# Improving Generalization of Deep Neural Networks by Optimum Shifting

**Yuyan Zhou[1], Ye Li[1*], Lei Feng[2], Sheng-Jun Huang[1]**

[1]MIIT Key Laboratory of Pattern Analysis and Machine Intelligence, College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China.
[2]Information Systems Technology and Design Pillar, Singapore University of Technology and Design
{yuyanzhou, yeli20, huangsj}@nuaa.edu.cn, lfengqaq@gmail.com

## Abstract

Recent studies showed that the generalization of neural networks is correlated with the sharpness of the loss landscape, and flat minima suggests a better generalization ability than sharp minima. In this paper, we propose a novel method called optimum shifting, which changes the parameters of a neural network from a sharp minimum to a flatter one while maintaining the same training loss value. Our method is based on the observation that when the input and output of a neural network are fixed, the matrix multiplications within the network can be treated as systems of under-determined linear equations, enabling adjustment of parameters in the solution space, which can be simply accomplished by solving a constrained optimization problem. Furthermore, we introduce a practical stochastic optimum shifting technique utilizing the Neural Collapse theory to reduce computational costs and provide more degrees of freedom for optimum shifting. Extensive experiments (including classification and detection) with various deep neural network architectures on benchmark datasets demonstrate the effectiveness of our method.

## Introduction

Deep neural networks (DNNs) are powerful and have shown remarkable results in various fields, including computer vision (Goodfellow et al. 2020; Zhou et al. 2023; Sohl-Dickstein et al. 2015; Lin et al. 2023; Shen et al. 2023; Guan et al. 2024a; Hu et al. 2024b; Sun 2024), natural language processing (OpenAI 2023; Wang et al. 2024; Zhou et al. 2024; Vaswani et al. 2017) and AIGC tasks (Lin et al. 2024b; Shen et al. 2024b; Shen and Tang 2024; Hu et al. 2024a). Contemporary deep learning approaches formulate the learning problem as an optimization problem and utilize stochastic gradient descent and its variants to minimize the loss function (Lin et al. 2024a; Guan et al. 2024b; Shen et al. 2024a; Wu et al. 2023; Sun et al. 2024). Nowadays, DNNs are generally over-parameterized and capable of providing a larger hypothesis space with normally better solutions having smaller training errors. However, this expansive hypothesis space is concurrently populated with different minima, each characterized by distinct generalization abilities. Recent studies have shown that the generalization of DNNs is correlated with the sharpness of the loss landscape, and flat minima suggest a better generalization ability than sharp minima (Keskar et al. 2016; Neyshabur 2017; Hochreiter and Schmidhuber 1994; Keskar et al. 2017; Chaudhari et al. 2019; Gatmiry et al. 2023). Existing works on sharpness minimization mainly focus on penalty-based methods (Foret et al. 2021; Kwon et al. 2021; Zhao, Zhang, and Hu 2022). However, in this work, we propose a penalty-free sharpness minimization technique aimed at addressing a fundamental question below:

*Can we change the solution of a DNN from one point to a flatter one while maintaining the same training loss value?*

In this paper, we propose a method called optimum shifting (OS) to attain this objective. This approach is based on the matrix multiplication within the neural network:

$$\boldsymbol{A}\boldsymbol{V} = \boldsymbol{Z}, \tag{1}$$

where $\boldsymbol{A} \in \mathbb{R}^{batch \times m}$ represents the input matrix, and $\boldsymbol{V} \in \mathbb{R}^{m \times n}$ denotes the parameters in the neural network's linear layer. Consider it a set of linear equations, and the parameter $\boldsymbol{V}$ can be modified in the solution space. Assume that the rows in $\boldsymbol{A}$ are independent without loss of generality, if the equation $\boldsymbol{A}\boldsymbol{V} = \boldsymbol{Z}$ is under-determined, it will have infinite solutions for $\boldsymbol{V}$. This property allows us to change the parameters of the neural network from the current point $\boldsymbol{V}$ in the solution space to another point $\boldsymbol{V}^*$ by minimizing the sharpness (i.e., the trace of its Hessian (Gatmiry et al. 2023; Yao et al. 2020; Blanc et al. 2020)), which can be calculated by solving a simple constrained optimization problem.

The main challenge for the above method is that it requires keeping training loss unchanged across all training samples. Using the entire training set for OS demands a significantly large memory capacity and huge computational costs. Moreover, OS in the whole dataset results in a less under-determined input matrix, which limits the degrees of freedom for optimum shifting. To overcome these challenges, we take inspiration from stochastic gradient descent, which uses a small batch to conduct gradient descent, thereby reducing computational costs. We propose stochastic optimum shifting, which performs OS on a small batch of data. In this way, the generalization ability of the neural network can be improved and the computational costs can be decreased. According to the theory of the Neural Collapse (Zhu et al. 2021; Tirer and Bruna 2022; Zhou et al.

*Corresponding author.

2022), *if the loss is unchanged in a small batch of training data (typically, "$batch \geq n$" where n is the class number), it is expected to be unchanged across all training data*. Therefore, the empirical loss is expected to remain unchanged with the stochastic optimum shifting algorithm. To summarize, our contributions include:

- We propose optimum shifting (OS), which enables us to change the parameters of neural networks from a sharp minimum to a flatter one while maintaining the same training loss. We prove that the generalization can be improved from the Hessian trace perspective.
- We conduct extensive experiments with various deep neural network architectures on benchmark datasets. Experimental results show that by using stochastic OS both during training and after training, the deep models can achieve better generalization performance.
- Our proposed OS is versatile and can be easily integrated into traditional regularization techniques, such as weight decay and recently proposed methods to find flatter minima (e.g., the sharpness-aware minimization method (Foret et al. 2021)).

## Related Work

**Sharpness and Generalization.** Research on the correlation between the generalization performance and the sharpness of the loss landscape can be traced back to Hochreiter and Schmidhuber (1994). Jiang et al. (2019) and Yao et al. (2020) performed a large-scale empirical study on various generalization measures and showed that the sharpness of the loss landscape is the most suitable measure correlated with the generalization performance. Keskar et al. (2017) observed that when the batch size of SGD is increased to train a model, the test error and the sharpness increase. Gatmiry et al. (2023) showed that with the standard restricted isometry property in measurement, minimizing the Hessian trace can lead to better generalization. It is also noteworthy that Dinh et al. (2017) argued that for networks with scaling invariance, there always exist models with good generalization but with large sharpness, but this does not contradict the sharpness minimization algorithms (Gatmiry et al. 2023).

