Supplementary Materials: Controllable Orthogonalization in Training DNNs

Lei Huang[1] Li Liu[1] Fan Zhu[1] Diwen Wan[1,2] Zehuan Yuan[3] Bo Li[4] Ling Shao[1]
[1]Inception Institute of Artificial Intelligence (IIAI), Abu Dhabi, UAE
[2]University of Electronic Science and Technology of China, Chengdu, China
[3]ByteDance AI Lab, Beijing, China
[4]University of Illinois at Urbana-Champaign Illinois, USA

Algorithm I Orthogonalization by Newton's Iteration.

1: **Input**: proxy parameters $\mathbf{Z} \in \mathbb{R}^{n\times d}$ and iteration numbers T.
2: Bounding $\mathbf{Z}$'s singular values: $\mathbf{V} = \frac{\mathbf{Z}}{\|\mathbf{Z}\|_F}$.
3: Calculate covariance matrix: $\mathbf{S} = \mathbf{V}\mathbf{V}^T$.
4: $\mathbf{B}_0 = \mathbf{I}$.
5: **for** $t = 1$ to T **do**
6: $\quad \mathbf{B}_t = \frac{3}{2}\mathbf{B}_{t-1} - \frac{1}{2}\mathbf{B}_{t-1}^3\mathbf{S}$.
7: **end for**
8: $\mathbf{W} = \mathbf{B}_T\mathbf{V}$.
9: **Output**: orthogonalized weight matrix: $\mathbf{W} \in \mathbb{R}^{n\times d}$.

A. Derivation of Back-Propagation

Given the layer-wise orthogonal weight matrix $\mathbf{W}$, we can perform the forward pass to calculate the loss of the deep neural networks (DNNs). It's necessary to back-propagate through the orthogonalization transformation, because we aim to update the proxy parameters $\mathbf{Z}$. For illustration, we first describe the proposed orthogonalization by Newton's iteration (ONI) in Algorithm I. Given the gradient with respect to the orthogonalized weight matrix $\frac{\partial\mathcal{L}}{\partial\mathbf{W}}$, we target to compute $\frac{\partial\mathcal{L}}{\partial\mathbf{Z}}$. The back-propagation is based on the chain rule. From Line 2 in Algorithm I, we have:

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\mathbf{Z}} &= \frac{1}{\|\mathbf{Z}\|_F}\frac{\partial\mathcal{L}}{\partial\mathbf{V}} + tr(\frac{\partial\mathcal{L}}{\partial\mathbf{V}}^T\mathbf{Z})\frac{\partial\frac{1}{\|\mathbf{Z}\|_F}}{\partial\|\mathbf{Z}\|_F}\frac{\partial\|\mathbf{Z}\|_F}{\partial\mathbf{Z}} \\ &= \frac{1}{\|\mathbf{Z}\|_F}\frac{\partial\mathcal{L}}{\partial\mathbf{V}} - tr(\frac{\partial\mathcal{L}}{\partial\mathbf{V}}^T\mathbf{Z})\frac{1}{\|\mathbf{Z}\|_F^2}\frac{\mathbf{Z}}{\|\mathbf{Z}\|_F} \\ &= \frac{1}{\|\mathbf{Z}\|_F}(\frac{\partial\mathcal{L}}{\partial\mathbf{V}} - \frac{tr(\frac{\partial\mathcal{L}}{\partial\mathbf{V}}^T\mathbf{Z})}{\|\mathbf{Z}\|_F^2}\mathbf{Z}),\end{aligned} \tag{1}$$

where $tr(\cdot)$ indicates the trace of the corresponding matrix and $\frac{\partial\mathcal{L}}{\partial\mathbf{V}}$ can be calculated from Lines 3 and 8 in Algorithm I:

$$\frac{\partial\mathcal{L}}{\partial\mathbf{V}} = (\mathbf{B}_T)^T\frac{\partial\mathcal{L}}{\partial\mathbf{W}} + (\frac{\partial\mathcal{L}}{\partial\mathbf{S}} + \frac{\partial\mathcal{L}}{\partial\mathbf{S}}^T)\mathbf{V}. \tag{2}$$

We thus need to calculate $\frac{\partial\mathcal{L}}{\partial\mathbf{S}}$, which can be computed from Lines 5, 6 and 7 in Algorithm I:

$$\frac{\partial L}{\partial\mathbf{S}} = -\frac{1}{2}\sum_{t=1}^{T}(\mathbf{B}_{t-1}^3)^T\frac{\partial L}{\partial\mathbf{B}_t}, \tag{3}$$

where $\frac{\partial\mathcal{L}}{\partial\mathbf{B}_T} = \frac{\partial\mathcal{L}}{\partial\mathbf{W}}\mathbf{V}^T$ and $\{\frac{\partial\mathcal{L}}{\partial\mathbf{B}_{t-1}}, t = T, ..., 1\}$ can be iteratively calculated from Line 6 in Algorithm I as follows:

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\mathbf{B}_{t-1}} = -\frac{1}{2}(&\frac{\partial\mathcal{L}}{\partial\mathbf{B}_t}(\mathbf{B}_{t-1}^2\mathbf{S})^T + (\mathbf{B}_{t-1}^2)^T\frac{\partial\mathcal{L}}{\partial\mathbf{B}_t}\mathbf{S}^T \\ &+ \mathbf{B}_{t-1}^T\frac{\partial\mathcal{L}}{\partial\mathbf{B}_t}(\mathbf{B}_{t-1}\mathbf{S})^T) + \frac{3}{2}\frac{\partial\mathcal{L}}{\partial\mathbf{B}_t}.\end{aligned} \tag{4}$$

Algorithm II Back-propagation of ONI.

1: **Input**: $\frac{\partial\mathcal{L}}{\partial\mathbf{W}} \in \mathbb{R}^{n\times d}$ and variables from respective forward pass: $\mathbf{Z}$, $\mathbf{V}$, $\mathbf{S}$, $\{\mathbf{B}_t\}_{t=1}^T$.
2: $\frac{\partial\mathcal{L}}{\partial\mathbf{B}_T} = \frac{\partial\mathcal{L}}{\partial\mathbf{W}}\mathbf{V}^T$.
3: **for** $t = T$ down to 1 **do**
4: $\quad \frac{\partial\mathcal{L}}{\partial\mathbf{B}_{t-1}} = -\frac{1}{2}(\frac{\partial\mathcal{L}}{\partial\mathbf{B}_t}(\mathbf{B}_{t-1}^2\mathbf{S})^T + (\mathbf{B}_{t-1}^2)^T\frac{\partial\mathcal{L}}{\partial\mathbf{B}_t}\mathbf{S}^T + \mathbf{B}_{t-1}^T\frac{\partial\mathcal{L}}{\partial\mathbf{B}_t}(\mathbf{B}_{t-1}\mathbf{S})^T) + \frac{3}{2}\frac{\partial\mathcal{L}}{\partial\mathbf{B}_t}$.
5: **end for**
6: $\frac{\partial L}{\partial\mathbf{S}} = -\frac{1}{2}\sum_{t=1}^{T}(\mathbf{B}_{t-1}^3)^T\frac{\partial L}{\partial\mathbf{B}_t}$.
7: $\frac{\partial\mathcal{L}}{\partial\mathbf{V}} = (\mathbf{B}_T)^T\frac{\partial\mathcal{L}}{\partial\mathbf{W}} + (\frac{\partial\mathcal{L}}{\partial\mathbf{S}} + \frac{\partial\mathcal{L}}{\partial\mathbf{S}}^T)\mathbf{V}$.
8: $\frac{\partial\mathcal{L}}{\partial\mathbf{Z}} = \frac{1}{\|\mathbf{Z}\|_F}(\frac{\partial\mathcal{L}}{\partial\mathbf{V}} - \frac{tr(\frac{\partial\mathcal{L}}{\partial\mathbf{V}}^T\mathbf{Z})}{\|\mathbf{Z}\|_F^2}\mathbf{Z})$.
9: **Output**: $\frac{\partial\mathcal{L}}{\partial\mathbf{Z}} \in \mathbb{R}^{n\times d}$.

In summary, the back-propagation of Algorithm I is shown in Algorithm II.

We further derive the back-propagation of the accelerated ONI method with the centering and more compact spectral bounding operation, as described in Section 3.3 of the paper. For illustration, Algorithm III describes the forward pass of the accelerated ONI. Following the calculation in Algorithm II, we can obtain $\frac{\partial\mathcal{L}}{\partial\mathbf{V}}$. To simplify the derivation, we represent Line 3 of Algorithm III as the following

Algorithm III ONI with acceleration.

