2015.

Fang, M. Y.-S., Manipatruni, S., Wierzynski, C., Khosrowshahi, A., and DeWeese, M. R. Design of optical neural networks with component imprecisions. *Optical Express*, 27(10):14009–14029, 2019.

Fazel, M., Ge, R., Kakade, S., and Mesbahi, M. Global convergence of policy gradient methods for the linear quadratic regulator. In *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 1467–1476, 2018.

Gu, J., Zhao, Z., Feng, C., Li, W., Chen, R. T., and Pan, D. Z. FLOPS: Efficient on-chip learning for optical neural networks through stochastic zeroth-order optimization. In *Proceedings of the 57th ACM/EDAC/IEEE Design Automation Conference*, 2020.

Gu, J., Feng, C., Zhao, Z., Ying, Z., Chen, R. T., and Pan, D. Z. Efficient on-chip learning for optical neural networks through power-aware sparse zeroth-order optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 7583–7591, 2021.

Hansen, N. The CMA evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*, 2016.

Hansen, N., Akimoto, Y., and Baudis, P. CMA-ES/pycma on Github. Zenodo, DOI:10.5281/zenodo.2559634, February 2019. URL https://doi.org/10.5281/zenodo.2559634.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Krizhevsky, A. Learning multiple layers of features from tiny images. Technical Report TR-2009, University of Toronto, 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

Kunstner, F., Hennig, P., and Balles, L. Limitations of the empirical Fisher approximation for natural gradient descent. *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019.

LeCun, Y. and Cortes, C. MNIST handwritten digit database, 2010. URL http://yann.lecun.com/exdb/mnist.

LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *Nature*, 521:436–444, 2015. doi: 10.1038/nature14539. URL https://doi.org/10.1038/nature14539.

Lezcano-Casado, M. and Martınez-Rubio, D. Cheap orthogonal constraints in neural networks: A simple parametrization of the orthogonal and unitary group. In *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 3794–3803, 2019.

Li, C., Belkin, D., Li, Y., Yan, P., Hu, M., Ge, N., Jiang, H., Montgomery, E., Lin, P., Wang, Z., et al. Efficient and self-adaptive in-situ learning in multilayer memristor neural networks. *Nature Communications*, 9(1):2385, 2018.

Lian, X., Zhang, H., Hsieh, C.-J., Huang, Y., and Liu, J. A comprehensive linear speedup analysis for asynchronous stochastic parallel optimization from zeroth-order to first-order. *Advances in Neural Information Processing Systems (NeurIPS)*, 29, 2016.

Liu, S., Chen, P.-Y., Kailkhura, B., Zhang, G., Hero III, A. O., and Varshney, P. K. A primer on zeroth-order optimization in signal processing and machine learning: Principals, recent advances, and applications. *IEEE Signal Processing Magazine*, 37(5):43–54, 2020.

Lupo, A., Picco, E., Zajnulina, M., and Massar, S. Deep photonic reservoir computer based on frequency multiplexing with fully analog connection between layers. *Optica*, 10 (11):1478–1485, 2023.

Malagò, L. and Pistone, G. Information geometry of the Gaussian distribution in view of stochastic optimization. In *Proceedings of the ACM Conference on Foundations of Genetic Algorithms XIII*, pp. 150–162, 2015.

Martens, J. *Second-order optimization for neural networks*. University of Toronto (Canada), 2016.

Martens, J. New insights and perspectives on the natural gradient method. *The Journal of Machine Learning Research*, 21(1):5776–5851, 2020.

Martens, J. and Grosse, R. Optimizing neural networks with Kronecker-factored approximate curvature. In *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 2408–2417, 2015.

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

Park, H., Amari, S., and Fukumizu, K. Adaptive natural gradient learning algorithms for various stochastic models. *Neural Networks*, 13(7):755–764, 2000.

Pascanu, R. and Bengio, Y. Revisiting natural gradient for deep networks. *arXiv preprint arXiv:1301.3584*, 2013.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019.

Petersen, K. and Pedersen, M. The matrix cookbook. *Technical University of Denmark*, 7(15):510, 2008.

