Orthogonal SVD Covariance Conditioning and Latent Disentanglement

Yue Song, *Member, IEEE,* Nicu Sebe, *Senior Member, IEEE,* Wei Wang, *Member, IEEE*

Abstract—Inserting an SVD meta-layer into neural networks is prone to make the covariance ill-conditioned, which could harm the model in the training stability and generalization abilities. In this paper, we systematically study how to improve the covariance conditioning by enforcing orthogonality to the Pre-SVD layer. Existing orthogonal treatments on the weights are first investigated. However, these techniques can improve the conditioning but would hurt the performance. To avoid such a side effect, we propose the Nearest Orthogonal Gradient (NOG) and Optimal Learning Rate (OLR). The effectiveness of our methods is validated in two applications: decorrelated Batch Normalization (BN) and Global Covariance Pooling (GCP). Extensive experiments on visual recognition demonstrate that our methods can simultaneously improve covariance conditioning and generalization. The combinations with orthogonal weight can further boost the performance. Moreover, we show that our orthogonality techniques can benefit generative models for better latent disentanglement through a series of experiments on various benchmarks. Code is available at: https://github.com/KingJamesSong/OrthoImproveCond.

Index Terms—Differentiable SVD, Covariance Conditioning, Orthogonality Constraint, Unsupervised Latent Disentanglement

✦

1 INTRODUCTION

The Singular Value Decomposition (SVD) can factorize a matrix into orthogonal eigenbases and non-negative singular values, serving as an essential step for many matrix operations. Recently in computer vision and deep learning, many approaches integrated the SVD as a meta-layer in the neural networks to perform some differentiable spectral transformations, such as the matrix square root and inverse square root. The applications arise in a wide range of methods, including Global Covariance Pooling (GCP) [1], [2], [3], decorrelated Batch Normalization (BN) [4], [5], [6], Whitening an Coloring Transform (WCT) for universal style transfer [7], [8], [9], and Perspective-n-Point (PnP) problems [10], [11], [12].

For the input feature map $\mathbf{X}$ passed to the SVD meta-layer, one often first computes the covariance of the feature as $\mathbf{X}\mathbf{X}^T$. This can ensure that the covariance matrix is both symmetric and positive semi-definite, which does not involve any negative eigenvalues and leads to the identical left and right eigenvector matrices. However, it is observed that inserting the SVD layer into deep models would typically make the covariance very ill-conditioned [2], resulting in deleterious consequences on the stability and optimization of the training process. For a given covariance $\mathbf{A}$, its conditioning is measured by the condition number:

$$\kappa(\mathbf{A}) = \sigma_{max}(\mathbf{A})\sigma_{min}^{-1}(\mathbf{A}) \quad (1)$$

where $\sigma(\cdot)$ denotes the eigenvalue of the matrix. Mathematically speaking, the condition number measures how sensitive the SVD is to the errors of the input. Matrices with low condition numbers are considered **well-conditioned**, while

matrices with high condition numbers are said to be **ill-conditioned**. Specific to neural networks, the ill-conditioned covariance matrices are harmful to the training process in several aspects, which we will analyze in detail later.

This phenomenon was first observed in the GCP methods by [2], and we found that it generally extrapolates to other SVD-related tasks, such as decorrelated BN. Fig. 1 depicts the covariance conditioning of these two tasks throughout the training. As can be seen, the integration of the SVD layer makes the generated covariance very ill-conditioned ($\approx 1e12$ for decorrelated BN and $\approx 1e16$ for GCP). By contrast, the conditioning of the approximate solver, *i.e.,* Newton-Schulz iteration (NS iteration) [13], is about $1e5$ for decorrelated BN and is around $1e15$ for GCP, while the standard BN only has a condition number of $1e3$.

Fig. 1: The covariance conditioning of the SVD meta-layer during the training process in the tasks of decorrelated BN (*left*) and GCP (*Right*). The decorrelated BN is based on ResNet-50 and CIFAR100, while ImageNet and ResNet-18 are used for the GCP.

Ill-conditioned covariance matrices can harm the training of the network in both the forward pass (FP) and the backward pass (BP). For the FP, mainly the SVD solver is influenced in terms of stability and accuracy. Since the ill-conditioned covariance has many trivially-small eigenvalues, it is difficult for an SVD solver to accurately estimate them

• *Yue Song and Nicu Sebe are with Department of Information Engineering and Computer Science, University of Trento, Trento 38123, Italy. Wei Wang is with Beijing Jiaotong University, Beijing, China. Wei Wang is the corresponding author.*
E-mail: {yue.song, nicu.sebe}@unitn.it, wei.wang@bjtu.edu.cn

and large round-off errors are likely to be triggered, which might hurt the network performances. Moreover, the very imbalanced eigenvalue distribution can easily make the SVD solver fail to converge and cause the training failure [2], [14]. For the BP, as pointed out in [4], [15], [16], the feature covariance is closely related to the Hessian matrix during the backpropagation. As the error curvature is given by the eigenvalues of the Hessian matrix [17], for the ill-conditioned Hessian, the Gradient Descent (GD) step would bounce back and forth in high curvature directions (large eigenvalues) and make slow progress in low curvature directions (small eigenvalues). As a consequence, the ill-conditioned covariance could cause slow convergence and oscillations in the optimization landscape. The generalization abilities of a deep model are thus harmed.

Due to the data-driven learning nature and the highly non-linear transform of deep neural networks, directly giving the analytical form of the covariance conditioning is intractable. Some simplifications have to be performed to ease the investigation. Since the covariance is generated and passed from the previous layer, the previous layer is likely to be the most relevant to the conditioning. Therefore, we naturally limit our focus to the Pre-SVD layer, *i.e.,* the layer before the SVD layer. To further simplify the analysis, we study the Pre-SVD layer in two consecutive training steps, which can be considered as a mimic of the whole training process. Throughout the paper, we mainly investigate some meaningful manipulations on the weight, the gradient, and the learning rate of the Pre-SVD layer in two sequential training steps. *Under our Pre-SVD layer simplifications, one promising direction to improve the conditioning is enforcing orthogonality on the weights.* Orthogonal weights have the norm-preserving property, which could improve the conditioning of the feature matrix. This technique has been widely studied in the literature of stable training and Lipschitz networks [18], [19], [20]. We select some representative methods and validate their effectiveness in the task of decorrelated BN. Our experiment reveals that these orthogonal techniques can greatly improve the covariance conditioning, but could only bring marginal performance improvements and even slight degradation. *This indicates that when the representation power of weight is limited, the improved conditioning does not necessarily lead to better performance. Orthogonalizing only the weight is thus insufficient to improve the generalization.* Instead of seeking orthogonality constraints on the weights, we propose our Nearest Orthogonal Gradient (NOG) and Optimal Learning Rate (OLR). These two techniques explore the orthogonality possibilities about the learning rate and the gradient. More specifically, our NOG modifies the gradient of the Pre-SVD layer into its nearest-orthogonal form and keeps the GD direction unchanged. On the other hand, the proposed OLR dynamically changes the learning rate of the Pre-SVD layer at each training step such that the updated weight is as close to an orthogonal matrix as possible. The experimental results demonstrate that the proposed two techniques not only significantly improve the covariance conditioning but also bring obvious improvements in the validation accuracy of both GCP and decorrelated BN. Moreover, when combined with the orthogonal weight treatments, the performance can have further improvements.

Besides the application on differentiable SVD, we propose

that our orthogonality techniques can be also used for unsupervised latent disentanglement of Generative Adversarial Networks (GANs) [21]. Recent works [22], [23] revealed that the latent disentanglement of GANs is closely related to the gradient or weight of the first projector after the latent code. In particular, the eigenvectors of the gradient or weight can be viewed as closed-formed solutions of interpretable directions [23]. This raises the need for enforcing orthogonal constraints on the projector. *As shown in Fig. 5, compared with non-orthogonal matrices, orthogonal matrices can lead to more disentangled representations and more precise attributes due to the property of equally-important eigenvectors.* Motivated by this observation, we propose to enforce our NOG and OLR as orthogonality constraints in generative models. Extensive experiments on various architectures and datasets demonstrate that our methods indeed improve the disentanglement ability of identifying semantic attributes and achieve state-of-the-art performance against other disentanglement approaches.

The main contributions are summarized below:

- We systematically study the problem of how to improve the covariance conditioning of the SVD meta-layer. We propose our Pre-SVD layer simplification to investigate this problem from the perspective of orthogonal constraints.
- We explore different techniques of orthogonal weights to improve the covariance conditioning. Our experiments reveal that these techniques could improve the conditioning but would harm the generalization abilities due to the limitation on the representation power of weight.
- We propose the nearest orthogonal gradient and optimal learning rate. The experiments on GCP and decorrelated BN demonstrate that these methods can attain better covariance conditioning and improved generalization. Their combinations with weight treatments can further boost the performance.
- We show that our proposed orthogonality approaches can be applied on the GANs projector for improved latent disentanglement ability of discovering precise semantic attributes, which opens the way for new applications of orthogonality techniques.

This paper is an extension of the previous conference paper [24]. In [24], we propose two orthogonality techniques and demonstrate that these methods can simultaneously improve the covaraince conditioning and generalization abilities of the SVD meta-layer. This journal extension motivates and proposes that these techniques can be also applied in generative models for better latent disentanglement. This point is validated through extensive experiments on various generative architectures and datasets. Moreover, we also investigate the probability of occurrence of our OLR throughout the training and show that the evaluation results agree well with our theoretical analysis.

The rest of the paper is organized as follows: Sec. 2 describes the related work in differentiable matrix decomposition, orthogonality applications, and unsupervised latent disentanglement. Sec. 3 introduces our Pre-SVD layer simplification and orthogonal weight treatments, and Sec. 4 presents the proposed orthogonality techniques. Sec. 5 motivates why orthogonality can improve latent disentanglement. Sec. 6 provides experimental results and some in-depth analysis. Finally, Sec. 7 summarizes the conclusions.

2 RELATED WORK

2.1 Differentiable Matrix Decomposition

The differentiable matrix decomposition is widely used in neural networks as a spectral meta-layer. Ionescu *et al.* [25], [26] first propose the theory of matrix back-propagation and laid a foundation for the follow-up research. In deep neural networks, the transformation of matrix square root and its inverse are often desired due to the appealing spectral property. Their applications cover a wide range of computer vision tasks [6], [27]. To avoid the huge time consumption of the SVD, some iterative methods are also developed to approximate the solution [6], [13], [27]. Recently Song *et al.* [28] propose a dedicated eigen-solver for improving the computation speed of batched matrices. In [4], [5], [6], [8], [29], [30], the inverse square root is used in the ZCA whitening transform to whiten the feature map, which is also known as the decorrelated BN. The Global Covariance Pooling (GCP) models [1], [2], [3], [31], [32], [33], [34] compute the matrix square root of the covariance as a spectral normalization, which achieves impressive performances on some recognition tasks, including large-scale visual classification [1], [2], [6], [33], fine-grained visual categorization [1], [31], [34], and video action recognition [3]. The Whitening and Coloring Transform (WCT), which uses both the matrix square root and inverse square root, is usually adopted in some image generation tasks such as neural style transfer [7], [9], image translation [35], [36], and domain adaptation [37], [38]. In the geometric vision problems, the differentiable SVD is usually applied to estimate the fundamental matrix and the camera pose [11], [12], [39]. Besides the SVD-based factorization, differentiating Cholesky decomposition [40] and some low-rank decomposition is used to approximate the attention mechanism [41], [42], [43] or to learn the constrained representations [44], [45].