1: **Input**: proxy parameters $\mathbf{Z} \in \mathbb{R}^{n\times d}$ and iteration numbers N.
2: Centering: $\mathbf{Z}_c = \mathbf{Z} - \frac{1}{d}\mathbf{Z}\mathbf{1}\mathbf{1}^T$.
3: Bounding $\mathbf{Z}$'s singular values: $\mathbf{V} = \frac{\mathbf{Z}_c}{\sqrt{\|\mathbf{Z}_c\mathbf{Z}_c^T\|_F}}$.
4: Execute Step. 3 to 8 in Algorithm I.
5: **Output**: orthogonalized weight matrix: $\mathbf{W} \in \mathbb{R}^{n\times d}$.

formulations:

$$\mathbf{M} = \mathbf{Z}_c\mathbf{Z}_c^T \tag{5}$$

$$\delta = \sqrt{\|\mathbf{M}\|_F} \tag{6}$$

$$\mathbf{V} = \frac{\mathbf{Z}_c}{\delta}. \tag{7}$$

It's easy to calculate $\frac{\partial\mathcal{L}}{\partial\mathbf{Z}_c}$ from Eqn.5 and Eqn.7 as follows:

$$\frac{\partial\mathcal{L}}{\partial\mathbf{Z}_c} = \frac{1}{\delta}\frac{\partial\mathcal{L}}{\partial\mathbf{V}} + (\frac{\partial\mathcal{L}}{\partial\mathbf{M}} + \frac{\partial\mathcal{L}}{\partial\mathbf{M}}^T)\mathbf{Z}_c, \tag{8}$$

where $\frac{\partial\mathcal{L}}{\partial\mathbf{M}}$ can be computed based on Eqn. 6 and Eqn. 7:

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\mathbf{M}} &= \frac{\partial\mathcal{L}}{\partial\delta}\frac{\partial\delta}{\partial\|\mathbf{M}\|_F}\frac{\partial\|\mathbf{M}\|_F}{\partial\mathbf{M}} \\ &= tr(\frac{\partial\mathcal{L}}{\partial\mathbf{V}}^T\mathbf{Z}_c)(-\frac{1}{\delta^2})\frac{1}{2\sqrt{\|\mathbf{M}\|_F}}\frac{\mathbf{M}}{\|\mathbf{M}\|_F} \\ &= -\frac{tr(\frac{\partial\mathcal{L}}{\partial\mathbf{V}}^T\mathbf{Z}_c)}{2\delta^5}\mathbf{M}.\end{aligned} \tag{9}$$

Based on Line 2 in Algorithm III, we can achieve $\frac{\partial\mathcal{L}}{\partial\mathbf{Z}}$ as follows:

$$\frac{\partial\mathcal{L}}{\partial\mathbf{Z}} = \frac{\partial\mathcal{L}}{\partial\mathbf{Z}_c} - \frac{1}{d}\mathbf{1}\mathbf{1}^T\frac{\partial\mathcal{L}}{\partial\mathbf{Z}_c}. \tag{10}$$

In summary, Algorithm IV describes the back-propagation of the Algorithm III.

B. Proof of Convergence Condition for Newton's Iteration

In Section 3 of the paper, we show that bounding the spectral of the proxy parameters matrix by

$$\mathbf{V} = \phi_N(\mathbf{Z}) = \frac{\mathbf{Z}}{\|\mathbf{Z}\|_F} \tag{11}$$

and

$$\mathbf{V} = \phi_N(\mathbf{Z}) = \frac{\mathbf{Z}}{\sqrt{\|\mathbf{Z}\mathbf{Z}^T\|_F}} \tag{12}$$

can satisfy the convergence condition of Newton's Iterations as follows:

$$\|\mathbf{I} - \mathbf{S}\|_2 < 1, \tag{13}$$

Algorithm IV Back-propagation of ONI with acceleration.

1: **Input**: $\frac{\partial\mathcal{L}}{\partial\mathbf{W}} \in \mathbb{R}^{n\times d}$ and variables from respective forward pass: $\mathbf{Z}_c$, $\mathbf{V}$, $\mathbf{S}$, $\{\mathbf{B}_t\}_{t=1}^T$.
2: Calculate $\frac{\partial\mathcal{L}}{\partial\mathbf{V}}$ from Line 2 to Line 7 in Algorithm II.
3: Calculate $\mathbf{M}$ and δ from Eqn. 5 and Eqn. 6.
4: Calculate $\frac{\partial\mathcal{L}}{\partial\mathbf{M}}$ based on Eqn. 9.
5: Calculate $\frac{\partial\mathcal{L}}{\partial\mathbf{Z}_c}$ based on Eqn. 8.
6: Calculate $\frac{\partial\mathcal{L}}{\partial\mathbf{Z}}$ based on Eqn. 10.
7: **Output**: $\frac{\partial\mathcal{L}}{\partial\mathbf{Z}} \in \mathbb{R}^{n\times d}$.

where $\mathbf{S} = \mathbf{V}\mathbf{V}^T$ and the singular values of $\mathbf{Z}$ are nonzero. Here we will prove this conclusion, and we also prove that $\|\mathbf{Z}\|_F > \sqrt{\|\mathbf{Z}\mathbf{Z}^T\|_F}$.

Proof. By definition, $\|\mathbf{Z}\|_F$ can be represented as $\|\mathbf{Z}\|_F = \sqrt{tr(\mathbf{Z}\mathbf{Z}^T)}$. Given Eqn.11, we calculate

$$\mathbf{S} = \mathbf{V}\mathbf{V}^T = \frac{\mathbf{Z}\mathbf{Z}^T}{tr(\mathbf{Z}\mathbf{Z}^T)}. \tag{14}$$

Let's denote $\mathbf{M} = \mathbf{Z}\mathbf{Z}^T$ and the eigenvalues of $\mathbf{M}$ are $\{\lambda_1, ..., \lambda_n\}$. We have $\lambda_i > 0$, since $\mathbf{M}$ is a real symmetric matrix and the singular values of $\mathbf{Z}$ are nonzero. We also have $\mathbf{S} = \frac{\mathbf{M}}{tr(\mathbf{M})}$ and the eigenvalues of $\mathbf{S}$ are $\frac{\lambda_i}{\sum_{i=1}^n\lambda_i}$. Furthermore, the eigenvalues of $(\mathbf{I} - \mathbf{S})$ are $1 - \frac{\lambda_i}{\sum_{i=1}^n\lambda_i}$, thus satisfying the convergence condition described by Eqn.13.

Similarly, given $\mathbf{V} = \phi_N(\mathbf{Z}) = \frac{\mathbf{Z}}{\sqrt{\|\mathbf{Z}\mathbf{Z}^T\|_F}}$, we have $\mathbf{S} = \frac{\mathbf{Z}\mathbf{Z}^T}{\|\mathbf{Z}\mathbf{Z}^T\|_F} = \frac{\mathbf{M}}{\|\mathbf{M}\|_F}$ and its corresponding eigenvalues are $\frac{\lambda_i}{\sqrt{\sum_{i=1}^n\lambda_i^2}}$. Therefore, the singular values of $(\mathbf{I} - \mathbf{S})$ are $1 - \frac{\lambda_i}{\sqrt{\sum_{i=1}^n\lambda_i^2}}$, also satisfying the convergence condition described by Eqn.13.

We have $\|\mathbf{Z}\|_F = \sqrt{tr(\mathbf{M})} = \sqrt{\sum_{i=1}^n\lambda_i}$ and $\sqrt{\|\mathbf{Z}\mathbf{Z}^T\|_F} = \sqrt{\|\mathbf{M}\|_F} = \sqrt[4]{\sum_{i=1}^n\lambda_i^2}$. It's easy to demonstrate that $\|\mathbf{Z}\|_F > \sqrt{\|\mathbf{Z}\mathbf{Z}^T\|_F}$, since $(\sum_{i=1}^n\lambda_i)^2 > \sum_{i=1}^n\lambda_i^2$. □

In Section 3 of the paper, we show that the Newton's iteration by bounding the spectrum with Eqn. 11 is equivalent to the Newton's iteration proposed in [11]. Here, we provide the details. In [11], they bound the covariance matrix $\mathbf{M} = \mathbf{Z}\mathbf{Z}^T$ by the trace of $\mathbf{M}$ as $\frac{\mathbf{M}}{tr(\mathbf{M})}$. It's clear that $\mathbf{S}$ used in Algorithm I is equal to $\frac{\mathbf{M}}{tr(\mathbf{M})}$, based on Eqn. 14 shown in the previous proof.