Reck, M. and Zeilinger, A. Experimental realization of any discrete unitary operator. *Phys. Rev. Lett.*, 73(1):58–61, 1994.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

Sawada, H., Aoyama, K., and Ikeda, K. Zeroth-order optimization of optical neural networks with linear combination natural gradient and calibrated model. In *Proceedings of the 61st ACM/IEEE Design Automation Conference*, 2024.

Sawada, H., Aoyama, K., and Notomi, M. Layered-parameter perturbation for zeroth-order optimization of optical neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 20292–20301, 2025.

Shen, Y., Harris, N. C., Skirlo, S., Prabhu, M., Baehr-Jones, T., Hochberg, M., Sun, X., Zhao, S., Larochelle, H., Englund, D., and Soljačić, M. Deep learning with coherent nanophotonic circuits. *Nature Photonics*, 11:441–446, 2017.

Tolstikhin, I., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., Yung, J., Steiner, A., Keysers, D., Uszkoreit, J., et al. MLP-mixer: An all-MLP architecture for vision. *Advances in Neural Information Processing Systems (NeurIPS)*, 34:24261–24272, 2021.

Wierstra, D., Schaul, T., Glasmachers, T., Sun, Y., Peters, J., and Schmidhuber, J. Natural evolution strategies. *The Journal of Machine Learning Research*, 15(1):949–980, 2014.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Zhang, H., Thompson, J., Gu, M., Jiang, X., Cai, H., Liu, P., Shi, Y., Zhang, Y., Karim, M., Lo, G., Luo, X., Dong, B., Kwek, L., and Liu, A. Efficient on-chip training of optical neural networks using genetic algorithm. *ACS Photonics*, 8(6):1662–1672, 2021.

Zhang, T., Wang, J., Dan, Y., Lanqiu, Y., Dai, J., Han, X., Sun, X., and Xu, K. Efficient training and design of photonic neural network through neuroevolution. *Optics Express*, 27(26):37150–37163, 2019.

Zhang, Y., Li, P., Hong, J., Li, J., Zhang, Y., Zheng, W., Chen, P.-Y., Lee, J. D., Yin, W., Hong, M., et al. Revisiting zeroth-order optimization for memory-efficient LLM fine-tuning: A benchmark. In *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 59173–59190, 2024.

Zhao, P., Chen, P.-Y., Wang, S., and Lin, X. Towards query-efficient black-box adversary with zeroth-order natural gradient descent. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 6909–6916, 2020.

Zhou, H., Zhao, Y., Wang, X., Gao, D., Dong, J., and Zhang, X. Self-configuring and reconfigurable silicon photonic signal processor. *ACS Photonics*, 7(3):792–799, 2020.

A. Proof of Theorem 3.2

When we assume $\lambda_P^{(\mathrm{NG})} = \lambda_P$, $\lambda_F^{(\mathrm{NG})} = \lambda_F$ and Assumption 3.1, the difference between the ZO gradient approximation by natural perturbations and the natural gradient is bounded by

$$\left\| \hat{\nabla}_{\boldsymbol{\theta}}^{(\mathrm{NP})} \ell_{\mathcal{D}}(\boldsymbol{\theta}) - \nabla_{\boldsymbol{\theta}}^{(\mathrm{NG})} \ell_{\mathcal{D}}(\boldsymbol{\theta}) \right\|_2 \leq \frac{\mu L}{2} \left(\frac{3+N}{\lambda_P} \right)^{3/2} . \tag{41}$$

Proof. From the assumptions $\lambda_P^{(\mathrm{NG})} = \lambda_P$ and $\lambda_F^{(\mathrm{NG})} = \lambda_F$, (22) can be written using (18) as

$$\nabla_{\boldsymbol{\theta}}^{(\mathrm{NG})} \ell_{\mathcal{D}}(\boldsymbol{\theta}) = \boldsymbol{\Sigma} \, \nabla_{\boldsymbol{\theta}} \ell_{\mathcal{D}}(\boldsymbol{\theta}) . \tag{42}$$

