Amortized Eigendecomposition for Neural Networks

Tianbo Li[1,*], **Zekun Shi**[1,2], **Jiaxi Zhao**[3], **Min Lin**[1]
[1]SEA AI Lab
[2] School of Computing, National University of Singapore
[3] Department of Mathematics, National University of Singapore
[*]litb@sea.com

Abstract

Performing eigendecomposition during neural network training is essential for tasks such as dimensionality reduction, network compression, image denoising, and graph learning. However, eigendecomposition is computationally expensive as it is orders of magnitude slower than other neural network operations. To address this challenge, we propose a novel approach called "amortized eigendecomposition" that relaxes the exact eigendecomposition by introducing an additional loss term called eigen loss. Our approach offers significant speed improvements by replacing the computationally expensive eigendecomposition with a more affordable QR decomposition at each iteration. Theoretical analysis guarantees that the desired eigenpair is attained as optima of the eigen loss. Empirical studies on nuclear norm regularization, latent-space principal component analysis, and graphs adversarial learning demonstrate significant improvements in training efficiency while producing nearly identical outcomes to conventional approaches. This novel methodology promises to integrate eigendecomposition efficiently into neural network training, overcoming existing computational challenges and unlocking new potential for advanced deep learning applications.

1 Introduction

Eigendecomposition is a fundamental technique in linear algebra that finds applications across numerous scientific domains ranging from quantum many-body problems to multivariate statistical analysis. In the context of deep learning, eigendecomposition also plays a crucial role in tasks such as weights normalization [8, 16, 41], dimensionality reduction [6, 44, 27, 38], network compression [17, 30], image denoising [13, 12, 14], graph adversarial learning [10, 18, 47]. By uncovering the structure of networks, eigendecomposition allows us to enforce low-rankness, ensuring generalization, robustness, and computational efficiency. Eigendecomposition is also instrumental in the spectral analysis of graphs, where it can detect community structure, which is essential in spectral graph neural networks. The ability of eigendecomposition to detect the intrinsic matrix structures and properties makes it a valuable tool in various machine learning tasks with neural networks.

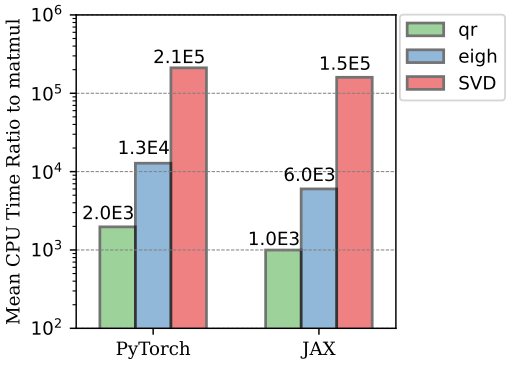


Figure 1: A comparison that illustrates the forward execution time of three linear algebra operations: `qr`, `eigh`, and `svd`, when performed on a 10000×10000 matrix using PyTorch and JAX. The presented values represent the mean ratios of the execution time relative to that of matrix multiplication (`matmul`) of 100 runs.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

Despite its straightforward definition, the computation of eigenvalues is quite challenging. Eigendecomposition algorithms are inherently iterative, involving a sequence of expensive operations such as Arnoldi iteration, QR iteration, and Rayleigh quotient Iteration [39]. Additionally, it usually takes thousands of iterations to reach a desirable tolerance level. For instance, the locally optimal block preconditioned conjugate gradient (LOBPCG) method [22]–a widely-used eigenvalue solver–often requires dozens to hundreds of iterations to achieve convergence [22, 9]. Figure 1 presents a comparative analysis of the execution times for eigendecomposition and other computational operations using PyTorch [33] and JAX [2]. The figure shows that the execution speeds for `eigh` and `svd` are remarkably slower—by 4 to 5 orders of magnitude—relative to matrix multiplication. This substantial disparity in execution speed indicates operations such as `eigh` and `svd`, once used, will be the bottleneck of computation cost. Conversely, the QR decomposition, often employed for orthogonality, is considerably less computationally expensive. This observation has motivated us to explore the possibility of reducing the iterative computation of eigendecomposition with lower-cost operations.

When eigendecomposition is incorporated into the training of a neural network, a nested loop scenario arises, where eigendecomposition acts as the inner loop and the neural network's loss minimization serves as the outer loop. Notably, it is not always easy to be aware of this inner loop of eigendecomposition, as it is encapsulated by the high-level functions provided within deep learning frameworks. However, in this context, this inner loop does not require full convergence during each iteration, given that the remaining parameters have not reached optima. The inner loop can be relaxed and optimized jointly with the training loss, allowing for a more flexible and efficient training process. The key idea can be summarized as follows:

Eigendecomposition within a neural network does not have to reach full convergence during each training step; it simply needs to contribute to the desired outcome by the end of the training process.

In this paper, we present a novel approach named "amortized eigendecomposition" for training neural networks that require eigenvalues or eigenvectors. Instead of using computationally expensive eigendecomposition decoupled with the training of neural networks, we proposed to relax it into an unconstrained optimization problem on the Stiefel manifold by adding an eigen loss. This relaxation only requires a QR decomposition at each iteration, thus is more efficient. Moreover, through empirical observations, we have found that although the relaxed optimization problem with eigen loss does not involve eigendecomposition in every iteration, the amortized optimization approach consistently achieves the desired results. It achieves nearly identical performance to traditional methods but with significantly improved speed.