C. Orthogonality for Group Based Method

In Section 3.4 of the paper, we argue that group based methods cannot ensure the whole matrix $\mathbf{W} \in \mathbb{R}^{n\times d}$ to be either row or column orthogonal, when $n > d$. Here we provide more details.

configurations	cudnn	cudnn + ONI-T1	cudnn + ONI-T3	cudnn + ONI-T5	cudnn + ONI-T7
$F_h = F_w = 3$, n=d=256, m=256	118.6	122.1	122.9	124.4	125.7
$F_h = F_w = 3$, n=d=256, m=32	15.8	18.3	18.9	19.5	20.8
$F_h = F_w = 3$, n=d=1024, m=32	71.1	81.7	84.3	89.5	94.2
$F_h = F_w = 1$, n=d=256, m=256	28.7	31.5	32.1	33.7	34.6
$F_h = F_w = 1$, n=d=256, m=32	10.1	13	13.6	14.2	15.3
$F_h = F_w = 1$, n=d=1024, m=32	22.2	27.6	29.7	32.9	37.0

Table A. Comparison of wall-clock time (ms). We fix the input with size $h = w = 32$. We evaluate the total wall-clock time of training for each iteration (forward pass + back-propagation pass). Note that 'cudnn + ONI-T5' indicates the 'cudnn' convolution wrapped in our ONI method, using an iteration number of 5.

	δ_{Row}	δ_{Column}
ONI-Full	5.66	0
OLM-G32	8	5.66
OLM-G16	9.85	8.07
OLM-G8	10.58	8.94

Table B. Evaluation for row and column orthogonalization with the group based methods. The entries of proxy matrix $\mathbf{Z} \in \mathbb{R}^{32\times 64}$ are sampled from the Gaussian distribution $N(0, 1)$. We evaluate the row orthogonality $\delta_{Row} = \|\mathbf{W}\mathbf{W}^T - \mathbf{I}\|_F$ and column orthogonality $\delta_{Column} = \|\mathbf{W}^T\mathbf{W} - \mathbf{I}\|_F$. 'OLM-G32' indicates the eigen decomposition based orthogonalization method described in [10], with a group size of 32.

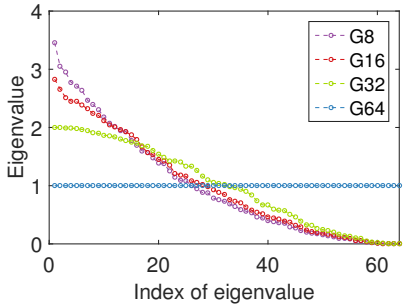


Figure I. The distribution of the eigenvalues of $\mathbf{W}\mathbf{W}^T$ with different group size G. The entries of proxy matrix $\mathbf{Z} \in \mathbb{R}^{64\times 64}$ are sampled from the Gaussian distribution $N(0, 1)$.

We follow the experiments described in Figure.3 of the paper, where we sample the entries of proxy matrix $\mathbf{Z} \in \mathbb{R}^{64\times 32}$ from the Gaussian distribution $N(0, 1)$. We apply the eigen decomposition based orthogonalization method [10] with group size G, to obtain the orthogonalized matrix $\mathbf{W}$. We vary the group size $G \in \{32, 16, 8\}$. We evaluate the corresponding row orthogonality $\delta_{Row} = \|\mathbf{W}\mathbf{W}^T - \mathbf{I}\|_F$ and column orthogonality $\delta_{Column} = \|\mathbf{W}^T\mathbf{W} - \mathbf{I}\|_F$. The results are shown in Table B. We observe that the group based orthogonalization method cannot ensure the whole matrix $\mathbf{W}$ to be either row or column orthogonal, while our ONI can ensure column orthogonality. We also observe that the group based method has degenerated orthogonality, with decreasing group size.

We also conduct an experiment when $n = d$, where we sample the entries of proxy matrix $\mathbf{Z} \in \mathbb{R}^{64\times 64}$ from the Gaussian distribution $N(0, 1)$. We vary the group size $G \in \{64, 32, 16, 8\}$. Note that $G = 64$ represents full orthogonalization. Figure I shows the distribution of the eigenvalues of $\mathbf{W}\mathbf{W}^T$. We again observe that the group based method cannot ensure the whole weight matrix to be row orthogonal. Furthermore, orthogonalization with smaller group size tends to be worse.

D. Comparison of Wall Clock Times

In Section 3.6 of the paper, we show that, given a convolutional layer with filters $\mathbf{W} \in \mathbb{R}^{n\times d\times F_h\times F_w}$ and m mini-batch data $\{\mathbf{x}_i \in \mathbb{R}^{d\times h\times w}\}_{i=1}^m$, the relative computational cost of ONI over the convolutional layer is $\frac{2n}{mhw} + \frac{3Nn^2}{mdhwF_hF_w}$. In this section, we compare the of wall clock time between the convolution wrapping with our ONI and the standard convolution. In this experiment, our ONI is implemented based on Torch [4] and we wrap it to the 'cudnn' convolution [3]. The experiments are run on a TITAN Xp.

We fix the input to size $h = w = 32$, and vary the kernel size (F_h and F_w), the feature dimensions (n and d) and the batch size m. Table A shows the wall clock time under different configurations. We compare the standard 'cudnn' convolution (denoted as 'cudnn') and the 'cudnn' wrapped with our ONI (denoted as 'cudnn + ONI').

We observe that our method introduces negligible computational costs when using a 3×3 convolution, feature dimension $n = d = 256$ and batch size of $m = 256$. Our method may degenerate in efficiency with a smaller kernel size, larger feature dimension and smaller batch size, based on the computational complexity analysis. However, our method (with iteration of 5) 'cudnn + ONI-T5' only costs $1.48\times$ over the standard convolution 'cudnn', under the worst configuration, $F_h = F_w = 1, n = d = 1024$ and m=32.

E. Proof of Theorems

Here we prove the two theorems described in Section 3.5 and 3.6 of the paper.

Theorem 1. *Let* $\hat{\mathbf{h}} = \mathbf{W}\mathbf{x}$, *where* $\mathbf{W}\mathbf{W}^T = \mathbf{I}$ *and* $\mathbf{W} \in \mathbb{R}^{n\times d}$. *Assume: (1)* $\mathbb{E}_{\mathbf{x}}(\mathbf{x}) = \mathbf{0}$, $cov(\mathbf{x}) = \sigma_1^2\mathbf{I}$, *and (2)* $\mathbb{E}_{\frac{\partial\mathcal{L}}{\partial\hat{\mathbf{h}}}}(\frac{\partial\mathcal{L}}{\partial\hat{\mathbf{h}}}) = \mathbf{0}$, $cov(\frac{\partial\mathcal{L}}{\partial\hat{\mathbf{h}}}) = \sigma_2^2\mathbf{I}$. *If* $n = d$, *we have the following properties: (1)* $\|\hat{\mathbf{h}}\| = \|\mathbf{x}\|$; *(2)* $\mathbb{E}_{\hat{\mathbf{h}}}(\hat{\mathbf{h}}) = \mathbf{0}$, $cov(\hat{\mathbf{h}}) = \sigma_1^2\mathbf{I}$; *(3)* $\|\frac{\partial\mathcal{L}}{\partial\mathbf{x}}\| = \|\frac{\partial\mathcal{L}}{\partial\hat{\mathbf{h}}}\|$; *(4)* $\mathbb{E}_{\frac{\partial\mathcal{L}}{\partial\mathbf{x}}}(\frac{\partial\mathcal{L}}{\partial\mathbf{x}}) = \mathbf{0}$, $cov(\frac{\partial\mathcal{L}}{\partial\mathbf{x}}) = \sigma_2^2\mathbf{I}$. *In particular, if* $n < d$, *property (2) and (3) hold; if* $n > d$, *property (1) and (4) hold.*

Proof. Based on $n = d$ and $\mathbf{W}\mathbf{W}^T = \mathbf{I}$, we have that $\mathbf{W}$ is a square orthogonal matrix. We thus have $\mathbf{W}^T\mathbf{W} = \mathbf{I}$. Besides, we have $\frac{\partial\mathcal{L}}{\partial\mathbf{x}} = \frac{\partial\mathcal{L}}{\partial\hat{\mathbf{h}}}\mathbf{W}$[1].

(1) Therefore, we have

$$\|\hat{\mathbf{h}}\|^2 = \hat{\mathbf{h}}^T\hat{\mathbf{h}} = \mathbf{x}^T\mathbf{W}^T\mathbf{W}\mathbf{x} = \mathbf{x}^T\mathbf{x} = \|\mathbf{x}\|^2. \tag{15}$$

We thus get $\|\hat{\mathbf{h}}\| = \|\mathbf{x}\|$.