Then, considering $\mathbb{E}_{\boldsymbol{u} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})} \left[\boldsymbol{u} \boldsymbol{u}^{\mathsf{T}} \right] = \boldsymbol{\Sigma}$ as its definition, we have

$$\nabla_{\boldsymbol{\theta}}^{(\mathrm{NG})} \ell_{\mathcal{D}}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{u} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})} \left[\boldsymbol{u} \boldsymbol{u}^{\mathsf{T}} \right] \nabla_{\boldsymbol{\theta}} \ell_{\mathcal{D}}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{u} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})} \left[\boldsymbol{u} \boldsymbol{u}^{\mathsf{T}} \nabla_{\boldsymbol{\theta}} \ell_{\mathcal{D}}(\boldsymbol{\theta}) \right] . \tag{43}$$

The difference between $\hat{\nabla}_{\boldsymbol{\theta}}^{(\mathrm{NP})} \ell_{\mathcal{D}}(\boldsymbol{\theta})$ defined in (21) and the above, measured by the l_2-norm, can be expressed as

$$\begin{aligned}
\left\| \hat{\nabla}_{\boldsymbol{\theta}}^{(\mathrm{NP})} \ell_{\mathcal{D}}(\boldsymbol{\theta}) - \nabla_{\boldsymbol{\theta}}^{(\mathrm{NG})} \ell_{\mathcal{D}}(\boldsymbol{\theta}) \right\|_2 &= \left\| \mathbb{E}_{\boldsymbol{u} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})} \left[\frac{\ell_{\mathcal{D}}(\boldsymbol{\theta} + \mu \boldsymbol{u}) - \ell_{\mathcal{D}}(\boldsymbol{\theta})}{\mu} \boldsymbol{u} \right] - \mathbb{E}_{\boldsymbol{u} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})} \left[\boldsymbol{u} \boldsymbol{u}^{\mathsf{T}} \nabla_{\boldsymbol{\theta}} \ell_{\mathcal{D}}(\boldsymbol{\theta}) \right] \right\|_2 && (44) \\
&= \left\| \mathbb{E}_{\boldsymbol{u} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})} \left[\frac{\ell_{\mathcal{D}}(\boldsymbol{\theta} + \mu \boldsymbol{u}) - \ell_{\mathcal{D}}(\boldsymbol{\theta})}{\mu} \boldsymbol{u} - \boldsymbol{u} \left\{ \boldsymbol{u}^{\mathsf{T}} \nabla_{\boldsymbol{\theta}} \ell_{\mathcal{D}}(\boldsymbol{\theta}) \right\} \right] \right\|_2 && (45) \\
&= \left\| \mathbb{E}_{\boldsymbol{u} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})} \left[\frac{\ell_{\mathcal{D}}(\boldsymbol{\theta} + \mu \boldsymbol{u}) - \ell_{\mathcal{D}}(\boldsymbol{\theta}) - \mu \boldsymbol{u}^{\mathsf{T}} \nabla_{\boldsymbol{\theta}} \ell_{\mathcal{D}}(\boldsymbol{\theta})}{\mu} \boldsymbol{u} \right] \right\|_2 && (46) \\
&\overset{(23)}{\leq} \left\| \mathbb{E}_{\boldsymbol{u} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})} \left[\frac{1}{\mu} \frac{L}{2} \|\mu \boldsymbol{u}\|_2^2 \, \boldsymbol{u} \right] \right\|_2 && (47) \\
&= \frac{\mu L}{2} \left\| \mathbb{E}_{\boldsymbol{u} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})} \left[\|\boldsymbol{u}\|_2^2 \boldsymbol{u} \right] \right\|_2 && (48) \\
&\leq \frac{\mu L}{2} \mathbb{E}_{\boldsymbol{u} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})} \left[\|\boldsymbol{u}\|_2^2 \, \|\boldsymbol{u}\|_2 \right] . && (49)
\end{aligned}$$