2.2 Orthogonality in Neural Network

Orthogonal weights have the benefit of the norm-preserving property, *i.e.*, the relation $||\mathbf{W}\mathbf{A}||_F = ||\mathbf{A}||_F$ holds for any orthogonal $\mathbf{W}$. When it comes to deep neural networks, such a property can ensure that the signal stably propagates through deep networks without either exploding or vanishing gradients [46], [47], which could speed up convergence and encourage robustness and generalization. In general, there are three ways to enforce orthogonality to a layer: orthogonal weight initialization [18], [48], [49], orthogonal regularization [19], [50], [51], [51], [52], and explicit orthogonal weight via Carley transform or matrix exponential [20], [53], [54]. Among these techniques, orthogonal regularization and orthogonal weight are most commonly used as they often bring some practical improvements in generalization. Since the covariance is closely related to the weight matrix of the Pre-SVD layer, enforcing the orthogonality constraint could help to improve the covariance conditioning of the SVD meta-layer. We will choose some representative methods and validate their impact in Sec. 3.2.

Notice that the focus of existing literature is different from our work. The orthogonality constraints are often used to improve the Lipschitz constants of the neural network layers, which is expected to improve the visual quality in image generation [55], [56], to allow for better adversarial robustness [20], [57], and to improve generalization abilities [19], [58]. Our work is concerned with improving the covariance conditioning and generalization performance. Moreover, the orthogonality literature mainly investigates how to enforce orthogonality to weight matrices, whereas less attention is put on the gradient and learning rate. In Sec. 4, we will explore such possibilities and propose our solutions: nearest orthogonal gradient and optimal learning rate which is optimal in the sense that the updated weight is as close to an orthogonal matrix as possible.

2.3 Unsupervised Latent Disentanglement of GANs

Interpreting latent spaces of GAN models in an unsupervised manner has received wide attention recently [59], [60], [61], [62]. This can help to identify semantic attributes of the image and to have precise control of the generation process, which could benefit both local and global image editing tasks [22], [63]. Voynov *et al.* [61] proposed to jointly learn a set of directions and an extra classifier such that the interpretable directions can be recognized. In [64], the authors proposed to perform PCA on the sampled data to capture the interpretable directions. More recently, Shen *et al.* [23] and Zhu *et al.* [22] pointed out that the semantic attributes are characterized by the eigenvectors of the weight or gradient of the first projector after the latent code. Motivated by this observation, we propose to enforce our orthogonality techniques to the gradient and weight matrices.

Besides our orthogonality techniques, a few works have applied implicit orthogonality into the training process of GANs to attain more disentangled representations [65], [66], [67], [68]. In [65], [68], the authors proposed to add orthogonal Hessian/Jacobian penalty to encourage disentanglement. He *et al.* [67] designed a dedicated GAN architecture where multi-level latent codes and orthogonal weight constraints are applied. Different from previous approaches, our orthogonality treatments do not rely on any implicit regularization. Instead, our NOG explicitly maps the original gradient into the nearest-orthogonal form, while our OLR keeps the updated weight in the closest form to orthogonal matrices.

3 PRE-SVD LAYER AND WEIGHT TREATMENTS

In this section, we first motivate our simplification of the Pre-SVD layer, and then validate the efficacy of some representative weight treatments.

3.1 Pre-SVD Layer Simplification

The neural network consists of a sequential of non-linear layers where the learning of each layer is data-driven. Stacking these layers leads to a highly non-linear and complex transform, which makes directly analyzing the covariance conditioning intractable. To solve this issue, we have to perform some simplifications.

Our simplifications involve limiting the analysis only to the layer previous to the SVD layer (which we dub as the Pre-SVD layer) in two consecutive training steps. The Pre-SVD layer directly determines the conditioning of the generated covariance, while the two successive training steps are a mimic of the whole training process. The idea is to simplify the complex transform by analyzing the sub-model

(two layers) and the sub-training (two steps), which can be considered as an "abstract representation" of the deep model and its complete training.

Let $\mathbf{W}$ denote the weight matrix of the Pre-SVD layer. Then for the input $\mathbf{X}_l$ passed to the layer, we have:

$$\mathbf{X}_{l+1} = \mathbf{W}\mathbf{X}_l + \mathbf{b} \tag{2}$$

where $\mathbf{X}_{l+1}$ is the feature passed to the SVD layer, and $\mathbf{b}$ is the bias vector. Since the bias $\mathbf{b}$ has a little influence here, we can sufficiently omit it for simplicity. The covariance in this step is computed as $\mathbf{W}\mathbf{X}_l\mathbf{X}_l^T\mathbf{W}^T$. After the BP, the weight matrix is updated as $\mathbf{W} - \eta\frac{\partial l}{\partial \mathbf{W}}$ where η denotes the learning rate of the layer. Let $\mathbf{Y}_l$ denote the passed-in feature of the next training step. Then the covariance is calculated as:

$$
\begin{aligned}
\mathbf{C} &= \left((\mathbf{W} - \eta\frac{\partial l}{\partial \mathbf{W}}) \cdot \mathbf{Y}_l \right)\left((\mathbf{W} - \eta\frac{\partial l}{\partial \mathbf{W}}) \cdot \mathbf{Y}_l \right)^T \\
&= \mathbf{W}\mathbf{Y}_l\mathbf{Y}_l^T\mathbf{W}^T - \eta\frac{\partial l}{\partial \mathbf{W}}\mathbf{Y}_l\mathbf{Y}_l^T\mathbf{W}^T \\
&\quad - \eta\mathbf{W}\mathbf{Y}_l\mathbf{Y}_l^T(\frac{\partial l}{\partial \mathbf{W}})^T + \eta^2\frac{\partial l}{\partial \mathbf{W}}\mathbf{Y}_l\mathbf{Y}_l^T(\frac{\partial l}{\partial \mathbf{W}})^T
\end{aligned} \tag{3}
$$

where $\mathbf{C}$ denotes the generated covariance of the second step. Now the problem becomes how to stop the new covariance $\mathbf{C}$ from becoming worse-conditioned than $\mathbf{W}\mathbf{X}_l\mathbf{X}_l^T\mathbf{W}^T$. In eq. (3), three variables could influence the conditioning: the weight $\mathbf{W}$, the gradient of the last step $\frac{\partial l}{\partial \mathbf{W}}$, and the learning rate η of this layer. Among them, the weight $\mathbf{W}$ seems to be the most important as it contributes to three terms of eq. (3). Moreover, the first term $\mathbf{W}\mathbf{Y}_l\mathbf{Y}_l^T\mathbf{W}^T$ computed by $\mathbf{W}$ is not attenuated by η or η^2 like the other terms. Therefore, it is natural to first consider manipulating $\mathbf{W}$ such that the conditioning of $\mathbf{C}$ could be improved.

3.2 General Treatments on Weights

In the literature of enforcing orthogonality to the neural network, there are several techniques to improve the conditioning of the weight $\mathbf{W}$. Now we introduce some representatives methods and validate their impacts.

3.2.1 Spectral Normalization (SN)

In [56], the authors propose a normalization method to stabilize the training of generative models [21] by dividing the weight matrix with its largest eigenvalue. The process is defined as:

$$\mathbf{W}/\sigma_{max}(\mathbf{W}) \tag{4}$$

Such a normalization can ensure that the spectral radius of $\mathbf{W}$ is always 1, *i.e.*, $\sigma_{max}(\mathbf{W})=1$. This could help to reduce the conditioning of the covariance since we have $\sigma_{max}(\mathbf{W}\mathbf{Y}_l)=\sigma_{max}(\mathbf{Y}_l)$ after the spectral normalization.

3.2.2 Orthogonal Loss (OL)

Besides limiting the spectral radius of $\mathbf{W}$, enforcing orthogonality constraint could also improve the covariance conditioning. As orthogonal matrices are norm-preserving (*i.e.*, $||\mathbf{W}\mathbf{Y}_l||_F=||\mathbf{W}||_F$), lots of methods have been proposed to encourage orthogonality on weight matrices for more stable training and better signal-preserving property [19], [20], [51], [54], [69]. One common technique is to apply *soft* orthogonality [19] by the following regularization:

$$l = ||\mathbf{W}\mathbf{W}^T - \mathbf{I}||_F \tag{5}$$

Fig. 2: Covariance conditioning during the training process. All weight treatments can improve conditioning.

Methods	mean±std	min
SVD	19.99±0.16	19.80
SVD + SN	19.94±0.33	19.60
SVD + OL	**19.73±0.28**	**19.54**
SVD + OW	20.06±0.17	19.94
NS iteration	19.45±0.33	19.01

TABLE 1: Performance of different weight treatments on ResNet-50 and CIFAR100 based on 10 runs.

This extra loss is added in the optimization objective to encourage more orthogonal weight matrices. However, since the constraint is achieved by regularization, the weight matrix is not exactly orthogonal at each training step.

3.2.3 Orthogonal Weights (OW)

Instead of applying *soft* orthogonality by regularization, some methods can explicitly enforce *hard* orthogonality to the weight matrices [20], [54]. The technique of [20] is built on the mathematical property: for any skew-symmetric matrix, its matrix exponential is an orthogonal matrix.

$$\exp(\mathbf{W} - \mathbf{W}^T)\exp(\mathbf{W} - \mathbf{W}^T)^T = \mathbf{I} \tag{6}$$

where the operation of $\mathbf{W} - \mathbf{W}^T$ is to make the matrix skew-symmetric, *i.e.*, the relation $\mathbf{W} - \mathbf{W}^T = -(\mathbf{W} - \mathbf{W}^T)^T$ always holds. Then $\exp(\mathbf{W} - \mathbf{W}^T)$ is used as the weight. This technique explicitly constructs the weight as an orthogonal matrix. The orthogonal constraint is thus always satisfied during the training.

We apply the above three techniques in the experiment of decorrelated BN. Fig. 2 displays the covariance conditioning throughout the training, and Table 1 presents the corresponding validation errors. As can be seen, all of these techniques attain much better conditioning, but the performance improvements are not encouraging. The SN reduces the conditioning to around 10^5, while the validation error marginally improves. The *soft* orthogonality by the OL brings slight improvement on the performance despite some variations in the conditioning. The conditioning variations occur because the orthogonality constraint by regularization is not strictly enforced. Among the weight treatments, the *hard* orthogonality by the OW achieves the best covariance conditioning, continuously maintaining the condition number around 10^3 throughout the training. However, the OW slightly hurts the validation error. This implies that better covariance conditioning does not necessarily correspond to the improved performance, and orthogonalizing only the weight cannot improve the generalization. *We conjecture that enforcing strict orthogonality only on the weight might limit its representation power.* Nonetheless, as will be discussed in Sec. 4.1, the side effect can be canceled when we simultaneously orthogonalize the gradient.

4 NEAREST ORTHOGONAL GRADIENT AND OPTIMAL LEARNING RATE

This section introduces our proposed techniques on modifying the gradient and learning rate of the Pre-SVD layer. The

combinations with weight treatments are also discussed.