(2) It's easy to calculate:

$$\mathbb{E}_{\hat{\mathbf{h}}}(\hat{\mathbf{h}}) \quad = \mathbb{E}_{\mathbf{x}}(\mathbf{W}\mathbf{x}) = \mathbf{W}\mathbb{E}_{\mathbf{x}}(\mathbf{x}) = \mathbf{0}. \tag{16}$$

The covariance of $\hat{\mathbf{h}}$ is given by

$$\begin{aligned} cov(\hat{\mathbf{h}}) &= \mathbb{E}_{\hat{\mathbf{h}}}((\hat{\mathbf{h}} - \mathbb{E}_{\hat{\mathbf{h}}}(\hat{\mathbf{h}}))\cdot(\hat{\mathbf{h}} - \mathbb{E}_{\hat{\mathbf{h}}}(\hat{\mathbf{h}}))^T) \\ &= \mathbb{E}_{\mathbf{x}}(\mathbf{W}(\mathbf{x} - \mathbb{E}_{\mathbf{x}}(\mathbf{x}))\cdot(\mathbf{W}(\mathbf{x} - \mathbb{E}_{\mathbf{x}}(\mathbf{x})))^T) \\ &= \mathbf{W}\mathbb{E}_{\mathbf{x}}((\mathbf{x} - \mathbb{E}_{\mathbf{x}}(\mathbf{x}))\cdot(\mathbf{x} - \mathbb{E}_{\mathbf{x}}(\mathbf{x}))^T)\mathbf{W}^T \\ &= \mathbf{W}cov(\mathbf{x})\mathbf{W}^T \\ &= \mathbf{W}\sigma_1^2\mathbf{I}\mathbf{W}^T \\ &= \sigma_1^2\mathbf{W}\mathbf{W}^T \\ &= \sigma_1^2. \end{aligned} \tag{17}$$

(3) Similar to the proof of (1),

$$\begin{aligned} \|\frac{\partial\mathcal{L}}{\partial\mathbf{x}}\|^2 = \frac{\partial\mathcal{L}}{\partial\mathbf{x}}\frac{\partial\mathcal{L}}{\partial\mathbf{x}}^T &= \frac{\partial\mathcal{L}}{\partial\hat{\mathbf{h}}}\mathbf{W}\mathbf{W}^T\frac{\partial\mathcal{L}}{\partial\hat{\mathbf{h}}}^T \\ &= \frac{\partial\mathcal{L}}{\partial\hat{\mathbf{h}}}\frac{\partial\mathcal{L}}{\partial\hat{\mathbf{h}}}^T = \|\frac{\partial\mathcal{L}}{\partial\hat{\mathbf{h}}}\|^2. \end{aligned} \tag{18}$$

We thus have $\|\frac{\partial\mathcal{L}}{\partial\mathbf{x}}\| = \|\frac{\partial\mathcal{L}}{\partial\hat{\mathbf{h}}}\|$.

(4) Similar to the proof of (2), we have

$$\mathbb{E}_{\frac{\partial\mathcal{L}}{\partial\mathbf{x}}}(\frac{\partial\mathcal{L}}{\partial\mathbf{x}}) = \mathbb{E}_{\frac{\partial\mathcal{L}}{\partial\hat{\mathbf{h}}}}(\frac{\partial\mathcal{L}}{\partial\hat{\mathbf{h}}}\mathbf{W}) = \mathbb{E}_{\frac{\partial\mathcal{L}}{\partial\hat{\mathbf{h}}}}(\frac{\partial\mathcal{L}}{\partial\hat{\mathbf{h}}})\mathbf{W} = \mathbf{0}. \tag{19}$$

[1]We follow the common setup where the vectors are column vectors when their derivations are row vectors.

The covariance of $\frac{\partial\mathcal{L}}{\partial\mathbf{x}}$ is given by

$$\begin{aligned} cov(\frac{\partial\mathcal{L}}{\partial\mathbf{x}}) &= \mathbb{E}_{\frac{\partial\mathcal{L}}{\partial\mathbf{x}}}((\frac{\partial\mathcal{L}}{\partial\mathbf{x}} - \mathbb{E}_{\frac{\partial\mathcal{L}}{\partial\mathbf{x}}}(\frac{\partial\mathcal{L}}{\partial\mathbf{x}}))^T(\frac{\partial\mathcal{L}}{\partial\mathbf{x}} - \mathbb{E}_{\frac{\partial\mathcal{L}}{\partial\mathbf{x}}}(\frac{\partial\mathcal{L}}{\partial\mathbf{x}}))) \\ &= \mathbb{E}_{\frac{\partial\mathcal{L}}{\partial\hat{\mathbf{h}}}}(((\frac{\partial\mathcal{L}}{\partial\hat{\mathbf{h}}} - \mathbb{E}_{\frac{\partial\mathcal{L}}{\partial\hat{\mathbf{h}}}}(\frac{\partial\mathcal{L}}{\partial\hat{\mathbf{h}}}))\mathbf{W})^T(\frac{\partial\mathcal{L}}{\partial\hat{\mathbf{h}}} - \mathbb{E}_{\frac{\partial\mathcal{L}}{\partial\hat{\mathbf{h}}}}(\frac{\partial\mathcal{L}}{\partial\hat{\mathbf{h}}}))\mathbf{W}) \\ &= \mathbf{W}^T\mathbb{E}_{\frac{\partial\mathcal{L}}{\partial\hat{\mathbf{h}}}}((\frac{\partial\mathcal{L}}{\partial\hat{\mathbf{h}}} - \mathbb{E}_{\frac{\partial\mathcal{L}}{\partial\hat{\mathbf{h}}}}(\frac{\partial\mathcal{L}}{\partial\hat{\mathbf{h}}}))^T(\frac{\partial\mathcal{L}}{\partial\hat{\mathbf{h}}} - \mathbb{E}_{\frac{\partial\mathcal{L}}{\partial\hat{\mathbf{h}}}}(\frac{\partial\mathcal{L}}{\partial\hat{\mathbf{h}}})))\mathbf{W} \\ &= \mathbf{W}^T cov(\frac{\partial\mathcal{L}}{\partial\hat{\mathbf{h}}})\mathbf{W} \\ &= \mathbf{W}^T\sigma_2^2\mathbf{I}\mathbf{W} \\ &= \sigma_2^2\mathbf{W}^T\mathbf{W} \\ &= \sigma_2^2. \end{aligned} \tag{20}$$

Besides, if $n < d$, it is easy to show that properties (2) and (3) hold; if $n > d$, properties (1) and (4) hold. □

Theorem 2. *Let* $\mathbf{h} = max(0, \mathbf{W}\mathbf{x})$, *where* $\mathbf{W}\mathbf{W}^T = \sigma^2\mathbf{I}$ *and* $\mathbf{W} \in \mathbb{R}^{n\times d}$. *Assume* $\mathbf{x}$ *is a normal distribution with* $\mathbb{E}_{\mathbf{x}}(\mathbf{x}) = \mathbf{0}$, $cov(\mathbf{x}) = \mathbf{I}$. *Denote the Jacobian matrix as* $\mathbf{J} = \frac{\partial\mathbf{h}}{\partial\mathbf{x}}$. *If* $\sigma^2 = 2$, *we have* $\mathbb{E}_{\mathbf{x}}(\mathbf{J}\mathbf{J}^T) = \mathbf{I}$.

Proof. For denotation, we use $\mathbf{A}_{i:}$ and $\mathbf{A}_{:j}$ to represent the i-th row and the j-th column of $\mathbf{A}$, respectively. Based on $\mathbf{W}\mathbf{W}^T = \sigma^2\mathbf{I}$, we obtain $\mathbf{W}_{i:}(\mathbf{W}_{j:})^T = 0$ for $i \neq j$ and $\mathbf{W}_{i:}(\mathbf{W}_{i:})^T = \sigma^2$ otherwise. Let $\hat{\mathbf{h}} = \mathbf{W}\mathbf{x}$. This yields

$$\mathbf{J}_{i:} = \frac{\partial\mathbf{h}_i}{\partial\hat{\mathbf{h}}_i}\frac{\partial\hat{\mathbf{h}}_i}{\mathbf{x}} = \frac{\partial\mathbf{h}_i}{\partial\hat{\mathbf{h}}_i}\mathbf{W}_{i:}. \tag{21}$$

Denote $\mathbf{M} = \mathbf{J}\mathbf{J}$. This yields the following equation from Eqn. 21:

$$\begin{aligned} \mathbf{M}_{ij} &= \mathbf{J}_{i:}(\mathbf{J}_{j:})^T \\ &= \frac{\partial\mathbf{h}_i}{\partial\hat{\mathbf{h}}_i}\mathbf{W}_{i:}(\mathbf{W}_{j:})^T\frac{\partial\mathbf{h}_j}{\partial\hat{\mathbf{h}}_j} \\ &= \frac{\partial\mathbf{h}_i}{\partial\hat{\mathbf{h}}_i}\frac{\partial\mathbf{h}_j}{\partial\hat{\mathbf{h}}_j}(\mathbf{W}_{i:}(\mathbf{W}_{j:})^T). \end{aligned} \tag{22}$$