**Sharpness Minimization Algorithm.** Although the above rescaling trick can maximize sharpness and maintain both the test and training loss, it does not contradict the existing sharpness minimization method, which only asserts the solution with a minimal trace of Hessian generalizes well, but not vice versa. Therefore, recent studies have proposed several penalty-based sharpness regularization methods to improve the generalization. SAM (Foret et al. 2021) was proposed to penalize the sharpness of the loss landscape to improve the generalization. The full-batch SAM aims to minimize worst-direction sharpness (Hessian spectrum) and 1-SAM aims to minimize the average-direction sharpness (Hessian trace) (Wen, Ma, and Li 2023). Furthermore, Kwon et al. (2021) proposed ASAM, where optimization could keep invariant to a specific weight-rescaling operation. In addition, Zhao, Zhang, and Hu (2022) proposes improving generalization by penalizing the gradient norm during optimization. It is worth noting that the methods above are all penalty-based methods, i.e. adding a penalty term to the loss function to encourage the flatness. Compared with them, our method is a constraint and objective separation method, which separates the flatness and the loss value as two isolated objectives to optimize. As a result, our OS can be easily integrated into penalty-based methods (e.g. weight decay) to achieve better performance.

**Neural Collapse.** Recent seminal works empirically demonstrated that last-layer features and classifiers of a trained DNN exhibit an intriguing Neural Collapse ($\mathcal{NC}$) phenomenon (Zhu et al. 2021; Tirer and Bruna 2022; Zhou et al. 2022). Specifically, it has been shown that the learned features (the output of the penultimate layer) of within-class samples converge to their class means. Moreover, $\mathcal{NC}$ seems to take place regardless of the choice of loss functions. We utilize this phenomenon and propose stochastic optimum shifting, which can reduce the computational costs and provide more degrees of freedom for optimum shifting. For example, when we apply optimum shifting to ResNet/DenseNet for CIFAR-100 classification, ensuring that the loss remains unchanged across 100 samples, the loss remains probabilistically unchanged across the entire set of 50,000 training data points.

## Optimum Shifting

**Notations.** we denote $\mathcal{S} \triangleq \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^n$ as the training set containing $n$ training samples, $L$ as the loss function, and $f_k(\boldsymbol{x}_i)$ as the $k$ layer neural network approximation. The vectorized output of $i$-th layer is represented using $\text{vec}(\boldsymbol{x}_{l,i}) \in \mathbb{R}^{m_l}$. We vectorize parameters in each layer and stack it as $\mathbf{W} = [\text{vec}(\mathbf{v}), \text{vec}(\mathbf{F}_l), \cdots, \text{vec}(\mathbf{F}_1))]$

### Motivation

Before proceeding, we first define the flatness of neural networks. There are many measures to define the flatness. But currently, the trace of Hessian has been theoretically proved with the generalization bound (Gatmiry et al. 2023). In this paper, we define flatness by the Hessian trace.

**Definition 1** *The flatness* $\text{Flat}(L)$ *is defined as the trace of the Hessian matrix* $\mathbf{H}_L$ *of the loss function L with respect to the network parameters* $\mathbf{W}$*:*

$$\text{Flat}(L) \triangleq \text{tr}(\mathbf{H}_L) = \text{tr}(\nabla^2_{\mathbf{W}} L(\boldsymbol{f}(\boldsymbol{x}_i), \boldsymbol{y}_i)). \tag{2}$$

Thus optimum shifting (OS) can be represented as:

$$\text{minimize} \quad \text{tr}(\nabla^2_{\boldsymbol{V}} L(\boldsymbol{f}(\boldsymbol{x}_i), \boldsymbol{y}_i)), \tag{3}$$

$$\text{subject to} \quad \boldsymbol{A}\boldsymbol{V} = \boldsymbol{Z}. \tag{4}$$

However, since Eq. 3 dose not closed-form solution, we find its upper and lower bound to optimize. Next, we show the main theorem of our paper. It shows that for different neural networks such as CNN, ResNet, DenseNet and MLP, the lower bound and the upper bound of the Hessian trace are linear with the Frobenius norm of the weight in the final linear layer. Therefore, when the output of training data is fixed, if we minimize the Frobenius norm of the weight in the final linear layer, both the upper bound and the lower bound of the Hessian trace will also be minimized, thereby suggesting a better generalization ability , which is shown in the following Theorem.

**Algorithm 1:** SOS algorithm during training

**Input:** Training set $\mathcal{S} \triangleq \cup_{i=1}^{n}\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}$, batch size $b_1, b_2$ for SGD and SOS, step size $\gamma > 0$.

**for** *number of training epochs* **do**
  Sample batch $\mathcal{B} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), ...(\boldsymbol{x}_{b_2}, \boldsymbol{y}_{b_2})\}$;
  Compute the input and output matrix ;
  $\boldsymbol{A} = \left[\boldsymbol{x}_{k,1}, \boldsymbol{x}_{k,2}, \cdots, \boldsymbol{x}_{k,b_2}\right]$ $\boldsymbol{Z} = \left[\boldsymbol{V}^T\boldsymbol{x}_{k,1}, \boldsymbol{V}^T\boldsymbol{x}_{k,2}, \cdots, \boldsymbol{V}^T\boldsymbol{x}_{k,b_2}\right]$ ;
  **for** *each columns* $\boldsymbol{V}_i$ *in the final linear layer* **do**
    Conduct Gaussian elimination to make $\boldsymbol{A}$ row-independent:
    $[\boldsymbol{A}_*, \boldsymbol{Z}_{i*}] = \textit{Gaussian eliminate}([\boldsymbol{A}, \boldsymbol{Z}_i])$ ;
    Update the parameters: $\boldsymbol{V}_i^* = \boldsymbol{A}_*(\boldsymbol{A}_*(\boldsymbol{A}_*)^T)^{-1}\boldsymbol{Z}_{i*}$;
  **for** $t = 0, 1, \cdots, s$ **do**
    Update all parameters using SGD;
    $\mathbf{W}_t = \mathbf{W}_{t-1} - \gamma\frac{1}{b_1}\sum_{i=1}^{b_1}\nabla_{\mathbf{W}_{t-1}}L$;

**Theorem 1** *For an $l$-layer MLP or convolutional (CNN, ResNet and Densenet) neural network, given the loss function $L$, the trace of Hessian can be bounded by:*

$$C_0 + C_1\|\boldsymbol{V}\|^2 \leq \mathrm{tr}(\mathbf{H}_L) \leq C_0 + C_2\|\boldsymbol{V}\|^2, \tag{5}$$

*where $\boldsymbol{V}$ is the weight matrix of the last linear layer and $C_0, C_1, C_2$ are constants and independent of the last hidden layer's weight $\boldsymbol{V}$. Therefore, if $\|\boldsymbol{V}\|^2$ is minimized, both the upper bound and the lower bound of the Hessian trace will also be minimized.*