4.1 Nearest Orthogonal Gradient (NOG)

As discussed in Sec. 3.1, the covariance conditioning is also influenced by the gradient $\frac{\partial l}{\partial \mathbf{W}}$. However, existing literature mainly focuses on orthogonalizing the weights. To make the gradient also orthogonal, we propose to find the nearest-orthogonal gradient of the Pre-SVD layer. Different matrix nearness problems have been studied in [70], and the nearest-orthogonal problem is defined as:

$$\min_{\mathbf{R}} ||\frac{\partial l}{\partial \mathbf{W}} - \mathbf{R}||_{\mathrm{F}} \; subject \; to \; \mathbf{R}\mathbf{R}^T = \mathbf{I} \qquad (7)$$

where $\mathbf{R}$ is the seeking solution. To obtain such an orthogonal matrix, we can construct the error function as:

$$e(\mathbf{R}) = Tr\Big(\big(\frac{\partial l}{\partial \mathbf{W}} - \mathbf{R}\big)^T\big(\frac{\partial l}{\partial \mathbf{W}} - \mathbf{R}\big)\Big) + Tr\Big(\mathbf{\Sigma}\mathbf{R}^T\mathbf{R} - \mathbf{I}\Big) \quad (8)$$

where $Tr(\cdot)$ is the trace measure, and $\mathbf{\Sigma}$ denotes the symmetric matrix Lagrange multiplier. The closed-form solution is given by:

$$\mathbf{R} = \frac{\partial l}{\partial \mathbf{W}}\Big((\frac{\partial l}{\partial \mathbf{W}})^T \frac{\partial l}{\partial \mathbf{W}}\Big)^{-\frac{1}{2}} \qquad (9)$$

The detailed derivation is given in the supplementary material. If we have the SVD of the gradient ($\mathbf{U}\mathbf{S}\mathbf{V}^T = \frac{\partial l}{\partial \mathbf{W}}$), the solution can be further simplified as:

$$\mathbf{R} = \mathbf{U}\mathbf{S}\mathbf{V}^T(\mathbf{V}\mathbf{S}^{-1}\mathbf{V}^T) = \mathbf{U}\mathbf{V}^T \qquad (10)$$

As indicated above, the nearest orthogonal gradient is achieved by setting the singular value matrix to the identity matrix, *i.e.*, setting $\mathbf{S}$ to $\mathbf{I}$. Notice that only the gradient of Pre-SVD layer is changed, while that of the other layers is not modified. Our proposed NOG can bring several practical benefits.

4.1.1 Orthogonal Constraint and Optimal Conditioning

The orthogonal constraint is exactly enforced on the gradient as we have $(\mathbf{U}\mathbf{V}^T)^T\mathbf{U}\mathbf{V}^T = \mathbf{I}$. Since we explicitly set all the singular values to 1, the optimal conditioning is also achieved, *i.e.*, $\kappa(\frac{\partial l}{\partial \mathbf{W}}) = 1$. This could help to improve the conditioning.

4.1.2 Keeping Gradient Descent Direction Unchanged

In the high-dimensional optimization landscape, the many curvature directions (GD directions) are characterized by the eigenvectors of gradient ($\mathbf{U}$ and $\mathbf{V}$). Although our modification changes the gradient, the eigenvectors and the GD directions are untouched. In other words, our NOG only adjusts the step size in each GD direction. This indicates that the modified gradients will not harm performance.

4.1.3 Combination with Weight Treatments

Our orthogonal gradient and the previous weight treatments are complementary. They can be jointly used to simultaneously orthogonalize the gradient and weight. In the following, we will validate their joint impact on the conditioning and performance.

Fig. 3 and Table 2 present the covariance conditioning of decorrelated BN and the corresponding validation errors,

Fig. 3: Covariance conditioning during the training process using orthogonal gradient and weight treatments.

Methods	mean±std	min
SVD	19.99±0.16	19.80
SVD + NOG	19.43±0.24	19.15
SVD + NOG + SN	19.43±0.21	19.20
SVD + NOG + OL	20.14±0.39	19.54
SVD + NOG + OW	**19.22±0.28**	**18.90**
NS iteration	19.45±0.33	19.01

TABLE 2: Performance of gradient treatments on ResNet-50 and CIFAR100. Each result is based on 10 runs.

respectively. As we can observe, solely using the proposed NOG can largely improve the covariance conditioning, decreasing the condition number from 10^{12} to 10^6. Though this improvement is not as significant as the orthogonal constraints (*e.g.*, OL and OW), our NOG can benefit more the generalization abilities, leading to the improvement of validation error by 0.6%. Combining the SN with our NOG does not lead to obvious improvements in either the conditioning or validation errors, whereas the joint use of NOG and OL harms the network performances. This is because the orthogonality constraint by loss might not be enforced under the gradient manipulation. When our NOG is combined with the OW, the side effect of using only OW is eliminated and the performance is further boosted by 0.3%. This phenomenon demonstrates that when the gradient is orthogonal, applying the orthogonality constraint to the weight could also be beneficial to the generalization.

4.2 Optimal Learning Rate (OLR)

So far, we only consider orthogonalizing $\mathbf{W}$ and $\frac{\partial l}{\partial \mathbf{W}}$ separately, but how to jointly optimize $\mathbf{W} - \eta\frac{\partial l}{\partial \mathbf{W}}$ has not been studied yet. Actually, it is desired to choose an appropriate learning rate η such that the updated weight is close to an orthogonal matrix. To this end, we need to achieve the following objective:

$$\min_{\eta} ||(\mathbf{W} - \eta\frac{\partial l}{\partial \mathbf{W}})(\mathbf{W} - \eta\frac{\partial l}{\partial \mathbf{W}})^T - \mathbf{I}||_{\mathrm{F}} \qquad (11)$$

This optimization problem can be more easily solved in the vector form. Let $\mathbf{w}$, $\mathbf{i}$, and $\mathbf{l}$ denote the vectorized $\mathbf{W}$, $\mathbf{I}$, and $\frac{\partial l}{\partial \mathbf{W}}$, respectively. Then we construct the error function as:

$$e(\eta) = \Big((\mathbf{w} - \eta\mathbf{l})^T(\mathbf{w} - \eta\mathbf{l}) - \mathbf{i}\Big)^T\Big((\mathbf{w} - \eta\mathbf{l})^T(\mathbf{w} - \eta\mathbf{l}) - \mathbf{i}\Big) \quad (12)$$

Expanding and differentiating the equation w.r.t. η lead to:

$$\frac{de(\eta)}{d\eta} \approx -4\mathbf{w}\mathbf{w}^T\mathbf{l}^T\mathbf{w} + 4\eta\mathbf{w}\mathbf{w}^T\mathbf{l}^T\mathbf{l} + 8\eta\mathbf{l}^T\mathbf{w}\mathbf{l}^T\mathbf{w} = 0$$

$$\eta^\star \approx \frac{\mathbf{w}^T\mathbf{w}\mathbf{l}^T\mathbf{w}}{\mathbf{w}^T\mathbf{w}\mathbf{l}^T\mathbf{l} + 2\mathbf{l}^T\mathbf{w}\mathbf{l}^T\mathbf{w}}$$

$$(13)$$

where some higher-order terms are neglected. The detailed derivation is given in the supplementary material. Though the proposed OLR yields the updated weight nearest to an orthogonal matrix theoretically, the value of $\eta^\star$ is unbounded for arbitrary $\mathbf{w}$ and $\mathbf{l}$. Directly using $\eta^\star$ might cause unstable training. To avoid this issue, we propose to use the OLR

Fig. 4: Covariance conditioning during the training process using optimal learning rate and hybrid treatments.

TABLE 3: Performance of optimal learning rate on ResNet-50 and CIFAR100 based on 10 runs.

Methods	mean±std	min
SVD	19.99±0.16	19.80
SVD + OLR	19.50±0.39	18.95
SVD + NOG + OLR	19.77±0.27	19.36
SVD + OW + OLR	20.61±0.22	20.43
SVD + NOG + OW +OLR	**19.05±0.31**	**18.77**
NS iteration	19.45±0.33	19.01

only when its value is smaller than the learning rate of other layers. Let lr denote the learning rate of the other layers. The switch process can be defined as:

$$\eta = \begin{cases} \eta^\star & if \ \eta^\star < lr \\ lr & otherwise \end{cases} \tag{14}$$

4.2.1 Combination with Weight/Gradient Treatments

When either the weight or the gradient is orthogonal, our OLR needs to be carefully used. When only $\mathbf{W}$ is orthogonal, $\mathbf{w}^T\mathbf{w}$ is a small constant and it is very likely to have $\mathbf{w}^T\mathbf{w} \ll \mathbf{l}^T\mathbf{w}$. Consequently, we have $\mathbf{w}^T\mathbf{w}\mathbf{l}^T\mathbf{w} \ll \mathbf{l}^T\mathbf{w}\mathbf{l}^T\mathbf{w}$ and $\eta^\star$ will attenuate to zero. Similarly for orthogonal gradient, we have $\mathbf{w}^T\mathbf{w}\mathbf{l}^T\mathbf{w} \ll \mathbf{l}^T\mathbf{w}\mathbf{l}^T\mathbf{l}$ and this will cause $\eta^\star$ close to zero. Therefore, the proposed OLR cannot work when either the weight or gradient is orthogonal. Nonetheless, we note that if both $\mathbf{W}$ and $\frac{\partial l}{\partial \mathbf{W}}$ are orthogonal, our $\eta^\star$ is bounded. Specifically, we have:

Proposition 1. *When both $\mathbf{W}$ and $\frac{\partial l}{\partial \mathbf{W}}$ are orthogonal, $\eta^\star$ is both upper and lower bounded. The upper bound is $\frac{N^2}{N^2+2}$ and the lower bound is $\frac{1}{N^2+2}$ where N denotes the row dimension of $\mathbf{W}$.*

We give the detailed proof in the supplementary material. Obviously, the upper bound of $\eta^\star$ is smaller than 1. For the lower bound, since the row dimension of N is often large (*e.g.*, 64), the lower bound of $\eta^\star$ can be according very small (*e.g.*, $2e-4$). This indicates that our proposed OLR could also give a small learning rate even in the later stage of the training process.

In summary, the optimal learning rate is set such that the updated weight is optimal in the sense that it become as close to an orthogonal matrix as possible. In particular, it is suitable when both the gradient and weight are orthogonal.

We give the covariance conditioning and the validation errors in Fig. 4 and in Table 3, respectively. Our proposed OLR significantly reduces the condition number to 10^4 and improves the validation error by 0.5%. When combined with either orthogonal weight or orthogonal gradient, there is a slight degradation on the validation errors. This meets our expectation as $\eta^\star$ would attenuate to zero in both cases. However, when both $\mathbf{W}$ and $\frac{\partial l}{\partial \mathbf{W}}$ are orthogonal, jointly using our OLR achieves the best performance, outperforming only OLR by 0.5% and beating OW+NOG by 0.2%. This observation confirms that the proposed OLR works well for simultaneously orthogonal $\mathbf{W}$ and $\frac{\partial l}{\partial \mathbf{W}}$.

5 ORTHOGONALITY FOR UNSUPERVISED LATENT DISENTANGLEMENT

In this section, we motivate why orthogonal treatments (orthogonal weight or gradient) would help in unsupervised latent disentanglement of GANs.

5.1 Image Manipulation in Latent Space of GANs

The latent space of GANs encodes rich semantics information, which can be used for image editing via vector arithmetic property [71]. Consider a generator $G(\cdot)$ and the latent code $\mathbf{z} \in \mathbb{R}^d$. The image manipulation is achieved by finding a semantically meaningful direction $\mathbf{n}$ such that

$$\texttt{edit}(G(\mathbf{z})) = G(\mathbf{z} + \alpha\mathbf{n}) \tag{15}$$

where $\texttt{edit}(\cdot)$ denotes the image editing process, and α represents the perturbation strength. That being said, moving the latent code $\mathbf{z}$ along with the interpretable direction $\mathbf{n}$ should change the targeting semantic concept of the image. Since the generator $G(\cdot)$ is highly non-linear and complex, directly analyzing $G(\mathbf{z} + \alpha\mathbf{n})$ is intractable. To avoid this issue, existing approaches propose to simplify the analysis by considering only the first projector matrix $G_1(\cdot)$ or performing local Taylor expansion [22], [23], [72], [73].