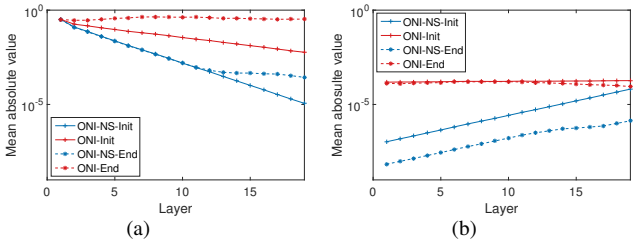
If $i \neq j$, we obtain $\mathbf{M}_{ij} = 0$. For $i = j$, we have:

$$\mathbf{M}_{ii} = \mathbf{J}_{i:}(\mathbf{J}_{i:})^T = (\frac{\partial\mathbf{h}_i}{\partial\hat{\mathbf{h}}_i})^2\sigma^2 = 1(\hat{\mathbf{h}}_i > 0)\sigma^2, \tag{23}$$

where $1(\hat{\mathbf{h}}_i > 0)$ indicates 1 for $\hat{\mathbf{h}}_i > 0$ and 0 otherwise. Since $\mathbf{x}$ is a normal distribution with $\mathbb{E}_{\mathbf{x}}(\mathbf{x}) = \mathbf{0}$ and $cov(\mathbf{x}) = \mathbf{I}$, we have that $\hat{\mathbf{h}}$ is also a normal distribution, with $\mathbb{E}_{\hat{\mathbf{h}}}(\hat{\mathbf{h}}) = \mathbf{0}$ and $cov(\hat{\mathbf{h}}) = \mathbf{I}$, based on Theorem 1. We thus obtain

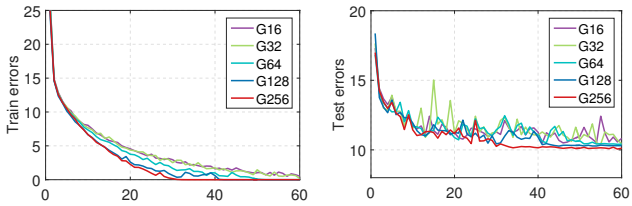
$$\mathbb{E}_{\mathbf{x}}(\mathbf{M}_{ii}) = \mathbb{E}_{\hat{\mathbf{h}}_i}1(\hat{\mathbf{h}}_i > 0)\sigma^2 = \frac{1}{2}\sigma^2. \tag{24}$$

Therefore, $\mathbb{E}_{\mathbf{x}}(\mathbf{J}\mathbf{J}^T) = \mathbb{E}_{\mathbf{x}}(\mathbf{M}) = \frac{\sigma^2}{2}\mathbf{I} = \mathbf{I}$. □



(a) (b)

Figure II. The magnitude of the activation and gradient for each layer on a 20-layer MLP. (a) The mean absolute value of the activation $\sigma_{\mathbf{x}}$ for each layer; and (b) the mean absolute value of the gradient $\sigma_{\frac{\partial \mathcal{L}}{\partial \mathbf{h}}}$ for each layer.



(a) Train Error (b) Test Error

Figure III. Effects of group size G in proposed 'ONI'. We evaluate the (a) training and (b) test errors on a 10-layer MLP.

F. Details and More Experimental Results on Discriminative Classification

F.1. MLPs on Fashion-MNIST

Details of Experimental Setup Fashion-MNIST consists of $60k$ training and $10k$ test images. Each image has a size of 28×28, and is associated with a label from one of 10 classes. We use the MLP with varying depths and the number of neurons in each layer is 256. We use ReLU [17] as the nonlinearity. The weights in each layer are initialized by random initialization [13] and we use an iteration number of 5 for ONI, unless otherwise stated. We employ stochastic gradient descent (SGD) optimization with a batch size of 256, and the learning rates are selected, based on the validation set ($5,000$ samples from the training set), from $\{0.05, 0.1, 0.5, 1\}$.

F.1.1 Vanished Activations and Gradients

In Section 4.1.1 of the paper, we observe that the deeper neural networks with orthogonal weight matrices are difficult to train without scaling them by a factor of $\sqrt{2}$. We argue the main reason for this is that the activation and gradient are exponentially vanished. Here we provide the details.

We evaluate the mean absolute value of the activation: $\sigma_{\mathbf{x}} = \sum_{i=1}^{m}\sum_{j=1}^{d}|\mathbf{x}_{ij}|$ for the layer-wise input $\mathbf{x} \in \mathbb{R}^{m\times d}$, and the mean absolute value of the gradient: $\sigma_{\frac{\partial \mathcal{L}}{\partial \mathbf{h}}} = \sum_{i=1}^{m}\sum_{j=1}^{n}|\frac{\partial \mathcal{L}}{\partial \mathbf{h}}_{ij}|$ for the layer-wise gradient $\frac{\partial \mathcal{L}}{\partial \mathbf{h}} \in \mathbb{R}^{m\times n}$. Figure II show the results on the 20-layer MLP. 'ONI-NS-Init' ('ONI-NS-End') indicates the ONI method without scaling a factor of $\sqrt{2}$ during initialization (the end of training). We observe that 'ONI-NS' suffers from a vanished activation and gradient during training. Our 'ONI' with a scaling factor of $\sqrt{2}$ has no vanished gradient.

F.1.2 Effects of Groups

We further explore the group based orthogonalization method [10] on ONI. We vary the group size G in $\{16, 32, 64, 128, 256\}$, and show the results in Figure III. We observe that our ONI can achieve slightly better performance with an increasing group size. The main reason for this is that the group based orthogonalization cannot ensure the whole weight matrix to be orthogonal.

F.2. CNNs on CIFAR10

We use the official training set of $50,000$ images and the standard test set of $10,000$ images. The data preprocessing and data augmentation follow the commonly used mean&std normalization and flip translation, as described in [8].

F.2.1 VGG-Style Networks

Details of Network Architectures The network starts with a convolutional layer of $32k$ filters, where k is the varying width based on different configurations. We then sequentially stack three blocks, each of which has g convolutional layers, and the corresponding convolutional layers have a filter numbers of $32k$, $64k$ and $128k$, respectively, and feature maps sizes of 32×32, 16×16 and 8×8, respectively. We use the first convolution in each block with stride 2 to carry out spatial sub-sampling for feature maps. The network ends with global average pooling and follows a linear transformation. We vary the depth with g in $\{2, 3, 4\}$ and the width with k in $\{1, 2, 3\}$.

Experimental Setup We use SGD with a momentum of 0.9 and batch size of 128. The best initial learning rate is chosen from $\{0.01, 0.02, 0.05\}$ over the validation set of 5,000 samples from the training set, and we divide the learning rate by 5 at 80 and 120 epochs, ending the training at 160 epochs. For 'OrthReg', we report the best results using a regularization coefficient λ in $\{0.0001, 0.0005\}$. For 'OLM', we use the group size of $G = 64$ and full orthogonalization, and report the best result.

Training Performance In Section 4.1.2 of the paper, we mention 'ONI' and 'OLM-$\sqrt{2}$' converge faster than other baselines, in terms of training epochs. Figure IV shows the training curves under different configurations (depth and width). It's clear that 'ONI' and 'OLM-$\sqrt{2}$' converge faster than other baselines under all network configurations, in terms of training epochs. The results support our conclusion that maintaining orthogonality can benefit optimization.