## Our Proposed Optimum Shifting Method

Motivated by Theorem 1 above, we aim to replace the current weight using the least $F$-norm solution in the linear space. A linear layer $\mathbb{R}^m \to \mathbb{R}^n$ with the activation function $\sigma$ can be represented as follows:

$$\phi^{\mathrm{fc}}(\boldsymbol{V}) = \sigma(\boldsymbol{A}\boldsymbol{V} + \boldsymbol{C}), \tag{6}$$

where $\boldsymbol{A} \in \mathbb{R}^{batch\times m}$, $\boldsymbol{V} \in \mathbb{R}^{m\times n}$, $\boldsymbol{C} \in \mathbb{R}^{batch\times n}$ and $batch$ is the batch size. We denote the result of $\boldsymbol{A}\boldsymbol{V}$ as

$$\boldsymbol{A}\boldsymbol{V} \coloneqq \boldsymbol{Z}. \tag{7}$$

The training loss value of a neural network will remain unchanged, regardless of how the parameter $\boldsymbol{V}$ is adjusted in the solution space, as long as the input matrix $\boldsymbol{A}$ and output matrix $\boldsymbol{Z}$ are fixed. Specifically, the equation $\boldsymbol{A}\boldsymbol{V} = \boldsymbol{Z}$ defines a system of linear equations. If this system is underdetermined, then $\boldsymbol{V}$ has an infinite number of solutions. Any option in the solution space is available as a replacement for the current $\boldsymbol{V}$. As stated in Section , both the upper bound and the lower bound of the Hessian trace are linear with $\|\boldsymbol{V}\|^2$. If $\|\boldsymbol{V}\|^2$ is minimized, both the upper bound and the lower bound of the Hessian trace will also be minimized. As a result, we prefer replacing the current solution with the one that has the least Frobenius norm in the solution space. This can be obtained by solving a least-square problem as follows:

$$\text{minimize} \quad \|\boldsymbol{V}\|^2, \tag{8}$$
$$\text{subject to} \quad \boldsymbol{A}\boldsymbol{V} = \boldsymbol{Z}. \tag{9}$$

Thus, we aim to find the point with the smallest Frobenius norm in the solution space to replace the current $\boldsymbol{V}$. To solve the least-square problem, we make two assumptions for the input matrix $\boldsymbol{A}$.

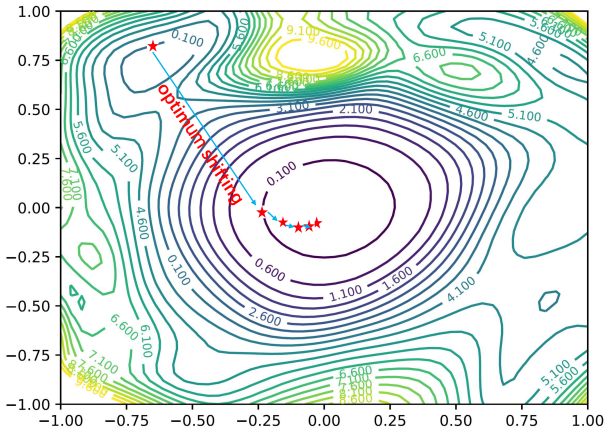



Figure 1: Schematic of the OS algorithm.[1]

**Assumption 1 (Low Rank Assumption)** *The rank of the input matrix $\boldsymbol{A}$ is smaller than its number of columns. i.e., $rank(\boldsymbol{A}) < m$.*

**Assumption 2 (Row Independence Assumption)** *The rows of the input matrix $\boldsymbol{A}$ are linearly independent. i.e., $\forall \boldsymbol{a} \in \mathbb{R}^{batch}$, $\boldsymbol{a}^\top\boldsymbol{A} = \mathbf{0}^\top$ if and only if $\boldsymbol{a} = \mathbf{0}$.*

The first assumption ensures that the linear system is underdetermined, and has many solutions in the solution space. The second assumption makes $\boldsymbol{A}\boldsymbol{A}^\top$ in the closed form solution in Eq. 13 is invertible. In the next section, we will show how to satisfy the two assumptions.

Because $\boldsymbol{V} \in \mathbb{R}^{m\times n}$ is a matrix, we need to decompose it into $n$ independent least-square problems. We denote the $i$-th column of $\boldsymbol{V}$ as $\boldsymbol{V}_i$, and the Lagrangian for the least-square problem can be expressed as follows:

$$L_1(\boldsymbol{V}_1, \cdots, \boldsymbol{V}_n, \boldsymbol{\lambda}_1, \cdots, \boldsymbol{\lambda}_n) = \sum_{i=1}^{n}(\boldsymbol{V}_i^\top\boldsymbol{V}_i + \boldsymbol{\lambda}_i^\top(\boldsymbol{A}\boldsymbol{V}_i - \boldsymbol{Z}_i)). \tag{10}$$

Since $L_1$ is a convex quadratic function of each variable $(\boldsymbol{V}_i, \boldsymbol{\lambda}_i)$, we can find the minimum $(\boldsymbol{V}_i^*, \boldsymbol{\lambda}_i^*)$ from the optimality condition:

$$\nabla_{\boldsymbol{V}_i}L_1 = 2\boldsymbol{V}_i + \boldsymbol{A}^\top\boldsymbol{\lambda}_i = \mathbf{0}, \tag{11}$$
$$\nabla_{\boldsymbol{\lambda}_i}L_1 = \boldsymbol{A}\boldsymbol{V}_i - \boldsymbol{Z}_i = \mathbf{0}, \tag{12}$$

which yeilds a closed-form solution $\boldsymbol{V}_i^* = \boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{A}^\top)^{-1}\boldsymbol{Z}_i$. By resolving $n$ independent least-square problems, we can finally get the point with the smallest Frobenius norm as

$$\boldsymbol{V}^* = [\boldsymbol{V}_1^*, \boldsymbol{V}_2^*, \cdots, \boldsymbol{V}_n^*] = \boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{A}^\top)^{-1}\boldsymbol{Z}. \tag{13}$$

In what follows, we will detail how to satisfy Assumption 1 and Assumption 2.

[1] Figure 1 is generated following Li et al. (2018), which shows the loss landscape for all parameters using 2D plots.