Eigenvector of the first projector. In SeFa [23], the authors propose to seek for interpretable directions from the eigenvector of the first projector matrix. Specifically, they consider the affine transformation of the layer as:

$$G_1(\mathbf{z} + \alpha\mathbf{n}) = \mathbf{A}\mathbf{z} + \mathbf{b} + \alpha\mathbf{A}\mathbf{n} = G_1(\mathbf{z}) + \alpha\mathbf{A}\mathbf{n} \tag{16}$$

where $\mathbf{A}$ is the weight matrix. Intuitively, a meaningful direction should lead to large variations of the generated image. So the problem can be cast into an optimization problem as:

$$\mathbf{n}^\star = \arg\max ||\mathbf{A}\mathbf{n}||^2 \tag{17}$$

All the possible closed-form solution correspond to the eigenvector of $\mathbf{A}^T\mathbf{A}$. The top-k eigenvectors are thus selected as the interpretable directions for image manipulation.

Eigenvector of the Jacobian. LowRankGAN [22] proposes to linearly approximate $G(\mathbf{z} + \alpha\mathbf{n})$ by the Taylor expansion as:

$$G(\mathbf{z} + \alpha\mathbf{n}) \approx G(\mathbf{z}) + \alpha\mathbf{J}_\mathbf{z}\mathbf{n} \tag{18}$$

where $\mathbf{J}_\mathbf{z}$ is the Jacobian matrix w.r.t. the latent code $\mathbf{z}$. Similarly to the deduction of eq. (17), the closed-form solution is given by the eigenvector of $\mathbf{J}_\mathbf{z}^T\mathbf{J}_\mathbf{z}$.

The above two formulations illustrate how the weight and gradient matrices are related with the interpretable direction discovery. Currently, most GAN models do not enforce orthogonality to their architectures. Now we turn to explaining the concrete benefit of introducing orthogonality to the latent disentanglement.

5.2 Usefulness of Orthogonality

Though few previous works have applied implicit orthogonality as regularization in GANs [61], [65], [67], [68], there are no generally accepted explanations on how the orthogonality is related to the disentangled representations. Here we give an intuitive explanation. As discussed in the above image manipulation modelling, the eigenvectors of weight and

Fig. 5: Illustration of the benefit of orthogonality in latent disentanglement. As revealed in [22], [23], the interpretable directions of latent codes are the eigenvectors of weight or gradient matrices. For non-orthogonal matrices, the principle eigenvector is of the most importance, which would make this direction correspond to many semantic attributes. The other eigenvectors might fail to capture any semantic information. By contrast, the eigenvectors of orthogonal matrices are equally important. The network with the orthogonal weight/gradient is likely to learn more disentangled representations.

gradient matrices naturally imply the interpretable directions for latent disentanglement. For common non-orthogonal matrices, the importance of each eigenvector is characterized by the corresponding eigenvalue. Each eigenvector is not equally important and the first few ones would dominate the spectrum. This imbalance would cause most semantic attributes entangled in the first few directions. Fig. 5 top illustrates this phenomenon: *moving the latent code along with the top-1 eigenvector direction triggers changes of many semantic attributes. On the contrary, the small eigenvector direction does not indicate any semantic changes. The learned representation are thus deemed entangled.*

The orthogonal matrices can greatly relieve this issue thanks to the flat spectrum and equally-important eigenvectors. As shown in Fig. 5 bottom, when our NOG and OLR are applied, each direction of the orthogonal matrix is equally important and corresponds to one semantic attribute. Shifting the latent code in one direction only changes the targeting semantic concept, while the identity and other attributes are not touched. Enforcing orthogonality would lead to the superior disentanglement of learned representations.

Our proposed NOG and OLR can serve as strict orthogonal gradient constraint and *relaxed* orthogonal weight constraint, respectively. Enforcing them on the first layer after the latent code during the training process is very likely to lead to more disentangled representations. In Sec. 6.2, we apply these two techniques in various GAN architectures and benchmarks for unsupervised latent disentanglement.

6 EXPERIMENTS

6.1 Covariance Conditioning

We validate the proposed approaches in two applications: GCP and decorrelated BN. These two tasks are very representative because they have different usages of the SVD meta-layer. The GCP uses the matrix square root, while the decorrelated BN applies the inverse square root. In addition, the models of decorrelated BN often insert the SVD meta-layer at the beginning of the network, whereas the GCP models integrate the layer before the FC layer.

6.1.1 Decorrelated Batch Normalization

Fig. 6: The scheme of the modified ResNet for decorrelated BN. We reduce the kernel size of the first convolution layer from $7{\times}7$ to $3{\times}3$. The BN after this layer is replaced with our decorrelated BN layer.

We use ResNet-50 [74] as the backbone for the experiment on CIFAR10 and CIFAR100 [75]. The kernel size of the first convolution layer of ResNet is $7{\times}7$, which might not suit the low resolution of these two datasets (the images are only of size $32{\times}32$). To avoid this issue, we reduce the kernel size of the first convolution layer to $3{\times}3$. The stride is also decreased from 2 to 1. The BN layer after this layer is replace with our decorrelated BN layer (see Fig. 6). Let $\mathbf{X}{\in}\mathbb{R}^{C\times BHW}$ denotes the reshaped feature. The whitening transform is performed as:

$$\mathbf{X}_{whitened} = (\mathbf{X}\mathbf{X}^T)^{-\frac{1}{2}}\mathbf{X} \qquad (19)$$

Compared with the vanilla BN that only standardizes the data, the decorrelated BN can further eliminate the data correlation between each dimension.

Methods	CIFAR10		CIFAR100	
	mean±std	min	mean±std	min
SVD	4.35±0.09	4.17	19.99±0.16	19.80
SVD + SN	4.31±0.10	4.15	19.94±0.33	19.60
SVD + OL	4.28±0.07	4.23	19.73±0.28	19.54
SVD + OW	4.42±0.09	4.28	20.06±0.17	19.94
SVD + NOG	**4.15±0.06**	**4.04**	**19.43±0.24**	**19.15**
SVD + OLR	**4.23±0.17**	**3.98**	**19.50±0.39**	**18.95**
SVD + NOG + OW	**4.09±0.07**	**4.01**	**19.22±0.28**	**18.90**
SVD + NOG + OW + OLR	**3.93±0.09**	**3.85**	**19.05±0.31**	**18.77**
NS iteration	4.20±0.11	4.11	19.45±0.33	19.01

TABLE 4: Performance comparison of decorrelated BN methods on CIFAR10/CIFAR100 [75] based on ResNet-50 [74]. We report each result based on 10 runs. The best four results are highlighted in **red**, **blue**, **green**, and **cyan** respectively.

Table 4 compares the performance of each method on CIFAR10/CIFAR100 [75] based on ResNet-50 [74]. Both of our NOG and OLR achieve better performance than other weight treatments and the SVD. Moreover, when hybrid treatments are adopted, we can observe step-wise

steady improvements on the validation errors. Among these techniques, the joint usage of OLR with NOG and OW achieves the best performances across metrics and datasets, outperforming the SVD baseline by 0.4% on CIFAR10 and by 0.9% on CIFAR100. This demonstrates that these treatments are complementary and can benefit each other.

Methods	DenseNet-121 [76]		MobileNet-v2 [77]	
	mean±std	min	mean±std	min
SVD	27.37±0.54	26.88	34.35±0.32	34.00
SVD + SN	27.05±0.44	26.51	34.19±0.37	33.82
SVD + OL	27.41±0.35	26.99	34.58±0.43	34.15
SVD + OW	27.25±0.47	26.67	34.27±0.46	33.77
SVD + NOG	**25.14±0.39**	**24.65**	**33.42±0.41**	**32.91**
SVD + OLR	**25.34±0.28**	**25.01**	**33.59±0.64**	**32.84**
SVD + NOG + OW	**24.49±0.43**	**23.97**	**33.13±0.55**	**32.61**
SVD + NOG + OW + OLR	**23.74±0.24**	**23.41**	**32.83±0.48**	**32.33**
NS iteration	25.87±0.43	25.31	33.67±0.51	33.24

TABLE 5: Performance comparison of decorrelated BN methods on CIFAR100 [75] with DenseNet-121 [76] and MobileNet-v2 [77] based on 10 runs. The best four results are highlighted in **red**, **blue**, **green**, and **cyan** respectively.

Table 5 presents the validation errors on CIFAR100 with DenseNet-121 [76] and MobileNet-v2 [77]. The results are coherent with those on ResNet-50 [74]: our methods bring consistent performance improvements to the ordinary SVD on different architectures. This demonstrates the model-agnostic property of the proposed orthogonality approaches. Fig. 7 displays the corresponding best validation accuracy during the training process. Our method can also accelerate the convergence of the training process. The acceleration is particularly significant in the initial training stage.

Fig. 7: The best validation accuracy during the training process. Our proposed techniques can consistently improve the convergence speed and help the model to achieve better accuracy within fewer training epochs.

Finally, we would like to note that the performance gain of our methods depends on the specific architectures and the ill-conditioned extent of the covariance. Generally speaking, the larger the model is, the worse-conditioned the covariance is and the larger the performance gain would be. Take the above decorrelated BN experiments as an example, the accuracy improvement on MobileNet is around 1.5%, while the performance gain on larger DenseNet is about 4.0%.

6.1.2 Global Covariance Pooling

We use ResNet-18 [74] for the GCP experiment and train it from scratch on ImageNet [78]. Fig. 8 displays the overview of a GCP model. For the ResNet backbone, the last Global

Fig. 8: The architecture of a GCP model [1], [2]. After all the convolution layers, the covariance square root of the feature is computed and used as the final representation.

Average Pooling (GAP) layer is replaced with our GCP layer. Consider the final batched convolutional feature $\mathbf{X}\in\mathbb{R}^{B\times C\times HW}$. We compute the matrix square root of its covariance as:

$$\mathbf{Q} = (\mathbf{X}\mathbf{X}^T)^{\frac{1}{2}} \tag{20}$$

where $\mathbf{Q}\in\mathbb{R}^{B\times C\times C}$ is used as the final representation and directly passed to the fully-connected (FC) layer.

Method	Failure Times	Top-1 Acc. (%)	Top-5 Acc. (%)
SVD	5	73.13	91.02
SVD + SN	2	73.28 (↑ 0.2)	91.11 (↑ 0.1)
SVD + OL	1	71.75 (↓ 1.4)	90.20 (↓ 0.8)
SVD + OW	2	73.07 (↓ 0.1)	90.93 (↓ 0.1)
SVD + NOG	1	**73.51 (↑ 0.4)**	**91.35 (↑ 0.3)**
SVD + OLR	0	**73.39 (↑ 0.3)**	**91.26 (↑ 0.2)**
SVD + NOG + OW	0	**73.71 (↑ 0.6)**	**91.43 (↑ 0.4)**
SVD + NOG + OW + OLR	0	**73.82 (↑ 0.7)**	**91.57 (↑ 0.6)**
NS iteration	0	73.36 (↑ 0.2)	90.96 (↓ 0.1)

TABLE 6: Performance comparison of different GCP methods on ImageNet [78] based on ResNet-18 [74]. The failure times denote the total times of non-convergence of the SVD solver during one training process. The best four results are highlighted in **red**, **blue**, **green**, and **cyan** respectively.

Table 6 presents the total failure times of the SVD solver in one training process and the validation accuracy on ImageNet [78] based on ResNet-18 [74]. The results are very coherent with our experiment of decorrelated BN. Among the weight treatments, the OL and OW hurt the performance, while the SN improves that of SVD by 0.2%. Our proposed NOG and OLR outperform the weight treatments and improve the SVD baseline by 0.4% and by 0.3%, respectively. Moreover, the combinations with the orthogonal weight further boost the performance. Specifically, combining NOG and OW surpasses the SVD by 0.6%. The joint use of OW with NOG and OLR achieves the best performance among all the methods and beats the SVD by 0.7%.