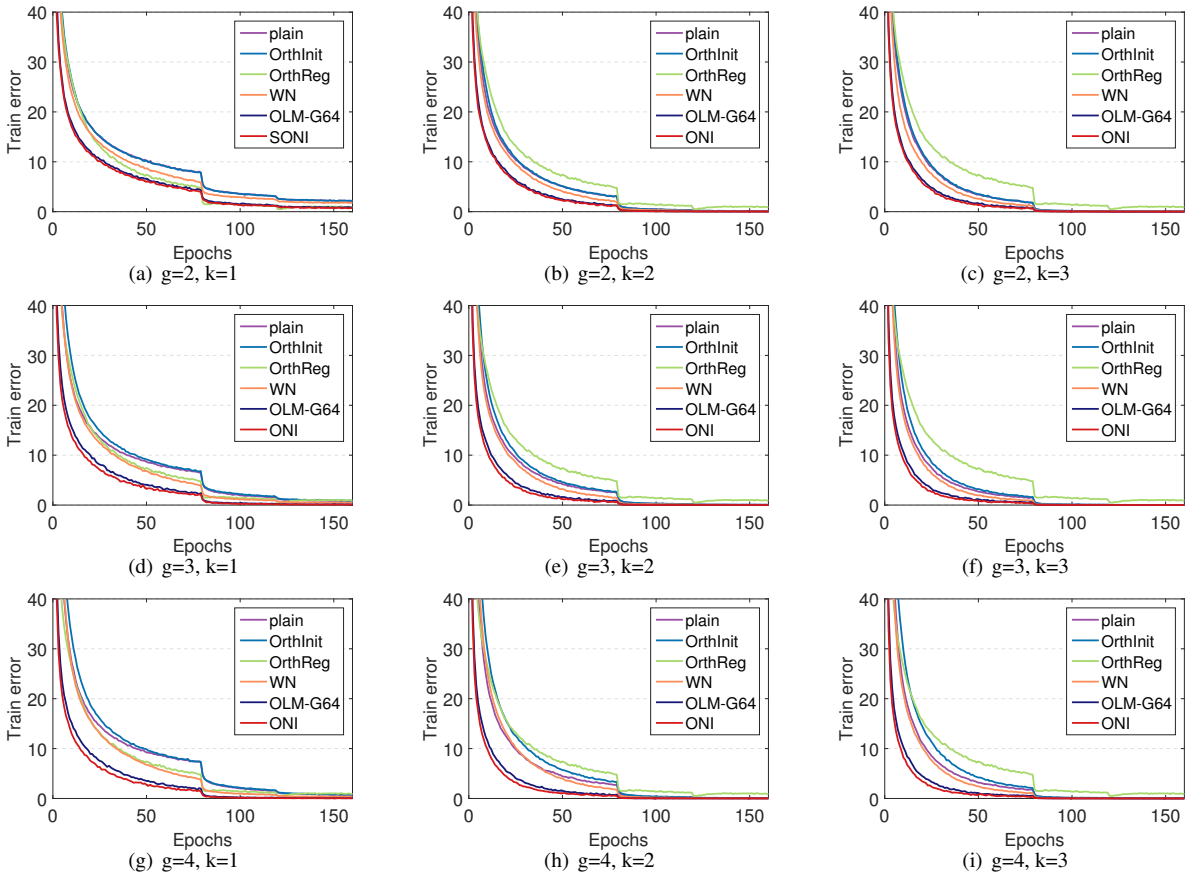


Figure IV. Comparison of training errors on VGG-style networks for CIFAR-10 image classification. From (a) to (i), we vary the depth $3g + 2$ and width $32k$, with $g \in \{2, 3, 4\}$ and $k \in \{1, 2, 3\}$.

F.2.2 Residual Network without Batch Normalization

Here we provide the details of the experimental setups and training performance of the experiments on a 110-layer residual network [8] without batch normalization (BN) [12], described in Section 4.1.2 of the paper.

Experimental Setups We run the experiments on one GPU. We apply SGD with a batch size of 128, a momentum of 0.9 and a weight decay of 0.0001. We set the initial learning rate to 0.1 by default, and divide it by 10 at 80 and 120 epochs, and terminate the training at 160 epochs. For Xavier Init [5, 1], we search the initial learning rate from $\{0.1, 0.01, 0.001\}$ and report the best result. For group normalization (GN) [23], we search the group size from $\{64, 32, 16\}$ and report the best result. For our ONI, we use the data-dependent initialization methods used in [21] to initial the learnable scale parameters.

For small batch size experiments, we train the network with an initial learning rate following the linear learning rate scaling rule [7], to adapt the batch size.

Training Performance Figure V (a) and (b) show the training curve and test curve respectively. We observe that

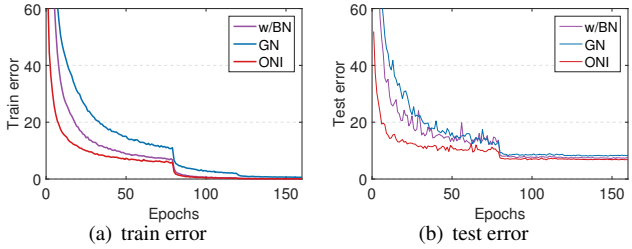


Figure V. Training performance comparison on 110-layer residual network without batch normalization for CIFAR-10 dataset. 'w/BN' indicates with BN. We show the (a) training error with respect to the epochs and (b) test error with respect to epochs.

'ONI' converges significantly faster than 'BN' and 'GN', in terms of training epochs.

F.3. Details of Experimental Setup on ImageNet

ImageNet-2012 consists of 1.28M images from 1,000 classes [19]. We use the official 1.28M labeled images provided for training and evaluate the top-1 and top-5 test classification errors on the validation set, with 50k images.

We keep almost all the experimental settings the same as the publicly available PyTorch implementation [18]: we apply SGD with a momentum of 0.9, and a weight decay of 0.0001; We train over 100 epochs in total and set the initial learning rate to 0.1, lowering it by a factor of 10 at epochs 30, 60 and 90. For 'WN' and 'ONI', we don't us weight decay on the learnable scalar parameters.

VGG Network We run the experiments on one GPU, with a batch size of 128. Apart from our 'ONI', all other methods ('plain', 'WN', 'OrthInit' and 'OrthReg') suffer from difficulty in training with a large learning rate of 0.1. We thus run the experiments with initial learning rates of $\{0.01, 0.05\}$ for these, and report the best result.

Residual Network We run the experiments on one GPU for the 18- and 50-layer residual network, and two GPUs for the 101-layer residual network. We use a batch size of 256. Considering that 'ONI' can improve the optimization efficiency, as shown in the ablation study on ResNet-18, we run the 50- and 101-layer residual network with a weight decay of $\{0.0001, 0.0002\}$ and report the best result from these two configurations for each method, for a more fair comparison.

F.4. Ablation Study on Iteration Number

We provide the details of the training performance for ONI on Fashion MNIST in Figure VI. We vary T, for a range of 1 to 7, and show the training (Figure VI (a)) and test (Figure VI (b)) errors with respect to the training epochs. We also provide the distribution of the singular values of the orthogonalized weight matrix $\mathbf{W}$, using our ONI with different iteration numbers T. Figure VI (c) shows the results from the first layer, at the 200th iteration. We also obtain similar observations for other layers.

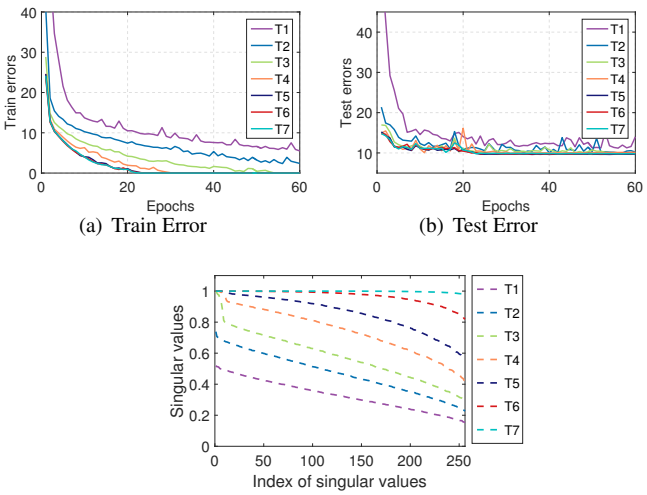
G. Details on Training GANs

For completeness, we provide descriptions of the main concepts used in the paper, as follows.

Inception score (IS) Inception score (IS) (the higher the better) was introduced by Salimans *et al*. [20]:

$$I_{\mathbf{D}} = exp(E_{\mathbf{D}}[KL(p(y|\mathbf{x})\|p(y))]), \tag{25}$$

where $KL(\cdot\|\cdot)$ denotes the Kullback-Leibler Divergence, $p(y)$ is approximated by $\frac{1}{N}\sum_{i=1}^{N} p(y|\mathbf{x}_i)$ and $p(y|\mathbf{x}_i)$ is the trained Inception model [22]. Salimans *et al*. [20] reported that this score is highly correlated with subjective human judgment of image quality. Following [20] and [16], we calculate the score for 5000 randomly generated examples from each trained generator to evaluate IS. We repeat 10 times and report the average and the standard deviation of IS.



(a) Train Error (b) Test Error

(c) Distribution of Singular Values

Figure VI. Effects of the iteration number T in proposed 'ONI'. We evaluate the training errors on a 10-layer MLP. We show (a) the training errors; (b) the test errors and (c) the distribution of the singular values of the orthogonalized weight matrix $\mathbf{W}$ from the first layer, at the 200th iteration.

Fréchet Inception Distance (FID) [9] Fréchet inception distance (FID) [9] (the lower the better) is another measure for the quality of the generated examples that uses second-order information from the final layer of the inception model applied to the examples. The Fréchet distance itself is a 2-Wasserstein distance between two distributions p_1 and p_2, assuming they are both multivariate Gaussian distributions:

$$F(p_1, p_2) = \|\mu_{p_1} - \mu_{p_2}\|_2^2 + tr(C_{p_1} + C_{p_2} - 2(C_{p_1}C_{p_2})^{\frac{1}{2}}), \tag{26}$$

where $\{\mu_{p_1}, C_{p_1}\}$, $\{\mu_{p_2}, C_{p_2}\}$ are the mean and covariance of samples from generated p_1 and p_2, respectively, and $tr(\cdot)$ indicates the trace operation. We calculate the FID between the 10K test examples (true distribution) and the 5K randomly generated samples (generated distribution).