We have applied Assumption 3.1 in the transition from (46) to (47). Now, by generating $\boldsymbol{u}$ via a linear transformation $\boldsymbol{u} = \boldsymbol{\Sigma}^{1/2} \boldsymbol{v}$ with sampled $\boldsymbol{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, we have

$$\begin{aligned}
\left\| \hat{\nabla}_{\boldsymbol{\theta}}^{(\mathrm{NP})} \ell_{\mathcal{D}}(\boldsymbol{\theta}) - \nabla_{\boldsymbol{\theta}}^{(\mathrm{NG})} \ell_{\mathcal{D}}(\boldsymbol{\theta}) \right\|_2 &\leq \frac{\mu L}{2} \mathbb{E}_{\boldsymbol{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[\boldsymbol{v}^{\mathsf{T}} \boldsymbol{\Sigma} \boldsymbol{v} \left\| \boldsymbol{\Sigma}^{1/2} \boldsymbol{v} \right\|_2 \right] && (50) \\
&\leq \frac{\mu L}{2} \left\| \boldsymbol{\Sigma}^{1/2} \right\|_2 \mathbb{E}_{\boldsymbol{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[\boldsymbol{v}^{\mathsf{T}} \boldsymbol{\Sigma} \boldsymbol{v} \, \|\boldsymbol{v}\|_2 \right] && (51) \\
&\leq \frac{\mu L}{2} \left\| \boldsymbol{\Sigma}^{1/2} \right\|_2 \|\boldsymbol{\Sigma}\|_2 \, \mathbb{E}_{\boldsymbol{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[\|\boldsymbol{v}\|_2^2 \, \|\boldsymbol{v}\|_2 \right] , && (52)
\end{aligned}$$

where the matrix norm is the spectral norm and thus the maximum eigenvalue. From the structure (18) of $\boldsymbol{\Sigma}$, we have

$$\|\boldsymbol{\Sigma}\|_2 = \max \left(\mathrm{eig} \left(\boldsymbol{\Sigma} \right) \right) = \max \left(\mathrm{eig} \left(\left(\lambda_P \cdot \mathbf{I} + \lambda_F \cdot \mathbf{F}_{\boldsymbol{\theta}} \right)^{-1} \right) \right) = \min \left(\mathrm{eig} \left(\lambda_P \cdot \mathbf{I} + \lambda_F \cdot \mathbf{F}_{\boldsymbol{\theta}} \right) \right)^{-1} \leq \lambda_P^{-1} \tag{53}$$

from (Petersen & Pedersen, 2008) and similarly $\left\| \boldsymbol{\Sigma}^{1/2} \right\|_2 \leq \lambda_P^{-1/2}$. For N-dimensional vectors $\boldsymbol{v}$, we have

$$\mathbb{E}_{\boldsymbol{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[\|\boldsymbol{v}\|_2^2 \, \|\boldsymbol{v}\|_2 \right] \leq (3+N)^{3/2} \tag{54}$$

from Lemma 1 of (Nesterov & Spokoiny, 2017). Therefore,

$$\left\| \hat{\nabla}_{\boldsymbol{\theta}}^{(\mathrm{NP})} \ell_{\mathcal{D}}(\boldsymbol{\theta}) - \nabla_{\boldsymbol{\theta}}^{(\mathrm{NG})} \ell_{\mathcal{D}}(\boldsymbol{\theta}) \right\|_2 \leq \frac{\mu L}{2} \left(\frac{3+N}{\lambda_P} \right)^{3/2} . \tag{55}$$

□

B. Differences between Natural perturbations and Natural evolution strategies

Natural evolution strategies (NES) (Wierstra et al., 2014) are a family of black-box optimization algorithms that use the natural gradient, and are closely related to covariance matrix adaptation evolution strategy (CMA-ES) (Hansen, 2016; Hansen et al., 2019). Since our proposed ZO optimization with natural perturbations and NES are both black-box optimization methods that use the Fisher information matrix (FIM), here we clarify the differences between them in Table 5 in terms of the FIM. As the table shows, the definitions and usage of the FIMs are fundamentally different. Due to the size of the FIM, our proposed method using natural perturbations can be applied to larger problems with more parameters than NES.

Table 5. Differences between Natural perturbations and Natural evolution strategies (NES)

	Natural perturbations	Natural evolution strategies (NES)
FIM is computed for	distributions that the neural network expresses	sampling distribution
FIM is used for	directly as the covariance matrix of the sampling distribution	iteratively updating the parameters of the sampling distribution with a small learning rate in a natural gradient manner
FIM of	neural network parameters, whose number is N, to be optimized	all kinds of parameters of sampling distribution, e.g., mean and covariance matrix for a multivariate normal distribution
size of FIM	$N \times N$	$(N + N^2) \times (N + N^2)$ for a multivariate normal distribution

C. Optical neural network (ONN) based on Mach-Zehnder interferometers (MZIs)

This appendix explains an ONN based on MZIs, which we assumed in the experiments as described in 5.1.