Fig. 10 depicts the covariance conditioning in the later training stage. Our OLR and the OW both reduce the condition number by around $1e15$, whereas the proposed NOG improves the condition number by $2e15$. When hybrid treatments are used, combining NOG and OW attains better conditioning than the separate usages. Furthermore, simultaneously using all the techniques leads to the best conditioning and improves the condition number by $5e15$.

The covariance conditioning of GCP tasks is not improved as much as that of decorrelated BN. This might stem from the unique architecture of GCP models: the covariance is directly used as the final representation and fed to the FC layer. We conjecture that this setup might cause the covariance

Fig. 9: Latent traversal on AnimeFace [79]. The EigenGAN has entangled attributes in the identified interpretable directions, while our methods achieve better disentanglement and each direction corresponds to a unique attribute.

Fig. 10: The covariance conditioning of GCP methods in the later stage of the training. The periodic spikes are caused by the evaluation on the validation set after every epoch.

to have a high condition number. The approximate solver (NS iteration) does not have well-conditioned matrices either ($\approx 1e15$), which partly supports our conjecture.

6.1.3 Computational Cost

Methods	FP (ms)	BP (ms)
SVD	44	95
SVD + NOG	44	97 (+2)
SVD + OLR	44	96 (+1)
SVD + OW	48 (+4)	102 (+7)
SVD + OW + NOG + OLR	49 (+5)	106 (+11)
NS Iteration	43	93
Vanilla ResNet-50	42	90

TABLE 7: Time consumption of each forward pass (FP) and backward pass (BP) measured on a RTX A6000 GPU. The evaluation is based on ResNet-50 and CIFAR100.

Table 7 compares the time consumption of a single training step for the experiment of decorrelated BN. Our NOG and OLR bring negligible computational costs to the BP (2% and 1%), while the FP is not influenced. Even when all techniques are applied, the overall time costs are marginally increased by 10%. Notice that NOG and OLR have no impact on the inference speed.

6.2 Latent Disentanglement

In this subsection, we first introduce the experiment setup, followed by the evaluation results on different GAN archi-

tectures and datasets. We defer the implementation details to the Supplementary Material.

6.2.1 Experimental Setup

Models. We evaluate our methods on EigenGAN [67] and vanilla GAN [21]. EigenGAN [67] is a particular GAN architecture dedicated to latent disentanglement. It progressively injects orthogonal subspaces into each layer of the generator, which can mine controllable semantic attributes in an unsupervised manner. For the vanilla GAN [21], we adopt the basic GAN model that consists of stacked convolutional layers and do not make any architectural modifications.

Datasets. For EigenGAN, we use AnimeFace [79] and FFHQ [80] datasets. AnimeFace [79] is comprised of $63,632$ aligned anime faces with resolution varying from $90{\times}90$ to $120{\times}120$. FFHQ [80] consists of $70,000$ high-quality face images that have considerable variations in identifies and have good coverage in common accessories. Since the vanilla GAN has a smaller architecture and fewer parameters, we use relatively simpler CelebA [81] and LSUN Church [82] datasets. CelebA [81] contains $202,599$ face images of $10,177$ celebrities, while LSUN Church [82] has $126,227$ scenes images of church.

Metrics. We use Frechet Inception Distance (FID) [83] to quantitatively evaluate the quality of generate images. For the performance of latent disentanglement, we use Variational Predictability (VP) [66] as the quantitative metric. The VP metric adopts the few-shot learning setting to measure the generalization abilities of a simple neural network in classifying the discovered latent directions.

Baselines. For the EigenGAN model that already has inherent orthogonality constraints and good disentanglement abilities, we compare the ordinary EignGAN with the modified version augmented by our proposed orthogonal techniques (NOG and OLR). For the vanilla GAN that suffers from limited disentanglement, we compare our NOG and OLR against other disentanglement schemes used in GANs, including (1) Hessian Penalty (HP) [65], (2) Orthogonal Jacobian Regularization (OrthoJar) [68], and (3) Latent Variational Predictability (LVP) [66].

6.2.2 EigenGAN Architecture and Modifications

Fig. 12 displays the overview of the EigenGAN. At each layer, the latent code $\mathbf{z}_i$ is multiplied with the orthogonal basis $\mathbf{U}_i$ and the diagonal importance matrix $\mathbf{L}_i$ to inject weighted orthogonal subspace for disentangled representation learning. The original EigenGAN [67] adopts the OL loss $||\mathbf{U}_i\mathbf{U}_i^T - \mathbf{I}||_F$ to enforce *relaxed* orthogonality to each

Fig. 11: Visualization of some fine-grained attributes learned by out method on FFHQ [80] dataset. Our method can learn very subtle and fine-grained attributes while keeping the identity unchanged.

subspace $\mathbf{U}_i$. Instead, we apply our NOG and OLR to achieve the weight and gradient orthogonality, respectively. Notice that when our NOG and OLR are applied, we do not use the OL loss of EigenGAN. This is because the *soft* orthogonality introduced by the OL loss might not be enforced under the gradient manipulation of our NOG, which is similar to our experimental results of decorrelated BN (see Sec. 4.1.3).

Fig. 12: Overview of the EigenGAN architecture.

6.2.3 Results on EigenGAN

Fig. 13: Subtle semantic attributes mined by our method.

Qualitative Evaluation. Fig. 9 compares the latent traversal results of the ordinary EigenGAN and our methods on AnimeFace. The interpretable direction of EigenGAN has many entangled attributes; the identity is poorly preserved during the latent traversal. By contrast, moving along with the discovered direction of our method would only introduce changes of a single semantic attribute. This demonstrates that our interpretable directions have more precisely-controlled semantics and our orthogonality techniques indeed help the model to learn more disentangled representations. Moreover, thanks to the power of orthogonality, our methods can mine many subtle and fine-grained attributes. Fig. 13 displays such attributes that are precisely captured by out method but are not learnt by EigenGAN. These attributes include very subtle local details of the image, such as facial blush, facial shadow, and mouth openness.

Fig. 14: Qualitative comparison on FFHQ. The attributes are entangled in one latent direction of EigenGAN, while our method can avoid this and discover orthogonal concepts.

Fig. 14 compares the exemplary latent traversal on FFHQ. Similar with the result on AnimeFace, the interpretable directions have more disentangled attributes when our orthogonality techniques are used. Since FFHQ covers a wide range of image attributes, our method is able to learn very fine-grained attributes (*e.g.*, angle and thickness of eyebrow) of a given super attribute (*e.g.*, eyebrow) accordingly. We give a few examples in Fig. 11. As can be observed, our method can precisely control the subtle detail of the image while keeping other attributes unchanged.

Methods	AnimeFace [79]		FFHQ [80]	
	FID (↓)	VP (↑)	FID (↓)	VP (↑)
EigenGAN	23.59	37.01	36.81	31.79
EigenGAN+NOG	19.48	43.53	33.34	37.27
EigenGAN+OLR	18.30	43.99	31.42	37.23
EigenGAN+OLR+NOG	**16.31**	**45.48**	**30.06**	**39.32**

TABLE 8: Quantitative evaluation on EigenGAN.

Quantitative Evaluation. Table 8 compares the performance of EigenGAN on AnimeFace and FFHQ datasets. Our proposed NOG and OLR can improve both the image quality score (FID) and the disentanglement score (VP). Furthermore,

when these two techniques are combined, the evaluation results achieve the best performance across metrics and datasets. This implies that enforcing simultaneous gradient and weight orthogonality allows for the learning of more disentangled representations and improved image fidelity.

Discussion. Both quantitative and qualitative evaluation on two datasets demonstrates that our orthogonality approaches lead to better latent disentanglement than the inherent orthogonality loss of EigenGAN. This behavior is coherent to our previous experiment of decorrelated BN: the proposed NOG and OLR also outperform OL in that case. This further confirms the general applicability of our orthogonal methods.

6.2.4 Vanilla GAN Architecture

For the vanilla GAN model, we use simple convolutional layers as building blocks. The orthogonality techniques are applied on the first convolution layer after the latent code.

6.2.5 Results on Vanilla GAN

Fig. 15: Qualitative comparison on CelebA. For HP [65], The latent traversal in one direction would introduce many attributes changes. By contrast, the image identity of our method is well preserved and only the target attribute varies.

Qualitative Evaluation. Fig. 15 presents the qualitative evaluation results on CelebA [81] against HP [65]. The semantic factors discovered by our methods controls traversal process more precisely; only a single attribute is changed when one latent code is modified. By contrast, a interpretable direction mined by HP [65] would correspond to multiple attributes

sometimes. This implies that the learned representations and attributes of our NOG and OLR are more disentangled. Fig. 16 displays some learned attributes of our methods. The complex scenes and structures of churches are preserved well, and each semantic factor precisely controls the image attribute. This also demonstrates the diverse application domains of our disentanglement method beyond face analysis.

Fig. 16: Latent traversal of our NOG on LSUN Church.

Methods	Time (ms)	CelebA [81]		LSUN Church [82]	
		FID ($\downarrow$)	VP ($\uparrow$)	FID ($\downarrow$)	VP ($\uparrow$)
OrJar [68]	23	32.43	24.24	38.96	11.62
HP [65]	30	31.65	24.67	39.20	13.73
LVP [66]	16	34.36	23.49	41.24	12.58
NOG	8	29.69	25.33	37.22	13.43
OLR	8	33.29	27.22	37.83	14.50
NOG+OLR	9	30.65	28.74	35.20	16.98

TABLE 9: Quantitative evaluation on vanilla GAN. We measure the time consumption of a single forward pass and backward pass. The best three results are highlighted in **red**, **blue**, and **green** respectively.

Quantitative Evaluation. Table. 9 reports the quantitative evaluation results on vanilla GAN. Our proposed orthogonality techniques outperform other disentanglement schemes in terms of both FID and VP, achieving state-of-the-art performance in the unsupervised latent disentanglement. Moreover, our approaches are much more efficient than other baselines due to the marginal computational cost.

Datasets	OrJar [68]	HP [65]	LVP [66]	NOG	OLR	NOG+OLR
CelebA [81]	2.75	2.67	2.78	2.28	2.14	**2.01**
Church [82]	2.48	2.57	2.66	2.13	2.09	**1.93**

TABLE 10: Condition number of the first convolution weight in vanilla GANs on CelebA [81] and LSUN Church [82].

Condition Number in Vanilla GANs. Similar to our previous experiments, we measure the condition number of the fist convolution weight in vanilla GANs (*i.e.,* the projection matrix that maps latent codes to features). Table 10 presents the evaluation results on CelebA [81] and LSUN Church [82]. As can be observed, our methods (NOG, OLR, and NOG+OLR) outperform other baselines and have much better condition numbers. This demonstrates that our methods can also improve the conditioning of the weight matrix of vanilla GANs. Notice that the convolution weight matrix is small in

dimensionality. The corresponding condition number is thus much smaller compared with the covariance conditioning in the previous experiments.

7 CONCLUSION

In this paper, we explore different approaches to improve the covariance conditioning of the SVD meta-layer. Existing treatments on orthogonal weight are first studied. Our experiments reveal that these techniques could improve the conditioning but might hurt the performance due to the limitation on the representation power. To avoid the side effect of orthogonal weight, we propose the nearest orthogonal gradient and the optimal learning rate, both of which could simultaneously attain better covariance conditioning and improved generalization abilities. Moreover, their combinations with orthogonal weight further boost the performance. Besides the usage on the SVD meta-layer, we show that our proposed orthogonality approaches can benefit generative models for better latent disentanglement.