GAN with Non-saturating Loss The standard non-saturating function for the adversarial loss is:

$$\mathcal{L}(G, D) = \mathbb{E}_{\mathbf{x}\sim q(\mathbf{x})}[logD(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[1 - logD(G(\mathbf{z}))], \tag{27}$$

where $q(\mathbf{x})$ is the distribution of the real data, $\mathbf{z} \in \mathbb{R}^{d_z}$ is a latent variable, $p(\mathbf{z})$ is the standard normal distribution $N(0, I)$, and G is a deterministic generator function. d_z is set to 128 for all experiments. Based on the suggestion in [6, 16], we use the alternate cost $-\mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[logD(G(\mathbf{z}))]$ to update G, while using the original cost defined in Eqn. 27 for updating D.

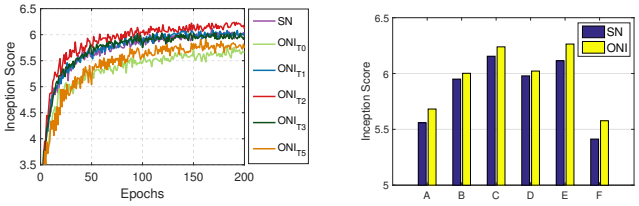
GAN with Hinge Loss The hinge loss for adversarial learning is:

$$\mathcal{L}_D(\hat{G}, D) = \mathbb{E}_{\mathbf{x}\sim q(\mathbf{x})}[max(0, 1 - D(\mathbf{x}))] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[max(0, 1 + D(G(\mathbf{z})))] \tag{28}$$

$$\mathcal{L}_G(G, \hat{D}) = -\mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}\hat{D}(G(\mathbf{z})) \tag{29}$$

Setting	α	β_1	β_2	n_{dis}
A	0.0001	0.5	0.9	5
B	0.0001	0.5	0.999	1
C	0.0002	0.5	0.999	1
D	0.001	0.5	0.9	5
E	0.001	0.5	0.999	5
F	0.001	0.9	0.999	5

Table C. Hyper-parameter settings in stability experiments on DCGAN, following [16].



(a) (b)

Figure VII. Comparison of SN and ONI on DCGAN. (a) The IS with respect to training epochs. (b)The stability experiments on the six configurations described in [16].

for the discriminator and the generator, respectively. This type of loss has already been used in [14, 16, 24, 2].

Our code is implemented in PyTorch [18] and the trained Inception model is from the official models in PyTorch [18]. The IS and FID for the real training data are 10.20 ± 0.13 and 3.07 respectively. Note that we do not use the learnable scalar in any the GAN experiment, and set $\sigma = 1$ in ONI, for more consistent comparisons with SN.

G.1. Experiments on DCGAN

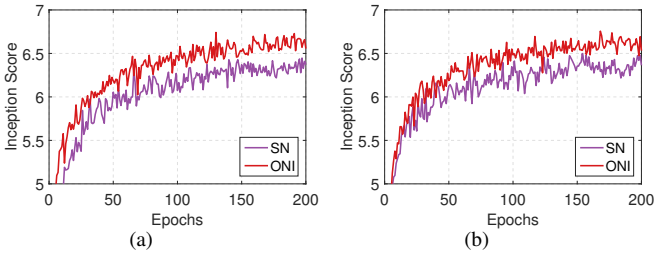
The DCGAN architecture follows the configuration in [16], and we provide the details in Table D for completeness. The spectral normalizaiton (SN) and our ONI are only applied on the discriminator, following the experimental setup in the SN paper [16].

Figure VII (a) shows the IS of SN and ONI when varying Newton's iteration number T from 0 to 5. We obtain the same observation as the FID evaluation, shown in Section 4.2 of the paper.

As discussed in Section 4.2 the paper, we conduct experiments to validate the stability of our proposed ONI under different experimental configurations, following [16]. Table C shows the corresponding configurations (denoted by A-F) when varying the learning rate α, first momentum β_1, second momentum β_2, and the number of updates of the discriminator per update of the generator n_{dis}. The results evaluated by IS are shown in Figure VII (b). We observe that our ONI is consistently better than SN under the IS evaluation.

G.2. Implementation Details of ResNet-GAN

The ResNet architecture also follows the configuration in [16], and we provide the details in Table E for completeness.



(a) (b)

Figure VIII. Comparison of SN and ONI on ResNet GAN. We show the IS with respect to training epochs using (a) the non-saturating loss and (b) the hinge loss.

The SN and our ONI are only applied on the discriminator, following the experimental setup in the SN paper [16].

We provide the results of SN and ONI in Figure VIII, evaluated by IS.

G.3. Qualitative Results of GAN

We provide the generated images in Figure IX, X and XI. Note that we don't hand-pick the images, and show all the results at the end of the training.

References

[1] Nils Bjorck, Carla P Gomes, Bart Selman, and Kilian Q Weinberger. Understanding batch normalization. In *NeurIPS*. 2018. 6

[2] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *ICLR*, 2019. 8

[3] Sharan Chetlur, Cliff Woolley, Philippe Vandermersch, Jonathan Cohen, John Tran, Bryan Catanzaro, and Evan Shelhamer. cudnn: Efficient primitives for deep learning. *CoRR*, abs/1410.0759, 2014. 3

[4] R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, 2011. 3

[5] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010. 6

[6] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*. 2014. 7

[7] Priya Goyal, Piotr Dollár, Ross B. Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: training imagenet in 1 hour. *CoRR*, abs/1706.02677, 2017. 6

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 5, 6

[9] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NeurIPS*. 2017. 7

(a) Generator

$z \in \mathbb{R}^{128} \sim \mathcal{N}(0, I)$
4×4, stride=1 deconv. BN 512 ReLU $\rightarrow 4 \times 4 \times 512$
4×4, stride=2 deconv. BN 256 ReLU
4×4, stride=2 deconv. BN 128 ReLU
4×4, stride=2 deconv. BN 64 ReLU
3×3, stride=1 conv. 3 Tanh

(b) Discriminator

RGB image $x \in \mathbb{R}^{32 \times 32 \times 3}$
3×3, stride=1 conv 64 lReLU
4×4, stride=2 conv 64 lReLU
3×3, stride=1 conv 128 lReLU
4×4, stride=2 conv 128 lReLU
3×3, stride=1 conv 256 lReLU
4×4, stride=2 conv 256 lReLU
3×3, stride=1 conv 512 lReLU
dense $\rightarrow 1$

Table D. DCGAN architectures for CIFAR10 dataset in our experiments. 'lReLU' indicates the leaky ReLU [15] and its slope is set to 0.1.

(a) Generator

$z \in \mathbb{R}^{128} \sim \mathcal{N}(0, I)$
dense, $4 \times 4 \times 128$
ResBlock up 128
ResBlock up 128
ResBlock up 128
BN, ReLU, 3×3 conv, 3 Tanh

(b) Discriminator

RGB image $x \in \mathbb{R}^{32 \times 32 \times 3}$
ResBlock down 128
ResBlock down 128
ResBlock 128
ResBlock 128
ReLU
Global sum pooling
dense $\rightarrow 1$

Table E. ResNet architectures for CIFAR10 dataset in our experiments. We use the same ResBlock as the SN paper [16].

[10] Lei Huang, Xianglong Liu, Bo Lang, Adams Wei Yu, Yongliang Wang, and Bo Li. Orthogonal weight normalization: Solution to optimization over multiple dependent stiefel manifolds in deep neural networks. In *AAAI*, 2018. 3, 5

[11] Lei Huang, Yi Zhou, Fan Zhu, Li Liu, and Ling Shao. Iterative normalization: Beyond standardization towards efficient whitening. In *CVPR*, 2019. 2

[12] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 6

[13] Yann LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. Effiicient backprop. In *Neural Networks: Tricks of the Trade*, 1998. 5

[14] Jae Hyun Lim and Jong Chul Ye. Geometric gan. *CoRR*, abs/1705.02894, 2017. 8

[15] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013. 9

[16] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *ICLR*, 2018. 7, 8, 9, 10, 11

[17] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010. 5

[18] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NeurIPS Autodiff Workshop*, 2017. 7, 8

[19] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 6

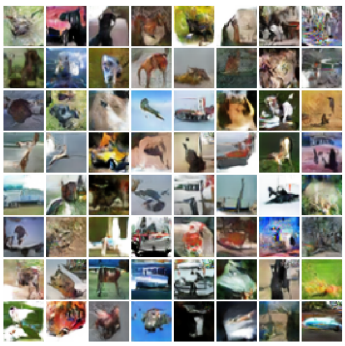
[20] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, and Xi Chen. Improved techniques for training gans. In *NeurIPS*, 2016. 7

[21] Tim Salimans and Diederik P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *NeurIPS*, 2016. 6