Figure 9 shows the structure. We used the Clements mesh (Clements et al., 2016) for a linear module. It consists of a layered array of MZIs colored in light blue. An MZI consists of two pairs of phase shifters and beam splitters, whose functionalities are shown in the 2×2 matrices. An MZI realizes a 2×2 unitary matrix depending on the settings of the phase parameters θ colored in orange. The Clements mesh realizes a unitary matrix of larger size by structurally combining multiple MZIs.

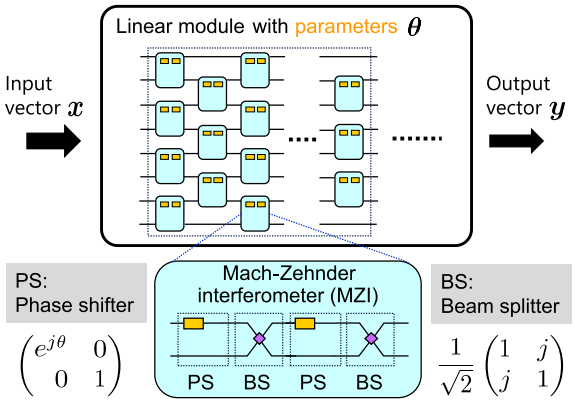

Figure 9. Optical neural network (ONN) based on Mach-Zehnder interferometers (MZIs)

D. Additional Experimental Results

This appendix reports additional experimental results. The test accuracy values for the Equalization were actually computed by the test R^2's (coefficients of determination), whose best possible value is 1.0 as the test accuracy.

D.1. Results with Various Learning Rates

The results reported in Section 5.3 were obtained by setting the learning rate (lr) as shown in Table 1. Here, Figure 10 shows the results with various learning rates for all tasks except CIFAR10. Specifically, we examined five learning rates for each task, where the third (middle) one corresponds to the setting reported in Section 5.3. We observe that ZO-NP consistently outperformed the other two methods with various learning rates, except for the extreme cases at both ends in the Copying memory task.

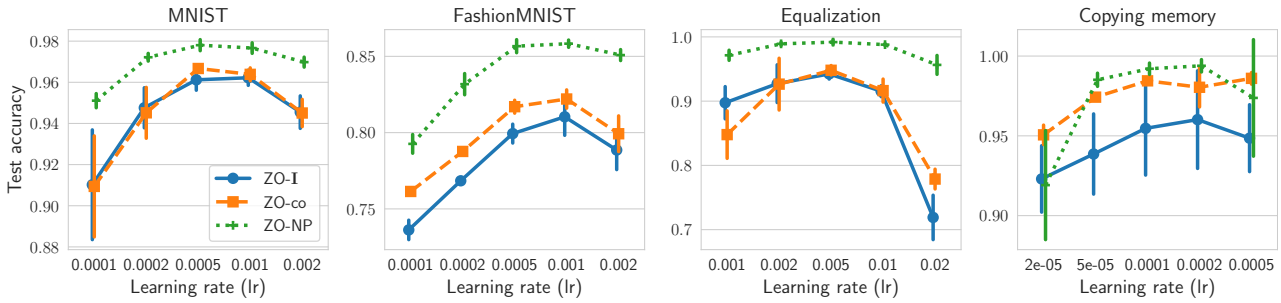

Figure 10. The effects of learning rate (lr) on test accuracies for the MNIST, FashionMNIST, Equalization, and Copying memory tasks. Vertical bars show one standard deviation over three independent runs.