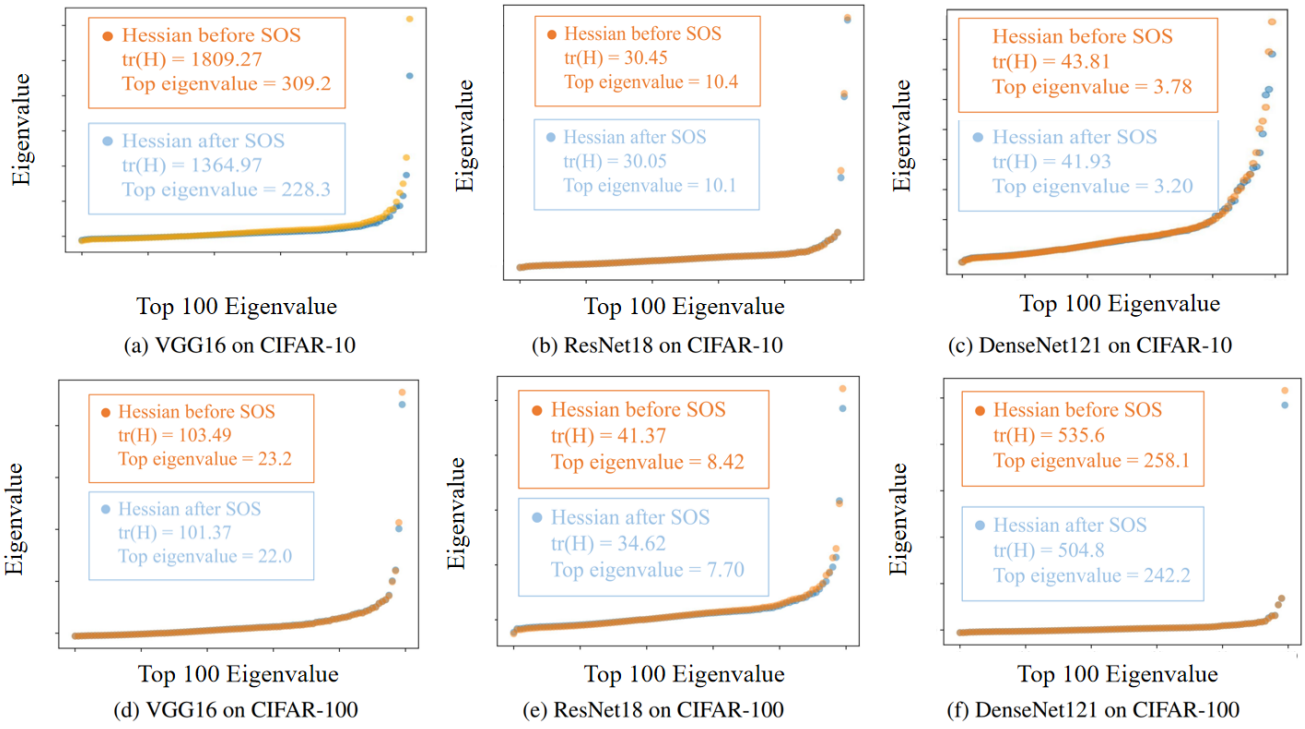



Figure 2: Top 100 eigenvalues of the Hessian matrix before and after SOS.

**Stochastic Optimum Shifting.** As mentioned in Assumption 1, OS relies on the low-rank property of the input matrix $\boldsymbol{A}$. This property ensures that the linear system is underdetermined, allowing for an infinite number of solutions within the solution space for OS to change. It is important to highlight that when using a small batch for OS, i.e., $batch < m$, we have

$$\text{rank}(A) \leq batch < m, \tag{14}$$

which satisfies the requirement specified in Assumption 1.

To achieve optimum shifting, the training loss is expected to remain unchanged across the entire dataset. However, for stochastic optimum shifting, we only maintain it unchanged in a small batch. Neural Collapse (Zhu et al. 2021; Tirer and Bruna 2022; Zhou et al. 2022) helps us to understand this phenomenon. It reveals that: as training progresses, the within-class variation of the activations becomes negligible as they collapse to their class means. For example, when training on the CIFAR-100 dataset, the images will converge to the 100 class means. Therefore, if the loss of 100 images remains unchanged after performing optimum shifting to the final fully connective layer, the loss of the entire dataset will also remain nearly unchanged, which can both satisfy Assumption 1 and reduce the computationl costs.

Moreover, the small batch for optimum shifting also offers more degrees of freedom for optimum shifting. For instance, when the last layer maps a vector with 1024 dimensions to 100 dimensions, i.e., the weight matrix $\boldsymbol{V} \in \mathbb{R}^{1024\times100}$. When feeding the entire dataset to the neural network, the input matrix $\boldsymbol{A} \in \mathbb{R}^{50000\times1024}$, which may not be under-determined. When using stochastic optimum shifting with $batch = 300$, the input matrix $\boldsymbol{A} \in \mathbb{R}^{300\times1024}$, which means that the systems of linear equations are underdetermined. Hence, it must have infinitely many solutions in the solution space.

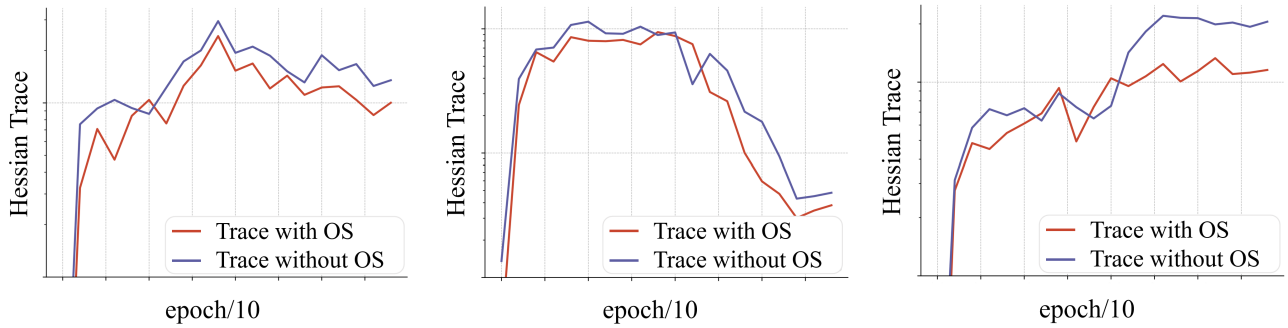
**Gaussian elimination.** To ensure the invertibility of $\boldsymbol{A}\boldsymbol{A}^\top$ in the closed-form solution of Eq. 13, it is imperative to fulfill the condition stated in Assumption 2. However, in practice, $\boldsymbol{A}$ may be row-dependent and $\boldsymbol{A}\boldsymbol{A}^\top$ may be singular. After Gaussian elimination, each non-zero row of input matrix $\boldsymbol{A}$ will be independent and those zero rows will be discarded. Consequently, the resulting matrix $\boldsymbol{A}_*$ in Algorithm 1 becomes row-independent, ensuring that the matrix $\boldsymbol{A}_*\boldsymbol{A}_*^\top$ is invertible.