ACKNOWLEDGEMENTS

REFERENCES

[1] P. Li, J. Xie, Q. Wang, and W. Zuo, "Is second-order information helpful for large-scale visual recognition?" in *ICCV*, 2017.
[2] Y. Song, N. Sebe, and W. Wang, "Why approximate matrix square root outperforms accurate svd in global covariance pooling?" in *ICCV*, 2021.
[3] Z. Gao, Q. Wang, B. Zhang, Q. Hu, and P. Li, "Temporal-attentive covariance pooling networks for video recognition," in *NeurIPS*, 2021.
[4] L. Huang, D. Yang, B. Lang, and J. Deng, "Decorrelated batch normalization," in *CVPR*, 2018.
[5] L. Huang, Y. Zhou, L. Liu, F. Zhu, and L. Shao, "Group whitening: Balancing learning efficiency and representational capacity," in *CVPR*, 2021.
[6] Y. Song, N. Sebe, and W. Wang, "Fast differentiable matrix square root," in *ICLR*, 2022.
[7] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang, "Universal style transfer via feature transforms," in *NeurIPS*, 2017.
[8] T.-Y. Chiu, "Understanding generalized whitening and coloring transform for universal style transfer," in *ICCV*, 2019.
[9] Z. Wang, L. Zhao, H. Chen, L. Qiu, Q. Mo, S. Lin, W. Xing, and D. Lu, "Diversified arbitrary style transfer via deep feature perturbation," in *CVPR*, 2020.
[10] E. Brachmann, A. Krull, S. Nowozin, J. Shotton, F. Michel, S. Gumhold, and C. Rother, "Dsac-differentiable ransac for camera localization," in *CVPR*, 2017.
[11] D. Campbell, L. Liu, and S. Gould, "Solving the blind perspective-n-point problem end-to-end with robust differentiable geometric optimization," in *ECCV*, 2020.
[12] Z. Dang, K. M. Yi, Y. Hu, F. Wang, P. Fua, and M. Salzmann, "Eigendecomposition-free training of deep networks for linear least-square problems," *TPAMI*, 2020.
[13] N. J. Higham, *Functions of matrices: theory and computation*. SIAM, 2008.
[14] W. Wang, Z. Dang, Y. Hu, P. Fua, and M. Salzmann, "Robust differentiable svd," *TPAMI*, 2021.
[15] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 9–48.
[16] S. Wiesler and H. Ney, "A convergence analysis of log-linear training," *NeurIPS*, 2011.
[17] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *ICML*, 2013.
[18] D. Mishkin and J. Matas, "All you need is a good init," *ICLR*, 2016.
[19] J. Wang, Y. Chen, R. Chakraborty, and S. X. Yu, "Orthogonal convolutional neural networks," in *CVPR*, 2020.
[20] S. Singla and S. Feizi, "Skew orthogonal convolutions," in *ICML*, 2021.
[21] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *NeurIPS*, 2014.
[22] J. Zhu, R. Feng, Y. Shen, D. Zhao, Z.-J. Zha, J. Zhou, and Q. Chen, "Low-rank subspaces in gans," *NeurIPS*, 2021.
[23] Y. Shen and B. Zhou, "Closed-form factorization of latent semantics in gans," in *CVPR*, 2021.
[24] Y. Song, N. Sebe, and W. Wang, "Improving covariance conditioning of the svd meta-layer by orthogonality," in *ECCV*, 2022.
[25] C. Ionescu, O. Vantzos, and C. Sminchisescu, "Matrix backpropagation for deep networks with structured layers," in *ICCV*, 2015.
[26] ——, "Training deep networks with structured layers by matrix backpropagation," *arXiv preprint arXiv:1509.07838*, 2015.
[27] Y. Song, N. Sebe, and W. Wang, "Fast differentiable matrix square root and inverse square root," *TPAMI*, 2022.
[28] ——, "Batch-efficient eigendecomposition for small and medium matrices," in *ECCV*, 2022.
[29] L. Huang, Y. Zhou, F. Zhu, L. Liu, and L. Shao, "Iterative normalization: Beyond standardization towards efficient whitening," in *CVPR*, 2019.
[30] L. Huang, L. Zhao, Y. Zhou, F. Zhu, L. Liu, and L. Shao, "An investigation into the stochasticity of batch whitening," in *CVPR*, 2020.
[31] P. Li, J. Xie, Q. Wang, and Z. Gao, "Towards faster training of global covariance pooling networks by iterative matrix square root normalization," in *CVPR*, 2018.
[32] Q. Wang, J. Xie, W. Zuo, L. Zhang, and P. Li, "Deep cnns meet global covariance pooling: Better representation and generalization," *TPAMI*, 2020.
[33] J. Xie, R. Zeng, Q. Wang, Z. Zhou, and P. Li, "So-vit: Mind visual tokens for vision transformer," *arXiv preprint arXiv:2104.10935*, 2021.
[34] Y. Song, N. Sebe, and W. Wang, "On the eigenvalues of global covariance pooling for fine-grained visual recognition," *IEEE TPAMI*, 2022.
[35] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis," in *CVPR*, 2017.
[36] W. Cho, S. Choi, D. K. Park, I. Shin, and J. Choo, "Image-to-image translation via group-wise deep whitening-and-coloring transformation," in *CVPR*, 2019.
[37] A. Abramov, C. Bayer, and C. Heller, "Keep it simple: Image statistics matching for domain adaptation," *arXiv preprint arXiv:2005.12551*, 2020.
[38] S. Choi, S. Jung, H. Yun, J. T. Kim, S. Kim, and J. Choo, "Robustnet: Improving domain generalization in urban-scene segmentation via instance selective whitening," in *CVPR*, 2021.
[39] R. Ranftl and V. Koltun, "Deep fundamental matrix estimation," in *ECCV*, 2018.
[40] I. Murray, "Differentiation of the cholesky decomposition," *arXiv preprint arXiv:1602.07527*, 2016.
[41] Z. Geng, M.-H. Guo, H. Chen, X. Li, K. Wei, and Z. Lin, "Is attention better than matrix decomposition?" in *ICLR*, 2021.
[42] Y. Xiong, Z. Zeng, R. Chakraborty, M. Tan, G. Fung, Y. Li, and V. Singh, "Nyströmformer: A nyström-based algorithm for approximating self-attention," in *AAAI*, 2021.
[43] J. Lu, J. Yao, J. Zhang, X. Zhu, H. Xu, W. Gao, C. Xu, T. Xiang, and L. Zhang, "Soft: softmax-free transformer with linear complexity," *NeurIPS*, 2021.
[44] T.-H. Chan, K. Jia, S. Gao, J. Lu, Z. Zeng, and Y. Ma, "Pcanet: A simple deep learning baseline for image classification?" *IEEE TIP*, 2015.
[45] Y. Yang, J. Sun, H. Li, and Z. Xu, "Admm-net: A deep learning approach for compressive sensing mri," *arXiv preprint arXiv:1705.06869*, 2017.
[46] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, 1994.
[47] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *AISTATS*, 2010.
[48] A. M. Saxe, J. L. McClelland, and S. Ganguli, "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks," in *ICLR*, 2014.

[49] L. Xiao, Y. Bahri, J. Sohl-Dickstein, S. Schoenholz, and J. Pennington, "Dynamical isometry and a mean field theory of cnns: How to train 10,000-layer vanilla convolutional neural networks," in *ICML*. PMLR, 2018.

[50] P. Rodríguez, J. Gonzalez, G. Cucurull, J. M. Gonfaus, and X. Roca, "Regularizing cnns with locally constrained decorrelations," in *ICLR*, 2016.

[51] N. Bansal, X. Chen, and Z. Wang, "Can we gain more from orthogonality regularizations in training deep networks?" in *NeurIPS*, 2018.

[52] H. Qi, C. You, X. Wang, Y. Ma, and J. Malik, "Deep isometric learning for visual recognition," in *ICML*. PMLR, 2020.

[53] K. D. Maduranga, K. E. Helfrich, and Q. Ye, "Complex unitary recurrent neural networks using scaled cayley transform," in *AAAI*, 2019.

[54] A. Trockman and J. Z. Kolter, "Orthogonalizing convolutional layers with the cayley transform," in *ICLR*, 2020.

[55] A. Brock, J. Donahue, and K. Simonyan, "Large scale gan training for high fidelity natural image synthesis," in *ICLR*, 2019.

[56] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," in *ICLR*, 2018.

[57] Y. Tsuzuku, I. Sato, and M. Sugiyama, "Lipschitz-margin training: Scalable certification of perturbation invariance for deep neural networks," *NeurIPS*, 2018.

[58] H. Sedghi, V. Gupta, and P. M. Long, "The singular values of convolutional layers," in *ICLR*, 2018.

[59] D. Bau, J.-Y. Zhu, H. Strobelt, B. Zhou, J. B. Tenenbaum, W. T. Freeman, and A. Torralba, "Gan dissection: Visualizing and understanding generative adversarial networks," in *ICLR*, 2019.

[60] A. Jahanian, L. Chai, and P. Isola, "On the" steerability" of generative adversarial networks," in *ICLR*, 2020.

[61] A. Voynov and A. Babenko, "Unsupervised discovery of interpretable directions in the gan latent space," in *ICML*. PMLR, 2020.

[62] C. Tzelepis, G. Tzimiropoulos, and I. Patras, "Warpedganspace: Finding non-linear rbf paths in gan latent space," in *ICCV*, 2021.

[63] Y. Shen, J. Gu, X. Tang, and B. Zhou, "Interpreting the latent space of gans for semantic face editing," in *CVPR*, 2020.

[64] E. Härkönen, A. Hertzmann, J. Lehtinen, and S. Paris, "Ganspace: Discovering interpretable gan controls," *NeurIPS*, 2020.

[65] W. Peebles, J. Peebles, J.-Y. Zhu, A. Efros, and A. Torralba, "The hessian penalty: A weak prior for unsupervised disentanglement," in *ECCV*. Springer, 2020.

[66] X. Zhu, C. Xu, and D. Tao, "Learning disentangled representations with latent variation predictability," in *ECCV*. Springer, 2020.

[67] Z. He, M. Kan, and S. Shan, "Eigengan: Layer-wise eigen-learning for gans," in *CVPR*, 2021.

[68] Y. Wei, Y. Shi, X. Liu, Z. Ji, Y. Gao, Z. Wu, and W. Zuo, "Orthogonal jacobian regularization for unsupervised disentanglement in image generation," in *ICCV*, 2021.

[69] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *ICML*, 2013.

[70] N. J. Higham, *Matrix nearness problems and applications*. Citeseer, 1988.

[71] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *ICLR*, 2016.

[72] J. Zhu, Y. Shen, Y. Xu, D. Zhao, and Q. Chen, "Region-based semantic factorization in gans," *ICML*, 2022.

[73] G. Balakrishnan, R. Gadde, A. Martinez, and P. Perona, "Rayleigh eigendirections (reds): Gan latent space traversals for multidimensional features," *arXiv preprint arXiv:2201.10423*, 2022.

[74] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[75] A. Krizhevsky, "Learning multiple layers of features from tiny images," *Master's thesis, University of Tront*, 2009.

[76] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *CVPR*, 2017.

[77] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[78] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*, 2009.

[79] B. Chao. (2019) Anime face dataset: a collection of high-quality anime faces. [Online]. Available: https://github.com/bchao1/Anime-Face-Dataset

[80] V. Kazemi and J. Sullivan, "One millisecond face alignment with an ensemble of regression trees," in *CVPR*, 2014.