D.2. Comparisons to CMA-ES

In addition to the comparison summary reported in Table 1, here we compare the results with covariance matrix adaptation evolution strategy CMA-ES (Hansen, 2016), which is a well-known black-box optimization method. It computes a covariance matrix using selected solutions from the population, in a fundamentally different way from ours $\mathbf{\Sigma}$ (18) and $\mathbf{\Sigma}_b$ (34). For performing CMA-ES, we used a python package (Hansen et al., 2019). Since it was difficult for the CMA-ES python package to introduce the block coordinate approach used in ZO-I and ZO-NP, we examined tasks with small numbers N of parameters, namely the MNIST and FashionMNIST dataset with the same CNN used in Section 3.2 ($N = 816$) and the Equalization task ($N = 560$). The initial standard deviation hyperparameter of CMA-ES was appropriately set to `sigma0` $= 0.005$, `sigma0` $= 0.002$, and `sigma0` $= 0.009$ for the MNIST, FashionMNIST, and Equalization tasks, respectively. These numbers were chosen as the best out of six different values for each task. Figure 11 shows the results. The proposed method ZO-NP consistently outperformed the other three methods.

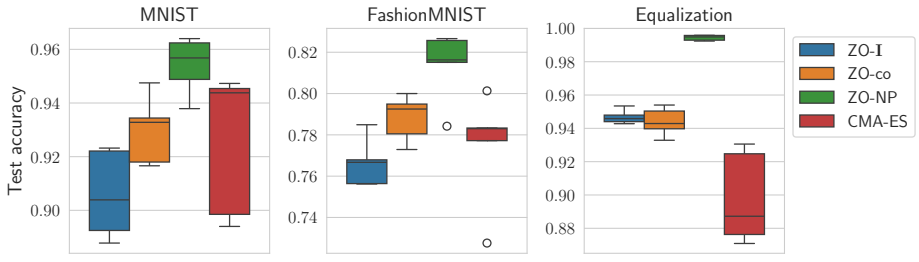

Figure 11. Box plots for the test accuracies obtained by the four methods including CMA-ES. Each box shows the distribution of the results by five independent runs. The number N of parameters were 816, 816, and 560 for the MNIST, FashionMNIST, and Equalization tasks, respectively.

D.3. Convergence Behavior

The convergence behavior for CIFAR10 is reported in Section 5.3.1. Here, Figure 12 shows those for the other four tasks. The upper plots show that ZO-NP completed fewer epochs (82, 82, 88, and 82 for the four tasks) than the other two methods (100, 100, 100, and 100 for the four tasks). This was because the query budget was 1.26×10^6, and ZO-NP consumed extra queries for the black-box Jacobian computation. The lower plots show how the test accuracies improved along the elapsed time. The times of ZO-NP were shorter than those of the other two in most cases. The reason was that ZO-NP was allowed fewer epochs as described above and the computational overhead of ZO-NP over the other two methods was not large.