## Experiments

To validate the effectiveness of our proposed SOS, we investigate the performance of different DNNs (VGG, ResNet, DenseNet, Shufflenet, ResNext, Yolo) on different computer vision tasks, including CIFAR-10/100, ImageNet classification and PASCAL VOC object detection. In the first subsection, SOS is applied to trained deep models. In the second subsection, it is further applied repeatedly in the training process. When SOS is applied in the training process, we compare our method with two other training schemes: one is the standard SGD training scheme with weight decay and the other is the SAM (Foret et al. 2021) training scheme. As we can see below, SOS improves the generalization ability of trained models, standard SGD scheme with weight decay,

| | CIFAR-100 | | CIFAR-10 | |
|---|---|---|---|---|
| VGG-16 | Basic | Mixup Augmentation | Basic | Mixup Augmentation |
| Standard | $72.12_{\pm 0.52}$ | $73.22_{\pm 0.41}$ | $92.01_{\pm 0.45}$ | $93.21_{\pm 0.32}$ |
| **+ SOS (ours)** | $\mathbf{72.23}_{\pm 0.54}$ | $\mathbf{73.38}_{\pm 0.81}$ | $\mathbf{92.28}_{\pm 0.22}$ | $\mathbf{93.35}_{\pm 0.43}$ |
| ResNet-18 | Basic | Mixup Augmentation | Basic | Mixup Augmentation |
| Standard | $78.03_{\pm 0.41}$ | $79.23_{\pm 0.19}$ | $95.31_{\pm 0.26}$ | $95.97_{\pm 0.21}$ |
| **+ SOS (ours)** | $\mathbf{78.25}_{\pm 0.37}$ | $\mathbf{79.88}_{\pm 0.76}$ | $\mathbf{95.47}_{\pm 0.26}$ | $\mathbf{96.04}_{\pm 0.25}$ |
| ResNet-50 | Basic | Mixup Augmentation | Basic | Mixup Augmentation |
| Standard | $79.30_{\pm 0.22}$ | $79.53_{\pm 0.21}$ | $95.30_{\pm 0.16}$ | $96.10_{\pm 0.08}$ |
| **+ SOS (ours)** | $\mathbf{79.57}_{\pm 0.24}$ | $\mathbf{79.72}_{\pm 0.31}$ | $\mathbf{95.47}_{\pm 0.28}$ | $\mathbf{96.22}_{\pm 0.21}$ |
| DenseNet-201 | Basic | Mixup Augmentation | Basic | Mixup Augmentation |
| Standard | $80.02_{\pm 0.32}$ | $81.36_{\pm 0.33}$ | $95.31_{\pm 0.22}$ | $96.06_{\pm 0.06}$ |
| **+ SOS (ours)** | $\mathbf{80.35}_{\pm 0.26}$ | $\mathbf{81.55}_{\pm 0.28}$ | $\mathbf{95.64}_{\pm 0.36}$ | $\mathbf{96.31}_{\pm 0.34}$ |

Table 1: Test Accuracy on CIFAR Classification for trained models




(a) VGG (b) ResNet18 (c) DenseNet201

Figure 3: Visualization of the Hessian trace for different models and dataset with and without SOS during training.

and SAM training scheme.

## Applying SOS to Trained Models

**Test Accuracy.** We first evaluate SOS by applying it to trained deep models on CIFAR-10 and CIFAR-100 dataset (Krizhevsky, Hinton et al. 2009), which consists of 50k training images and 10k testing images in 10 and 100 classes. Different convolutional neural network architectures are tested, including relatively simple architectures, such as VGG (Simonyan and Zisserman 2014), and complex architectures, such as ResNet (He et al. 2016) and DenseNet (Huang et al. 2017). For the training datasets, we employ data augmentations. One is the basic augmentation (basic normalization and random horizontal flip) and the other is Mixup augmentation (Zhang et al. 2017). Table 1 shows the results. We can see all the test accuracy has been improved slightly by SOS. For example, the test accuracy of VGG-16 on CIFAR-100 has been improved from $72.23_{\pm 0.54}\%$ to $73.38_{\pm 0.81}\%$.

**Hessian Analysis.** We visualize the top 100 eigenvalues of the Hessian matrix before and after SOS in ascending order as shown in Figure 2. The orange and blue spots represent the eigenvalue before and after SOS. We also calculate the trace and top-1 eigenvalue for analysis. We can see that both the trace (average-direction sharpness) and the top 1 eigenvalue (worst-direction sharpness) have been minimized by SOS, which is consistent with our proposed Theorem 1.

| Network | SOS | Top-1 acc | Top-5 acc |
|---|---|---|---|
| ResNet50 | Before SOS | 76.14 | 92.86 |
| | After SOS | $\mathbf{76.59}_{\pm 0.04}$ | $\mathbf{93.14}_{\pm 0.01}$ |
| Shufflenet_V2 | Before SOS | 76.23 | 93.00 |
| | After SOS | $\mathbf{76.30}_{\pm 0.02}$ | $\mathbf{93.04}_{\pm 0.01}$ |
| ResNext50 | Before SOS | 77.61 | 93.68 |
| | After SOS | $\mathbf{77.69}_{\pm 0.02}$ | $\mathbf{93.72}_{\pm 0.01}$ |

Table 2: **Applying SOS to pre-trained models on ImageNet dataset.** We also use three PyTorch official pre-trained models: ResNet50, Shufflenet_V2, and ResNext50 to test the effectiveness of our method. We can see that both the top-1 and top-5 accuracy have been increased.