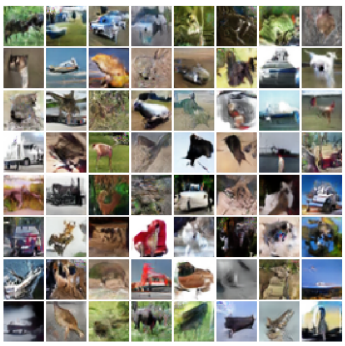
[22] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 7

[23] Yuxin Wu and Kaiming He. Group normalization. In *ECCV*, 2018. 6

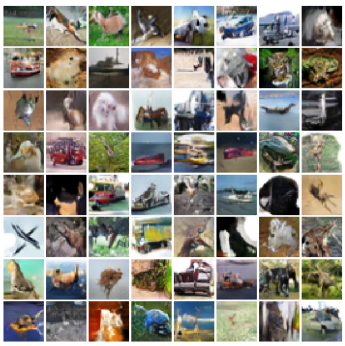
[24] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *ICML*, 2019. 8



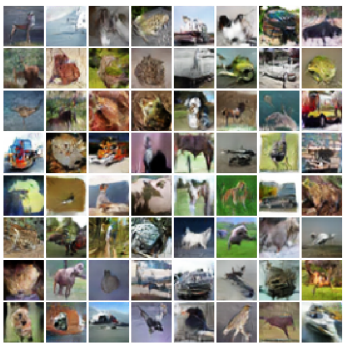
(a) $T = 0$



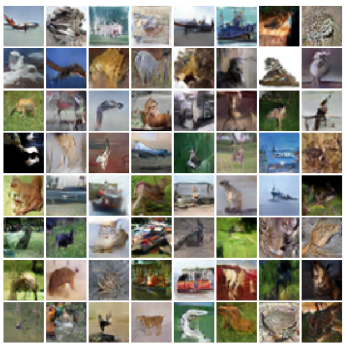
(b) $T = 1$



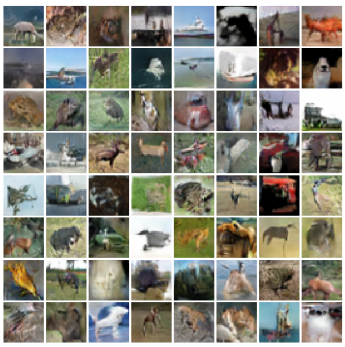
(c) $T = 2$



(d) $T = 3$

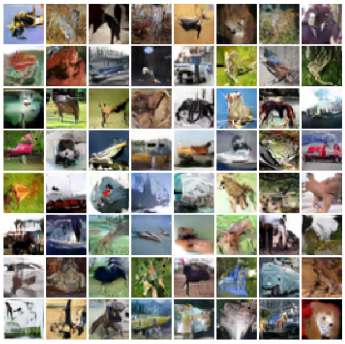


(e) $T = 4$

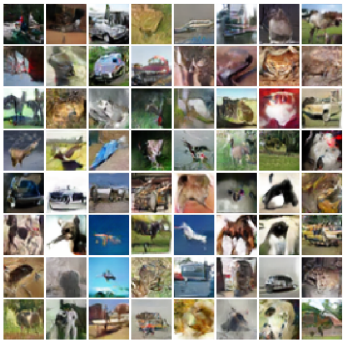


(f) $T = 5$

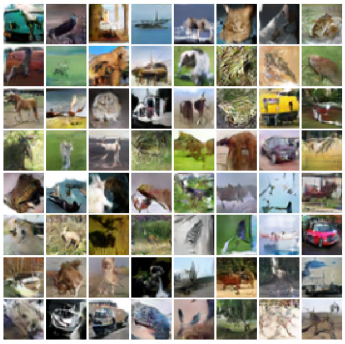
Figure IX. Generated images for CIFAR-10 by our ONI with different iterations, using DCGAN [16].



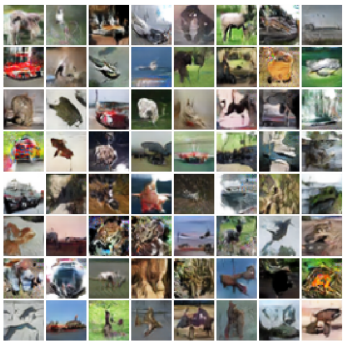
(a) SN-A



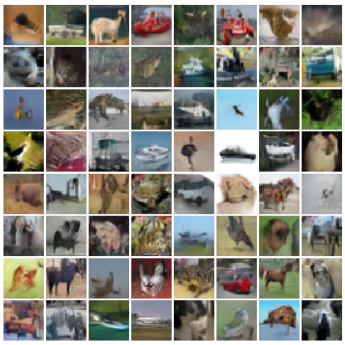
(b) SN-B



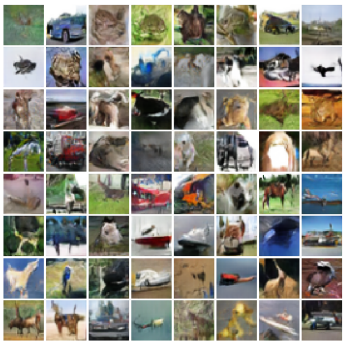
(c) SN-C



(d) ONI-A

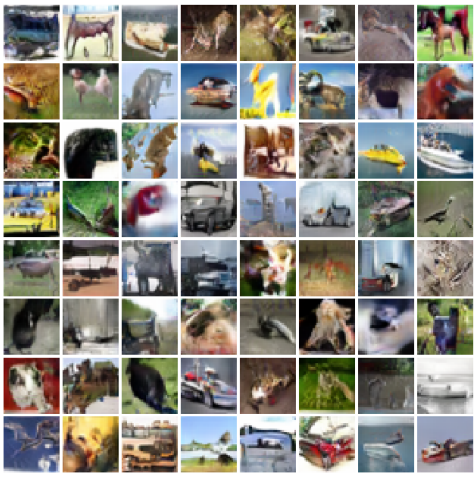


(e) ONI-B

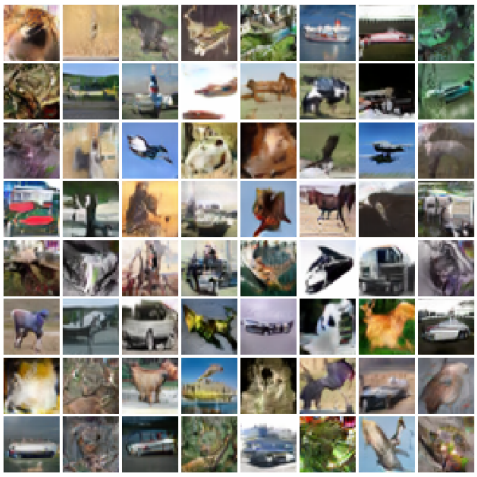


(f) ONI-C

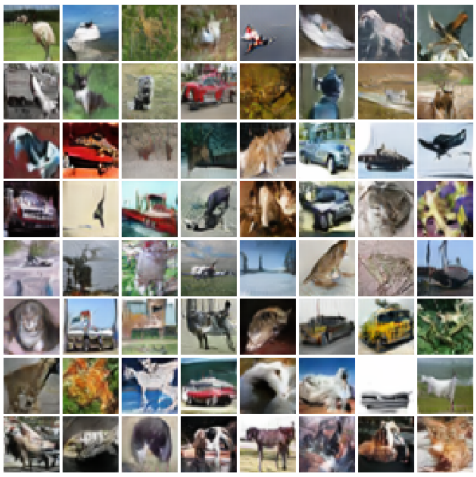
Figure X. Generated images for CIFAR-10 by SN and ONI, using DCGAN [16]. We show the results of SN and ONI, with configuration A, B and C.



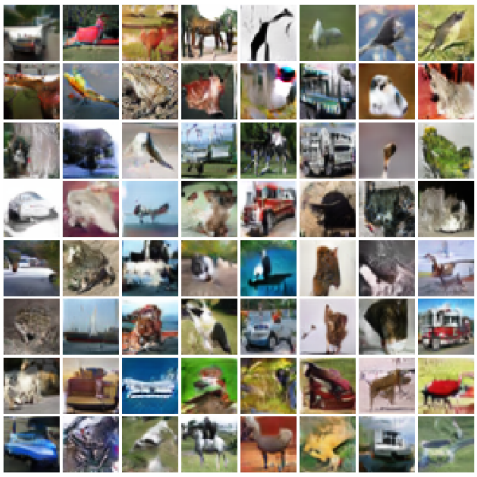
(a) SN with non-satruating loss



(b) SN with hinge loss



(c) ONI with non-satruating loss



(d) ONI with hinge loss

Figure XI. Generated images for CIFAR-10 by SN and ONI, using ResNet [16]. We show the results of SN and ONI, with the non-satruating and hinge loss.