[81] Z. Liu, P. Luo, X. Wang, and X. Tang. Large-scale celebfaces attributes (CelebA) dataset. [Online]. Available: https://mmlab.ie.cuhk.edu.hk/projects/CelebA.html

[82] F. Yu, A. Seff, Y. Zhang, S. Song, T. Funkhouser, and J. Xiao, "Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop," *arXiv preprint arXiv:1506.03365*, 2015.

[83] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," *NeurIPS*, 2017.

[84] A. Siarohin, E. Sangineto, and N. Sebe, "Whitening and coloring batch transform for gans," in *ICLR*, 2018.

[85] L. Mirsky, "A trace inequality of john von neumann," *Monatshefte für mathematik*, vol. 79, no. 4, pp. 303–306, 1975.

[86] R. D. Grigorieff, "A note on von neumann's trace inequality," *Mathematische Nachrichten*, vol. 151, no. 1, pp. 327–328, 1991.

Yue Song received the B.Sc. *cum laude* from KU Leuven, Belgium and the joint M.Sc. *summa cum laude* from the University of Trento, Italy and KTH Royal Institute of Technology, Sweden. Currently, he is a Ph.D. student with the Multimedia and Human Understanding Group (MHUG) at the University of Trento, Italy. His research interests are computer vision, deep learning, and numerical analysis and optimization.

Nicu Sebe is Professor with the University of Trento, Italy, leading the research in the areas of multimedia information retrieval and human behavior understanding. He was the General Co- Chair of ACM Multimedia 2013, and the Program Chair of ACM Multimedia 2007 and 2011, ECCV 2016, ICCV 2017 and ICPR 2020. He is a fellow of the International Association for Pattern Recognition.

Wei Wang is an Assistant Professor of Computer Science at University of Trento, Italy. Previously, after obtaining his PhD from University of Trento in 2018, he became a Postdoc at EPFL, Switzerland. His research interests include machine learning and its application to computer vision and multimedia analysis.

APPENDIX A
BACKGROUND: SVD META-LAYER

This subsection presents the background knowledge about the propagation rules of the SVD meta-layer.

A.1 Forward Pass

Given the reshape feature $\mathbf{X} \in \mathbb{R}^{d \times N}$ where d denotes the feature dimensionality (*i.e.*, the number of channels) and N represents the number of features (*i.e.*, the product of spatial dimensions of features), an SVD meta-layer first computes the sample covariance as:

$$\mathbf{P} = \mathbf{X}\mathbf{J}\mathbf{X}^T, \mathbf{J} = \frac{1}{N}(\mathbf{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^T) \tag{21}$$

where $\mathbf{J}$ represents the centering matrix, $\mathbf{I}$ denotes the identity matrix, and $\mathbf{1}$ is a column vector whose values are all ones, respectively. The covariance is always positive semi-definite (PSD) and does not have any negative eigenvalues. Afterward, the eigendecomposition is performed using the SVD:

$$\mathbf{P} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T, \ \mathbf{\Lambda} = \text{diag}(\lambda_1, \dots, \lambda_\text{d}) \tag{22}$$

where $\mathbf{U}$ is the orthogonal eigenvector matrix, $\text{diag}(\cdot)$ denotes transforming a vector to a diagonal matrix, and $\mathbf{\Lambda}$ is the diagonal matrix in which the eigenvalues are sorted in a non-increasing order *i.e.*, $\lambda_i \geq \lambda_{i+1}$. Then depending on the application, the matrix square root or the inverse square root is calculated as:

$$\mathbf{Q} \triangleq \mathbf{P}^{\frac{1}{2}} = \mathbf{U}\mathbf{\Lambda}^{\frac{1}{2}}\mathbf{U}^T, \mathbf{\Lambda}^{\frac{1}{2}} = \text{diag}(\lambda_1^{\frac{1}{2}}, \dots, \lambda_d^{\frac{1}{2}})$$
$$\mathbf{S} \triangleq \mathbf{P}^{-\frac{1}{2}} = \mathbf{U}\mathbf{\Lambda}^{-\frac{1}{2}}\mathbf{U}^T, \mathbf{\Lambda}^{-\frac{1}{2}} = \text{diag}(\lambda_1^{-\frac{1}{2}}, \dots, \lambda_d^{-\frac{1}{2}}) \tag{23}$$

The matrix square root $\mathbf{Q}$ is often used in GCP-related tasks [1], [2], [33], while the application of decorrelated BN [4], [84] widely applies the inverse square root $\mathbf{S}$. In certain applications such as WCT, both $\mathbf{Q}$ and $\mathbf{S}$ are required.

A.2 Backward Pass

Let $\frac{\partial l}{\partial \mathbf{Q}}$ and $\frac{\partial l}{\partial \mathbf{S}}$ denote the partial derivative of the loss l w.r.t to the matrix square root $\mathbf{Q}$ and the inverse square root $\mathbf{S}$, respectively. Then the gradient passed to the eigenvector is computed as:

$$\frac{\partial l}{\partial \mathbf{U}}\Big|_\mathbf{Q} = (\frac{\partial l}{\partial \mathbf{Q}} + (\frac{\partial l}{\partial \mathbf{Q}})^T)\mathbf{U}\mathbf{\Lambda}^{\frac{1}{2}}, \ \frac{\partial l}{\partial \mathbf{U}}\Big|_\mathbf{S} = (\frac{\partial l}{\partial \mathbf{S}} + (\frac{\partial l}{\partial \mathbf{S}})^T)\mathbf{U}\mathbf{\Lambda}^{-\frac{1}{2}} \tag{24}$$

Notice that the gradient equations for $\mathbf{Q}$ and $\mathbf{S}$ are different. For the eigenvalue, the gradient is calculated as:

$$\frac{\partial l}{\partial \mathbf{\Lambda}}\Big|_\mathbf{Q} = \frac{1}{2}\text{diag}(\lambda_1^{-\frac{1}{2}}, \dots, \lambda_\text{d}^{-\frac{1}{2}})\mathbf{U}^\text{T}\frac{\partial l}{\partial \mathbf{Q}}\mathbf{U},$$
$$\frac{\partial l}{\partial \mathbf{\Lambda}}\Big|_\mathbf{S} = -\frac{1}{2}\text{diag}(\lambda_1^{-\frac{3}{2}}, \dots, \lambda_\text{d}^{-\frac{3}{2}})\mathbf{U}^\text{T}\frac{\partial l}{\partial \mathbf{S}}\mathbf{U} \tag{25}$$

Subsequently, the derivative of the SVD step can be calculated as:

$$\frac{\partial l}{\partial \mathbf{P}} = \mathbf{U}((\mathbf{K}^T \circ (\mathbf{U}^T\frac{\partial l}{\partial \mathbf{U}})) + (\frac{\partial l}{\partial \mathbf{\Lambda}})_\text{diag})\mathbf{U}^T \tag{26}$$

where $\circ$ denotes the matrix Hadamard product, and the matrix $\mathbf{K}$ consists of entries $K_{ij} = 1/(\lambda_i - \lambda_j)$ if $i \neq j$ and

$K_{ij} = 0$ otherwise. This step is the same for both $\mathbf{Q}$ and $\mathbf{S}$. Finally, we have the gradient passed to the feature $\mathbf{X}$ as:

$$\frac{\partial l}{\partial \mathbf{X}} = (\frac{\partial l}{\partial \mathbf{P}} + (\frac{\partial l}{\partial \mathbf{P}})^T)\mathbf{X}\mathbf{J} \tag{27}$$

With the above rules, the SVD function can be easily inserted into any neural networks and trained end-to-end as a meta-layer.

APPENDIX B
MATHEMATICAL DERIVATION AND PROOF

B.1 Derivation of Nearest Orthogonal Gradient

The problem of finding the nearest orthogonal gradient can be defined as:

$$\min_\mathbf{R} ||\frac{\partial l}{\partial \mathbf{W}} - \mathbf{R}||_\text{F} \ subject \ to \ \mathbf{R}\mathbf{R}^T = \mathbf{I} \tag{28}$$

To solve this constrained optimization problem, We can construct the following error function:

$$e(\mathbf{R}) = Tr\Big((\frac{\partial l}{\partial \mathbf{W}} - \mathbf{R})^T(\frac{\partial l}{\partial \mathbf{W}} - \mathbf{R})\Big) + Tr\Big(\mathbf{\Sigma}\mathbf{R}^T\mathbf{R} - \mathbf{I}\Big) \tag{29}$$

where $Tr(\cdot)$ is the trace measure, and $\mathbf{\Sigma}$ denotes the symmetric matrix Lagrange multiplier. Setting the derivative to zero leads to:

$$\frac{de(\mathbf{R})}{d\mathbf{R}} = -2(\frac{\partial l}{\partial \mathbf{W}} - \mathbf{R}) + 2\mathbf{R}\mathbf{\Sigma} = 0$$
$$\frac{\partial l}{\partial \mathbf{W}} = \mathbf{R}(\mathbf{I} + \mathbf{\Sigma}), \ \mathbf{R} = \frac{\partial l}{\partial \mathbf{W}}(\mathbf{I} + \mathbf{\Sigma})^{-1} \tag{30}$$

The term $(\mathbf{I} + \mathbf{\Sigma})$ can be represented using $\frac{\partial l}{\partial \mathbf{W}}$. Consider the covariance of $\frac{\partial l}{\partial \mathbf{W}}$:

$$(\frac{\partial l}{\partial \mathbf{W}})^T\frac{\partial l}{\partial \mathbf{W}} = (\mathbf{I} + \mathbf{\Sigma})^T\mathbf{R}^T\mathbf{R}(\mathbf{I} + \mathbf{\Sigma}) = (\mathbf{I} + \mathbf{\Sigma})^T(\mathbf{I} + \mathbf{\Sigma})$$
$$(\mathbf{I} + \mathbf{\Sigma}) = \Big((\frac{\partial l}{\partial \mathbf{W}})^T\frac{\partial l}{\partial \mathbf{W}}\Big)^{\frac{1}{2}} \tag{31}$$

Substituting the term $(\mathbf{I} + \mathbf{\Sigma})$ in eq. (30) with the above equation leads to the closed-form solution of the nearest orthogonal gradient:

$$\mathbf{R} = \frac{\partial l}{\partial \mathbf{W}}\Big((\frac{\partial l}{\partial \mathbf{W}})^T\frac{\partial l}{\partial \mathbf{W}}\Big)^{-\frac{1}{2}} \tag{32}$$

B.2 Derivation of Optimal Learning Rate

To jointly optimize the updated weight $\mathbf{W} - \eta\frac{\partial l}{\partial \mathbf{W}}$, we need to achieve the following objective:

$$\min_\eta ||(\mathbf{W} - \eta\frac{\partial l}{\partial \mathbf{W}})(\mathbf{W} - \eta\frac{\partial l}{\partial \mathbf{W}})^T - \mathbf{I}||_\text{F} \tag{33}$$

This optimization problem can be more easily solved in the form of vector. Let $\mathbf{w}$, $\mathbf{i}$, and $\mathbf{l}$ denote the vectorized $\mathbf{W}$, $\mathbf{I}$, and $\frac{\partial l}{\partial \mathbf{W}}$, respectively. Then we construct the error function as:

$$e(\eta) = \Big((\mathbf{w} - \eta\mathbf{l})^T(\mathbf{w} - \eta\mathbf{l}) - \mathbf{i}\Big)^T\Big((\mathbf{w} - \eta\mathbf{l})^T(\mathbf{w} - \eta\mathbf{l}) - \mathbf{i}\Big) \tag{34}$$