**ImageNet Classification.** We evaluate our method on the

| | CIFAR-100 | | CIFAR-10 | |
|---|---|---|---|---|
| VGG-16 | Basic | Mixup Augmentation | Basic | Mixup Augmentation |
| SGD | $72.12_{\pm 0.52}$ | $73.22_{\pm 0.41}$ | $92.01_{\pm 0.45}$ | $93.21_{\pm 0.32}$ |
| **+ SOS (ours)** | $\mathbf{73.35}_{\pm 0.43}$ | $\mathbf{74.36}_{\pm 0.72}$ | $\mathbf{92.37}_{\pm 0.34}$ | $\mathbf{93.62}_{\pm 0.26}$ |
| SAM | $74.63_{\pm 0.12}$ | $74.96_{\pm 0.56}$ | $94.47_{\pm 0.33}$ | $94.82_{\pm 0.52}$ |
| **+ SOS (ours)** | $\mathbf{74.92}_{\pm 0.03}$ | $\mathbf{75.38}_{\pm 0.29}$ | $\mathbf{94.76}_{\pm 0.47}$ | $\mathbf{95.10}_{\pm 0.31}$ |
| ResNet-18 | Basic | Mixup Augmentation | Basic | Mixup Augmentation |
| SGD | $78.03_{\pm 0.41}$ | $79.23_{\pm 0.19}$ | $95.31_{\pm 0.26}$ | $95.97_{\pm 0.21}$ |
| **+ SOS (ours)** | $\mathbf{78.43}_{\pm 0.24}$ | $\mathbf{79.97}_{\pm 0.46}$ | $\mathbf{95.54}_{\pm 0.21}$ | $\mathbf{96.17}_{\pm 0.04}$ |
| SAM | $78.63_{\pm 0.33}$ | $80.18_{\pm 0.12}$ | $95.94_{\pm 0.24}$ | $96.11_{\pm 0.32}$ |
| **+ SOS (ours)** | $\mathbf{79.66}_{\pm 0.18}$ | $\mathbf{80.43}_{\pm 0.23}$ | $\mathbf{96.18}_{\pm 0.11}$ | $\mathbf{96.48}_{\pm 0.51}$ |
| ResNet-50 | Basic | Mixup Augmentation | Basic | Mixup Augmentation |
| SGD | $79.30_{\pm 0.22}$ | $79.53_{\pm 0.21}$ | $95.30_{\pm 0.16}$ | $96.10_{\pm 0.08}$ |
| **+ SOS (ours)** | $\mathbf{79.63}_{\pm 0.27}$ | $\mathbf{79.90}_{\pm 0.43}$ | $\mathbf{95.53}_{\pm 0.23}$ | $\mathbf{96.33}_{\pm 0.12}$ |
| SAM | $78.62_{\pm 0.36}$ | $81.65_{\pm 0.27}$ | $95.82_{\pm 0.18}$ | $96.51_{\pm 0.27}$ |
| **+ SOS (ours)** | $\mathbf{78.97}_{\pm 0.19}$ | $\mathbf{82.41}_{\pm 0.33}$ | $\mathbf{96.31}_{\pm 0.21}$ | $\mathbf{96.88}_{\pm 0.30}$ |
| DenseNet-201 | Basic | Mixup Augmentation | Basic | Mixup Augmentation |
| SGD | $80.02_{\pm 0.32}$ | $81.36_{\pm 0.33}$ | $95.31_{\pm 0.22}$ | $96.06_{\pm 0.06}$ |
| **+ SOS (ours)** | $\mathbf{80.43}_{\pm 0.13}$ | $\mathbf{81.60}_{\pm 0.14}$ | $\mathbf{95.83}_{\pm 0.12}$ | $\mathbf{96.40}_{\pm 0.21}$ |
| SAM | $80.14_{\pm 0.56}$ | $82.78_{\pm 0.42}$ | $96.08_{\pm 0.25}$ | $96.82_{\pm 0.14}$ |
| **+ SOS (ours)** | $\mathbf{81.06}_{\pm 0.23}$ | $\mathbf{83.13}_{\pm 0.13}$ | $\mathbf{96.47}_{\pm 0.32}$ | $\mathbf{97.05}_{\pm 0.42}$ |

Table 3: Testing accuracy of different CNNs on CIFAR-10 and CIFAR-100 when implementing the four training schemes.

ImageNet classification dataset using three PyTorch official pre-trained deep models: ResNet50, Shufflenet_V2 (Huang et al. 2017), and ResNext50 and apply SOS to test its effectiveness. Table 2 reports the test accuracy on Imagenet classification with different models. We can see that SOS also improves both top-1 and top-5 accuracy for different model architectures. For example, for ResNet-50, the top-1 accuracy has been increased from 76.13% to 76.61% and the top-5 accuracy has been increased from 92.86% to 93.14%, which further verified our method can also improve the generalization ability in the large scale dataset.

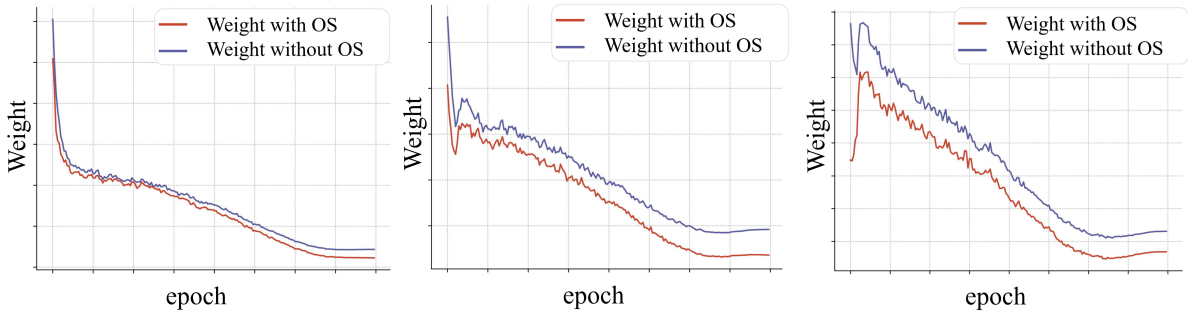
## Applying SOS in Training Process

**Generalization Ability Test.** Next, we demonstrate that SOS can also improve the generalization ability of neural networks when applied in the training process. We apply SGD to train the same four CNN models under the CIFAR-10 and CIFAR-100 datasets with the same data augmentation strategies. Following (Huang et al. 2017), the weight decay is $10^{-4}$ and a Nesterov momentum of 0.9 without damping. The batch size is set to 64 and the models are trained for 300 epochs. The initial learning rate is set to 0.1 and is reduced by a factor of 10 at 50% and 75% of the total training epochs. All images are applied with a simple random horizontal flip and normalized using their mean and standard deviation. We focus on the comparisons between four different training schemes, namely the standard SGD scheme, SOS on SGD schemes, SAM scheme (Foret et al. 2021), and SOS on SAM schemes. Given that both SGD and SAM search optimal parameters within local regions, our SOS operates globally and thus can be integrated with SGD and SAM to attain better generalization. As we can see in Table 3, all the accuracy of the four CNN architectures on CIFAR-10 and CIFAR-100 have been improved with SOS compared to the standard SGD and SAM without SOS. The test accuracy is also increased, for example, the test accuracy of ResNet-50 on CIFAR-100 has been improved from $81.65_{\pm 0.27}\%$ to $82.41_{\pm 0.33}\%$.