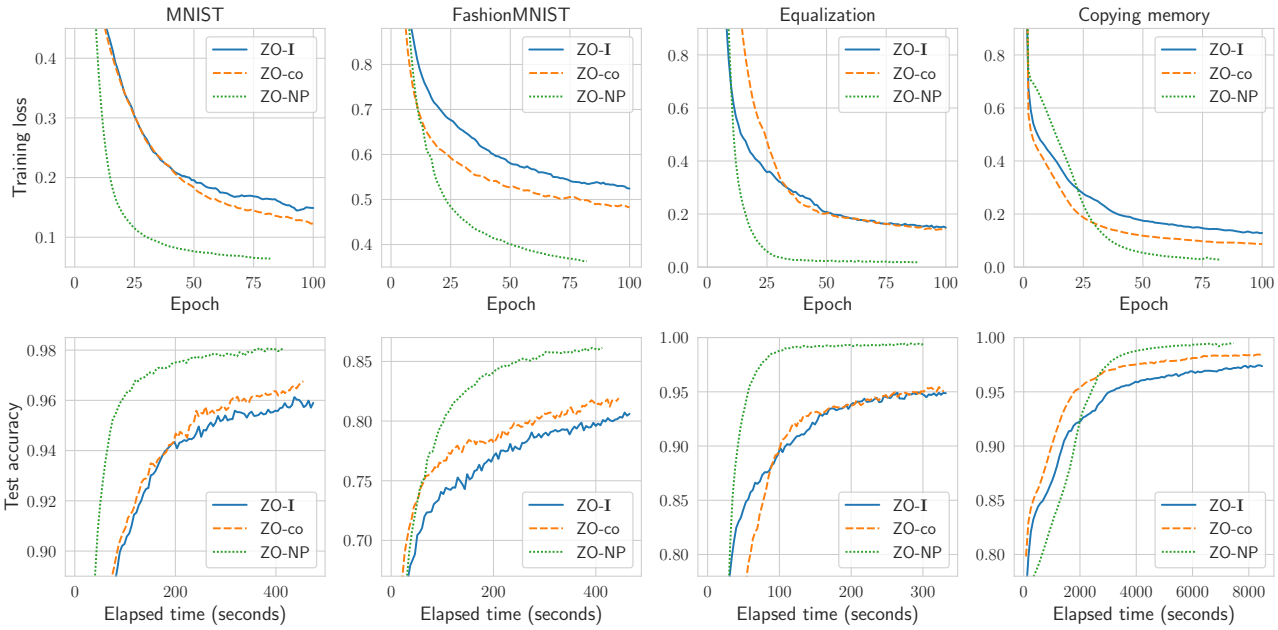

Figure 12. Convergence behaviors for the MNIST, FashionMNIST, Equalization, and Copying memory tasks. Each column corresponds to each task. Top and bottom rows correspond to the training losses and the test accuracies, respectively.

D.4. Computational costs for a neural network with one million parameters

This appendix part shows the computational scalability of our proposed method ZO-NP for a neural network with one million parameters, as well as its limitations.

The largest neural network reported in the main body is the MLP-mixer with $N = 33,642$ parameters, as shown at the last row of Table 1. Here, we report the computational costs of the three ZO methods applied to a much larger neural network. Specifically, for the CIFAR10 task with the MLP-mixer, we increased the number of mixers from 3 to 12, the channel width from 32 to 256, and consequently the number N of parameters from 33,642 to 1,706,762. Table 6 shows the computational cost of the enlarged MLP-mixer. We can observe the computational overhead of ZO-NP over ZO-**I** and ZO-co by looking at the second to fourth rows. While the elapsed time (reported as seconds/epoch) overhead of ZO-NP over ZO-co was $650.79 - 593.84 = 56.95$, which was less than 10% of 593.84, the memory footprint overhead was $31.10 - 12.10 = 19.00$, which was considerable. We could reduce the memory footprint overhead by decreasing the maximum block size $N_{\max}$ as shown in the last two rows. As a tradeoff, however, this increased the number B of blocks and consequently the elapsed time in seconds per epoch.

Table 6. The elapsed times in seconds per epoch and the memory footprints in gigabytes for the CIFAR10 task using the enlarged MLP-mixer with $N = 1,706,762$ parameters.

method	$N_{\max}$	B	seconds/epoch	memory (GB)
ZO-**I**	512	3334	636.85	12.10
ZO-co	512	3334	593.84	12.10
ZO-NP	512	3334	650.79	31.10
ZO-NP	256	6668	693.11	16.55
ZO-NP	128	13335	772.62	13.73

A more critical problem of ZO-NP for the enlarged network with a large number B of blocks is that computing the FIM for all blocks takes a significant amount of epochs and practically does not finish. In other words, the FIMs of many blocks remain unchanged as the initialized identity matrix $\mathbf{I}$. Table 7 shows such situations. The second row corresponds to the last row in Table 1, where the number B was not so large. The third to fifth rows correspond to the above introduced enlarged MLP-mixer. Since the number B of blocks was very large in these enlarged networks, many blocks remained as initialized even after 1000 epochs with the update frequency hyperparameter $T_{\mathrm{ud}} = 100$.

Table 7. The number of blocks where the FIM was updated from the initialized identity matrix $\mathbf{I}$ as epochs proceeded in ZO-NP.

		epochs									
network	B	1	2	5	10	20	50	100	200	500	1000
MLP-mixer in Table 1	66	5	10	22	36	52	66	66	66	66	66
Enlarged MLP-mixer	3334	5	10	25	49	98	237	465	871	1751	2580
Enlarged MLP-mixer	6668	5	10	25	50	100	249	488	939	2086	3530
Enlarged MLP-mixer	13335	5	10	25	50	100	248	493	966	2285	4176