Expanding the equation leads to:

$$e(\eta) = (\mathbf{w}^T\mathbf{w} - 2\eta\mathbf{l}^T\mathbf{w} + \eta^2\mathbf{l}^T\mathbf{l} - \mathbf{i})^T(\mathbf{w}^T\mathbf{w} - 2\eta\mathbf{l}^T\mathbf{w} + \eta^2\mathbf{l}^T\mathbf{l} - \mathbf{i}) \tag{35}$$

Differentiating $e(\eta)$ w.r.t. η yields:

$$\frac{de(\eta)}{d\eta} = -4\mathbf{w}\mathbf{w}^T\mathbf{l}^T\mathbf{w} + 4\eta\mathbf{w}\mathbf{w}^T\mathbf{l}^T\mathbf{l}$$
$$+8\eta\mathbf{l}^T\mathbf{w}\mathbf{l}^T\mathbf{w} - 12\eta^2\mathbf{l}^T\mathbf{w}\mathbf{l}^T\mathbf{l} + 4\mathbf{l}\mathbf{w}^T\mathbf{i} + 4\eta^3\mathbf{l}\mathbf{l}^T - 4\eta\mathbf{i}\mathbf{l}\mathbf{l}^T \quad (36)$$

Since η is typically very small, the higher-order terms (*e.g.*, η^2 and η^3) are sufficiently small such that they can be neglected. After omitting these terms, the derivative becomes:

$$\frac{de(\eta)}{d\eta} \approx -4\mathbf{w}\mathbf{w}^T\mathbf{l}^T\mathbf{w} + 4\eta\mathbf{w}\mathbf{w}^T\mathbf{l}^T\mathbf{l} + 8\eta\mathbf{l}^T\mathbf{w}\mathbf{l}^T\mathbf{w} + 4\mathbf{l}\mathbf{w}^T\mathbf{i} - 4\eta\mathbf{i}\mathbf{l}\mathbf{l}^T \quad (37)$$

Setting the derivative to zero leads to the optimal learning rate:

$$\eta^\star \approx \frac{\mathbf{w}^T\mathbf{w}\mathbf{l}^T\mathbf{w} - \mathbf{l}^T\mathbf{w}\mathbf{i}}{\mathbf{w}^T\mathbf{w}\mathbf{l}^T\mathbf{l} + 2\mathbf{l}^T\mathbf{w}\mathbf{l}^T\mathbf{w} - \mathbf{l}^T\mathbf{l}\mathbf{i}} \quad (38)$$

Notice that $\mathbf{i}$ is the vectorization of the identify matrix $\mathbf{I}$, which means that $\mathbf{i}$ is very sparse (*i.e.*, lots of zeros) and the impact can be neglected. The optimal learning rate can be further simplified as:

$$\eta^\star \approx \frac{\mathbf{w}^T\mathbf{w}\mathbf{l}^T\mathbf{w}}{\mathbf{w}^T\mathbf{w}\mathbf{l}^T\mathbf{l} + 2\mathbf{l}^T\mathbf{w}\mathbf{l}^T\mathbf{w}} \quad (39)$$

B.3 Proof of the learning rate bounds

Proposition 1. *When both $\mathbf{W}$ and $\frac{\partial l}{\partial \mathbf{W}}$ are orthogonal, $\eta^\star$ is both upper and lower bounded. The upper bound is $\frac{N^2}{N^2+2}$ and the lower bound is $\frac{1}{N^2+2}$ where N denotes the row dimension of $\mathbf{W}$.*

Proof. Since the vector product is equivalent to the matrix Frobenius inner product, we have the relation:

$$\mathbf{l}^T\mathbf{w} = \langle \frac{\partial l}{\partial \mathbf{W}}, \mathbf{W} \rangle_F \quad (40)$$

For a given matrix pair $\mathbf{A}$ and $\mathbf{B}$, the Frobenius product $\langle \cdot \rangle_F$ is defined as:

$$\langle \mathbf{A}, \mathbf{B} \rangle_F = \sum A_{i,j} B_{i,j} \leq \sigma_1(\mathbf{A})\sigma_1(\mathbf{B}) + \cdots + \sigma_N(\mathbf{A})\sigma_N(\mathbf{B}) \quad (41)$$

where $\sigma(\cdot)_i$ represents the i-th largest eigenvalue, N denotes the matrix size, and the inequality is given by Von Neumann's trace inequality [85], [86]. The equality takes only when $\mathbf{A}$ and $\mathbf{B}$ have the same eigenvector. When both $\mathbf{W}$ and $\frac{\partial l}{\partial \mathbf{W}}$ are orthogonal, *i.e.*, their eigenvalues are all 1, we have the following relation:

$$\langle \frac{\partial l}{\partial \mathbf{W}}, \frac{\partial l}{\partial \mathbf{W}} \rangle_F = N, \ \langle \frac{\partial l}{\partial \mathbf{W}}, \mathbf{W} \rangle_F \leq N \quad (42)$$

This directly leads to:

$$\langle \frac{\partial l}{\partial \mathbf{W}}, \mathbf{W} \rangle_F \leq \langle \frac{\partial l}{\partial \mathbf{W}}, \frac{\partial l}{\partial \mathbf{W}} \rangle_F, \ \mathbf{l}^T\mathbf{w} \leq \mathbf{l}^T\mathbf{l} \quad (43)$$

Exploiting this inequality, the optimal learning rate has the relation:

$$\eta^\star \approx \frac{\mathbf{w}^T\mathbf{w}\mathbf{l}^T\mathbf{w}}{\mathbf{w}^T\mathbf{w}\mathbf{l}^T\mathbf{l} + 2\mathbf{l}^T\mathbf{w}\mathbf{l}^T\mathbf{w}} \leq \frac{\mathbf{w}^T\mathbf{w}\mathbf{l}^T\mathbf{l}}{\mathbf{w}^T\mathbf{w}\mathbf{l}^T\mathbf{l} + 2\mathbf{l}^T\mathbf{w}\mathbf{l}^T\mathbf{w}} \quad (44)$$

For $\mathbf{l}^T\mathbf{w}$, we have the inequality as:

$$\mathbf{l}^T\mathbf{w} = \langle \frac{\partial l}{\partial \mathbf{W}}, \mathbf{W} \rangle_F = \sum_{i,j} \frac{\partial l}{\partial \mathbf{W}}_{i,j} \mathbf{W}_{i,j}$$
$$\geq \sigma_{min}(\frac{\partial l}{\partial \mathbf{W}})\sigma_{min}(\mathbf{W}) = 1 \quad (45)$$

Then we have the upper bounded of $\eta^\star$ as:

$$\eta^\star \leq \frac{\mathbf{w}^T\mathbf{w}\mathbf{l}^T\mathbf{l}}{\mathbf{w}^T\mathbf{w}\mathbf{l}^T\mathbf{l} + 2\mathbf{l}^T\mathbf{w}\mathbf{l}^T\mathbf{w}}$$
$$= \frac{N^2}{N^2 + 2\mathbf{l}^T\mathbf{w}\mathbf{l}^T\mathbf{w}} < \frac{N^2}{N^2 + 2} \quad (46)$$

For the lower bound, since we also have $\mathbf{l}^T\mathbf{w} \leq \mathbf{w}^T\mathbf{w}$, $\eta^\star$ can be re-written as:

$$\eta^\star \approx \frac{\mathbf{w}^T\mathbf{w}\mathbf{l}^T\mathbf{w}}{\mathbf{w}^T\mathbf{w}\mathbf{l}^T\mathbf{l} + 2\mathbf{l}^T\mathbf{w}\mathbf{l}^T\mathbf{w}}$$
$$\geq \frac{\mathbf{l}^T\mathbf{w}\mathbf{l}^T\mathbf{w}}{\mathbf{w}^T\mathbf{w}\mathbf{l}^T\mathbf{l} + 2\mathbf{l}^T\mathbf{w}\mathbf{l}^T\mathbf{w}}$$
$$= \frac{1}{\frac{\mathbf{w}^T\mathbf{w}\mathbf{l}^T\mathbf{l}}{\mathbf{l}^T\mathbf{w}\mathbf{l}^T\mathbf{w}} + 2}$$
$$= \frac{1}{\frac{N^2}{\mathbf{l}^T\mathbf{w}\mathbf{l}^T\mathbf{w}} + 2} \quad (47)$$

Injecting eq. (45) into eq. (47) leads to the further simplification:

$$\eta^\star \approx \frac{1}{\frac{N^2}{\mathbf{l}^T\mathbf{w}\mathbf{l}^T\mathbf{w}} + 2} \geq \frac{1}{N^2 + 2} \quad (48)$$

As indicated above, the optimal learning rate $\eta^\star$ has a lower bound of $\frac{1}{N^2+2}$. $\qquad\square$

APPENDIX C
DETAILED EXPERIMENTAL SETTINGS

In this section, we introduce the implementation details and experimental settings.

C.1 Covariance Conditioning

C.1.1 Decorrelated Batch Normalization

The training lasts 350 epochs and the learning rate is initialized with 0.1. The SGD optimizer is used with momentum 0.9 and weight decay $5e{-}4$. We decrease the learning rate by 10 every 100 epochs. The batch size is set to 128. We use the technique proposed in [2] to compute the stable SVD gradient. The Pre-SVD layer in this experiment is the 3×3 convolution layer.

C.1.2 Global Covariance Pooling

The training process lasts 60 epochs and the learning rate is initialize with 0.1. We decrease the learning rate by 10 at epoch 30 and epoch 45. The SGD optimizer is used with momentum 0.9 and weight decay $1e{-}4$. The model weights are randomly initialized and the batch size is set to 256. The images are first resized to 256×256 and then randomly cropped to 224×224 before being passed to the model. The data augmentation of randomly horizontal flip is used. We use the technique proposed in [2] to compute the stable SVD gradient. The Pre-SVD layer denotes the convolution transform of the previous layer.

Fig. 17: Scheme of learning rate during the training process of decorrelated BN. For the orthogonal weight and gradient, our OLR has a much higher probability of occurrence and can enforce a stronger orthogonality constraint.

C.2 Latent Disentanglement

C.2.1 EigenGAN

The input image is resize to $128{\times}128$ for AnimeFace [79] and to $256{\times}256$ for FFHQ [80]. We set the batch size to 128, and the training process lasts $500,000$ steps. The subspace dimension of each layer is set to 6, *i.e.,* each layer has 6 interpretable directions. All the orthogonality techniques are enforced on the projection matrix $\mathbf{U}_i$ at each layer.

C.2.2 Vanilla GAN

For both CelebA [81] and LSUN Church [82], we resize the input image to the resolution of $128{\times}128$. The training lasts 200 epochs for CelebA and lasts 400 epochs for LSUN Church. We set the batch size to 128 and set the latent dimension to 30.

APPENDIX D
OCCURRENCE OF OLR

Since our proposed OLR needs manual tuning during the training, it would be interesting to investigate the probability of occurrence in different settings. Fig. 17 depicts the learning rate schemes of decorrelated BN with ordinary learning rate (*left*), OLR for non-orthogonal weight/gradient (*middle*), and OLR for orthogonal weight/gradient (*right*). As can be seen, in both settings (orthogonal and non-orthogonal weight/-gradient), our OLR occurs with a reasonable probability during the training, which enforces a *related* orthogonality constraint on the weight. When the weight and gradient are non-orthogonal, our OLR mainly occurs at the first training stage where the ordinary learning rate is relative large. For orthogonal gradient and weight, the OLR happens more frequently and consistently occurs throughout all the training stages. This meets our theoretical analysis in Prop. 1: our OLR suits simultaneously orthogonal weight and gradient.