**Hessian Analysis.** The evolution of Hessian trace during training with and without SOS is shown in Figure 3. We train VGG16 and ResNet18 on the CIFAR-10 dataset, and DenseNet201 on the CIFAR-100 dataset, for 200 epochs, and visualize the trace of the Hessian during training. It shows that the trace of Hessian during training keeps in-

| SOS batch size | w/o SOS | batch = 200 | batch = 300 | batch = 400 |
|---|---|---|---|---|
| VGG-16 | $72.12_{\pm 0.52}$ | $72.91_{\pm 0.19}$ | $\mathbf{73.35}_{\pm 0.43}$ | $73.18_{\pm 0.43}$ |
| ResNet-18 | $78.03_{\pm 0.41}$ | $78.43_{\pm 0.24}$ | $\mathbf{79.23}_{\pm 0.19}$ | $79.01_{\pm 0.12}$ |
| ResNet-50 | $79.30_{\pm 0.22}$ | $79.53_{\pm 0.21}$ | $\mathbf{79.63}_{\pm 0.27}$ | $79.30_{\pm 0.22}$ |
| DenseNet-201 | $80.02_{\pm 0.32}$ | $80.27_{\pm 0.23}$ | $\mathbf{80.43}_{\pm 0.13}$ | $80.33_{\pm 0.18}$ |

Table 4: Investigation of SOS batch size on CIFAR100




(a) VGG16 (b) ResNet18 (c) DenseNet201

Figure 4: Weight visualization of different models on CIFAR10 datase.

| VOC | Yolo-V5s | Yolo-V5x |
|---|---|---|
| mAP w/ SOS | $83.4_{\pm 0.2}$ | $87.1_{\pm 0.2}$ |
| mAP w/o SOS | $\mathbf{83.7}_{\pm 0.1}$ | $\mathbf{87.4}_{\pm 0.2}$ |

Table 5: Experiment Result on Object Detection.

creasing. However, when we apply SOS to the model during training, the trace of Hessian is minimized, which indicates a better generalization ability.

**Analysis on the Last-Layer Weight.** The model parameters' weight with and without SOS is shown in Figure 4. The Frobenius norm of the last linear layer's weight increases a lot before dividing the learning rate by 10. It has been slowed down when the learning rate is divided and the weight starts to decrease rapidly for the model with OS. When SOS is applied, the increase rate is slowed down for the DenseNet. For VGG, the weight does not increase but decreases from the first epoch.

**Discussion on SOS Batch Size.** The SOS algorithm we proposed to satisfy Assumption 1 introduces a hyper-parameter batch size, which has a significant impact on the performance of the algorithm. For SOS batch size, there is a trade-off. When the batch size is too large, it will limit the degrees of freedom for SOS to solve $AV = Z$. However, when the batch size is too small, it may not keep the training loss value unchanged. We conduct experiments to validate our statement on CIFAR100 dataset and show in Table 4. As Table 4 shows, with the increase in batch size, the improvement increases initially and then decreases. We can see that with $batchsize = 300$, it attains the best performance.

**Object Detection.** Our method improves generalization performance on other recognition tasks. We conduct experiments on other computer vision tasks, such as object detection to validate the improved performance. Table 5 shows the object detection baseline results with and without SOS on PASCAL VOC dataset (Everingham et al. 2010). We adopt YOLOv5s and YOLOv5x (Jocher et al. 2020) as the detection model and we see that the performance (mAP) of the two models is improved when applied with SOS.

## Conclusion

In this paper, we introduce a novel technique called optimum shifting, which moves the parameters of neural networks from sharper minima to flatter minima while keeping the training loss unchanged. This technique treats the matrix multiplications in the network as systems of underdetermined linear equations and adjusts the parameters in the solution space. To reduce computational costs and increase the degrees of freedom for optimum shifting, we propose stochastic optimum shifting, which selects a small batch for this process. The neural collapse phenomenon guarantees that the proposed stochastic optimum shifting keeps the empirical loss unchanged. We conduct experiments to demonstrate that stochastic optimum shifting reduces the Hessian trace and improves generalization across various vision tasks.

## Acknowledgments

This work was supported by the National Natural Science Foundation of China (No.62106103, No.62222605), and the

basic research project (ILF240021A24). This work is also partially supported by High Performance Computing Platform of Nanjing University of Aeronautics and Astronautics.

## References

Blanc, G.; Gupta, N.; Valiant, G.; and Valiant, P. 2020. Implicit regularization for deep neural networks driven by an ornstein-uhlenbeck like process. In *Conference on learning theory*, 483–513. PMLR.

Chaudhari, P.; Choromanska, A.; Soatto, S.; LeCun, Y.; Baldassi, C.; Borgs, C.; Chayes, J.; Sagun, L.; and Zecchina, R. 2019. Entropy-SGD: Biasing gradient descent into wide valleys. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12): 124018.

Dinh, L.; Pascanu, R.; Bengio, S.; and Bengio, Y. 2017. Sharp minima can generalize for deep nets. In *ICML*, 1019–1028. PMLR.

Everingham, M.; Gool, L. V.; Williams, C. K. I.; Winn, J. M.; and Zisserman, A. 2010. The Pascal Visual Object Classes (VOC) Challenge. *Int. J. Comput. Vis.*, 88(2): 303–338.

Foret, P.; Kleiner, A.; Mobahi, H.; and Neyshabur, B. 2021. Sharpness-aware minimization for efficiently improving generalization. *ICLR*.

Gatmiry, K.; Li, Z.; Chuang, C.-Y.; Reddi, S.; Ma, T.; and Jegelka, S. 2023. The Inductive Bias of Flatness Regularization for Deep Matrix Factorization. *arXiv preprint arXiv:2306.13239*.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2020. Generative adversarial networks. *Communications of the ACM*, 63(11): 139–144.

Guan, R.; Li, Z.; Tu, W.; Wang, J.; Liu, Y.; Li, X.; Tang, C.; and Feng, R. 2024a. Contrastive Multiview Subspace Clustering of Hyperspectral Images Based on Graph Convolutional Networks. *IEEE Transactions on Geoscience and Remote Sensing*, 62: 1–14.

Guan, R.; Tu, W.; Li, Z.; Yu, H.; Hu, D.; Chen, Y.; Tang, C.; Yuan, Q.; and Liu, X. 2024b. Spatial-Spectral Graph Contrastive Clustering with Hard Sample Mining for Hyperspectral Images. *IEEE Transactions on Geoscience and Remote Sensing*, 1–16.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *CVPR*.

Hochreiter, S.; and Schmidhuber, J. 1994. Simplifying neural nets by discovering flat minima. *NeurIPS*, 7.

Hu, D.; Guan, R.; Liang, K.; Yu, H.; Quan, H.; Zhao, Y.; Liu, X.; and He, K. 2024a. scEGG: an exogenous gene-guided clustering method for single-cell transcriptomic data. *Briefings in Bioinformatics*, 25(6): bbae483.

Hu, D.; Liu, S.; Wang, J.; Zhang, J.; Wang, S.; Hu, X.; Zhu, X.; Tang, C.; and Liu, X. 2024b. Reliable Attribute-missing Multi-view Clustering with Instance-level and feature-level Cooperative Imputation. In *Proceedings of the 32nd ACM International Conference on Multimedia*, 1456–1466.

Huang, G.; Liu, Z.; Van Der Maaten, L.; and Weinberger, K. Q. 2017. Densely connected convolutional networks. In *CVPR*.

Jiang, Y.; Neyshabur, B.; Mobahi, H.; Krishnan, D.; and Bengio, S. 2019. Fantastic generalization measures and where to find them. *arXiv preprint arXiv:1912.02178*.

Jocher, G.; Stoken, A.; Borovec, J.; NanoCode012; ChristopherSTAN; Changyu, L.; Laughing; tkianai; Hogan, A.; lorenzomammana; yxNONG; AlexWang1900; Diaconu, L.; Marc; wanghaoyang0106; ml5ah; Doug; Ingham, F.; Frederik; Guilhen; Hatovix; Poznanski, J.; Fang, J.; Yu, L.; changyu98; Wang, M.; Gupta, N.; Akhtar, O.; PetrDvoracek; and Rai, P. 2020. ultralytics/yolov5: v3.1 - Bug Fixes and Performance Improvements.

Keskar, N. S.; Mudigere, D.; Nocedal, J.; Smelyanskiy, M.; and Tang, P. T. P. 2016. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*.

Keskar, N. S.; Mudigere, D.; Nocedal, J.; Smelyanskiy, M.; and Tang, P. T. P. 2017. On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. In *ICLR*.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images. Technical report, MIT and NYU.

Kwon, J.; Kim, J.; Park, H.; and Choi, I. K. 2021. Asam: Adaptive sharpness-aware minimization for scale-invariant learning of deep neural networks. In *ICML*. PMLR.

Li, H.; Xu, Z.; Taylor, G.; Studer, C.; and Goldstein, T. 2018. Visualizing the loss landscape of neural nets. *NeurIPS*.

Lin, X.; Ren, C.; Liu, X.; Huang, J.; and Lei, Y. 2023. Unsupervised Image Denoising in Real-World Scenarios via Self-Collaboration Parallel Generative Adversarial Branches. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 12642–12652.

Lin, X.; Yue, J.; Ding, S.; Ren, C.; Qi, L.; and Yang, M.-H. 2024a. Dual Degradation Representation for Joint Deraining and Low-Light Enhancement in the Dark. *IEEE Transactions on Circuits and Systems for Video Technology*.

Lin, X.; Zhou, Y.; Yue, J.; Ren, C.; Chan, K. C.; Qi, L.; and Yang, M.-H. 2024b. Re-boosting Self-Collaboration Parallel Prompt GAN for Unsupervised Image Restoration. *arXiv preprint arXiv:2408.09241*.

Neyshabur, B. 2017. Implicit regularization in deep learning. *arXiv preprint arXiv:1709.01953*.

OpenAI. 2023. GPT-4 Technical Report. arXiv:2303.08774.

Shen, F.; and Tang, J. 2024. IMAGPose: A Unified Conditional Framework for Pose-Guided Person Generation. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Shen, F.; Xie, Y.; Zhu, J.; Zhu, X.; and Zeng, H. 2023. Git: Graph interactive transformer for vehicle re-identification. *IEEE Transactions on Image Processing*, 32: 1039–1051.

Shen, F.; Ye, H.; Liu, S.; Zhang, J.; Wang, C.; Han, X.; and Yang, W. 2024a. Boosting consistency in story visualization with rich-contextual conditional diffusion models. *arXiv preprint arXiv:2407.02482*.

Shen, F.; Ye, H.; Zhang, J.; Wang, C.; Han, X.; and Wei, Y. 2024b. Advancing Pose-Guided Image Synthesis with Progressive Conditional Diffusion Models. In *The Twelfth International Conference on Learning Representations*.

Simonyan, K.; and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Sohl-Dickstein, J.; Weiss, E.; Maheswaranathan, N.; and Ganguli, S. 2015. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*, 2256–2265. PMLR.

Sun, H. 2024. Ultra-High Resolution Segmentation via Boundary-Enhanced Patch-Merging Transformer. arXiv:2412.10181.

Sun, H.; Xu, L.; Jin, S.; Luo, P.; Qian, C.; and Liu, W. 2024. PROGRAM: PROtotype GRAph Model based Pseudo-Label Learning for Test-Time Adaptation. In *The Twelfth International Conference on Learning Representations*.

Tirer, T.; and Bruna, J. 2022. Extended unconstrained features model for exploring deep neural collapse. In *ICML*, 21478–21505. PMLR.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *NeurIPS*.

Wang, L.; Ma, C.; Feng, X.; Zhang, Z.; Yang, H.; Zhang, J.; Chen, Z.; Tang, J.; Chen, X.; Lin, Y.; et al. 2024. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6): 186345.

Wen, K.; Ma, T.; and Li, Z. 2023. Sharpness Minimization Algorithms Do Not Only Minimize Sharpness To Achieve Better Generalization. *arXiv preprint arXiv:2307.11007*.

Wu, T.; Qian, H.; Liu, Z.; Zhou, J.; and Zhou, A. 2023. Bi-objective evolutionary Bayesian network structure learning via skeleton constraint. *Frontiers of Computer Science*, 17(6): 176350.

Yao, Z.; Gholami, A.; Keutzer, K.; and Mahoney, M. W. 2020. Pyhessian: Neural networks through the lens of the Hessian. In *2020 IEEE international conference on big data (Big data)*, 581–590. IEEE.

Zhang, H.; Cisse, M.; Dauphin, Y. N.; and Lopez-Paz, D. 2017. Mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*.

Zhao, Y.; Zhang, H.; and Hu, X. 2022. Penalizing gradient norm for efficiently improving generalization in deep learning. In *ICML*. PMLR.

Zhou, J.; Li, X.; Ding, T.; You, C.; Qu, Q.; and Zhu, Z. 2022. On the optimization landscape of neural collapse under MSE loss: Global optimality with unconstrained features. In *ICML*. PMLR.

Zhou, Y.; Liang, D.; Chen, S.; Huang, S.-J.; Yang, S.; and Li, C. 2023. Improving lens flare removal with general-purpose pipeline and multiple light sources recovery. In *Proceedings of the IEEE/CVF international conference on computer vision*, 12969–12979.

Zhou, Y.; Song, L.; Wang, B.; and Chen, W. 2024. MetaGPT: Merging Large Language Models Using Model Exclusive Task Arithmetic. *arXiv preprint arXiv:2406.11385*.

Zhu, Z.; Ding, T.; Zhou, J.; Li, X.; You, C.; Sulam, J.; and Qu, Q. 2021. A geometric analysis of neural collapse with unconstrained features. *NeurIPS*.