2 Eigendecomposition in Neural Networks

In this paper, we consider a general class of neural networks that incorporate eigendecomposition. We formulate this family of problems as a constrained optimization problem:

$$\min_{\boldsymbol{\theta}} f\big(h_{\boldsymbol{\theta}}(\boldsymbol{X}), \boldsymbol{V}, \boldsymbol{\Lambda}\big), \qquad \text{s.t. } \boldsymbol{V}^\mathsf{T}\boldsymbol{\Lambda}\boldsymbol{V} = \boldsymbol{A} \tag{1}$$

Here, the encoder $h_{\boldsymbol{\theta}}$ maps the data $\boldsymbol{X} \in \mathbb{R}^{n\times p}$ into a latent space. In addition to the latent representation, the loss function f also incorporates the eigenpair $\boldsymbol{\Lambda}$ and $\boldsymbol{V}$ of a symmetric matrix $\boldsymbol{A}$. The matrix $\boldsymbol{A}$ can be constructed from $h_{\boldsymbol{\theta}}(\boldsymbol{X})$, such as a covariance matrix or a similarity matrix, or it can depend solely on the parameters. Notably, $\boldsymbol{A}$ is subject to changes during network training due to its dependency on $\boldsymbol{\theta}$. The computational graph for each iteration of such model structure can be written as,

$$\underbrace{\boldsymbol{X} \longrightarrow h_{\boldsymbol{\theta}}(\boldsymbol{X})}_{\text{encoding}} \longrightarrow \underbrace{\boldsymbol{A} \longrightarrow (\boldsymbol{V}, \boldsymbol{\Lambda})}_{\text{eigendecomposition}} \longrightarrow \underbrace{f\big(h_{\boldsymbol{\theta}}(\boldsymbol{X}), \boldsymbol{V}, \boldsymbol{\Lambda}\big)}_{\text{downstream task}}. \tag{2}$$

The corresponding algorithm of the above computational graph is shown in Algorithm 1. Solving such problems, however, is computationally expensive, since it requires preserving the eigendecomposition constraint of $\boldsymbol{A}$ after each update of $\boldsymbol{\theta}$. This structure encompasses a wide range of learning problems. We present several representative examples and investigate them in our numerical experiments next.

Nuclear Norm Regularization The nuclear norm of a matrix is defined as the sum its the singular values. Due to its convexity, regularization via the nuclear norm is employed to encourage low-rank

structures within the learned parameters. This approach proves beneficial in a variety of applications, such as matrix completion [5, 4], image denoising [13, 12, 14, 45]. Furthermore, eigendecomposition and singular value decomposition are also used for pruning or compressing neural networks, by decomposing the weight matrices of the network and approximating the original network with fewer parameters [17, 30]. The objective function of this type of problem can be written as,

$$\min_{\boldsymbol{\theta}} f\big(h_{\boldsymbol{\theta}}(\boldsymbol{X})\big) + \eta\|\boldsymbol{\theta}\|_* \tag{3}$$

Here, η is a regularization coefficient that controls the rank of the parameter matrix. In classical methods, the nuclear norm $\|\boldsymbol{\theta}\|_*$ is usually calculated via singular value decomposition.

Whitening in Neural Networks Whitening is a transformation technique extensively utilized in neural networks to standardize features by ensuring they have zero mean and unit variance, and are decorrelated from each other. In neural network applications, whitening can be categorized into parameter-space whitening and feature-space whitening. Parameter-space whitening, often achieved through PCA/ZCA, is a prevalent method applied during neural network training to improve stability and accelerate convergence [37, 41, 16, 8]. Feature-space whitening, in contrast, applies PCA to the intermediate representations within the network. This process aligns features with the axes of greatest variance, thereby facilitating dimensionality reduction [15, 42, 34, 25]. A concrete example of feature-space whitening is its incorporation into an auto-encoder architecture, which can be expressed mathematically as:

$$\min_{\boldsymbol{\omega},\boldsymbol{\theta}} \Big\|\mathsf{Dec}_{\boldsymbol{\omega}}\Big(\boldsymbol{V}\boldsymbol{V}^{\mathsf{T}}\mathsf{Enc}_{\boldsymbol{\theta}}(\boldsymbol{X})\Big) - \boldsymbol{X}\Big\|_{\mathrm{F}} \tag{4}$$

where $\boldsymbol{V}$ is spanned by the first several largest eigenvectors of the covariance of $\mathsf{Enc}_{\boldsymbol{\theta}}(\boldsymbol{X})$ and $\boldsymbol{\theta}, \boldsymbol{\omega}$ represent the parameters of the decoder and encoder, respectively.

Graph Structure Learning Graph structure learning is an area of machine learning that aims to deduce the latent structure of a graph or network from observed data [23]. This domain has significant applications in graph adversarial learning, which seeks to bolster the robustness of graph neural networks against adversarial attacks [10, 18, 47]. In such contexts, the adjacency matrix is usually often compromised by adversarial modifications while feature matrix $\boldsymbol{X}$ remains unaffected. The primary objective is to learn both a clean graph structure and perform accurate node classification. From this perspective, we follow the approach proposed in [18] and the objective function can be formulated as:

$$\min_{\boldsymbol{\theta},\widehat{\boldsymbol{L}}} \big\|\mathsf{GNN}_{\boldsymbol{\theta}}(\boldsymbol{X},\widehat{\boldsymbol{L}}) - \boldsymbol{Y}\big\|_2 + \alpha\big\|\widehat{\boldsymbol{L}}\big\|_* + \beta\big\|\widehat{\boldsymbol{L}} - \boldsymbol{L}\big\|_{\mathrm{F}}^2 \tag{5}$$

Here $\widehat{\boldsymbol{L}}$ represents a low-rank Laplacian matrix that approximates the original graph Laplacian $\boldsymbol{L}$. The three parts of the loss function correspond to the node classification loss, low-rank constraint, and unnoticeable adversarial attacks, respectively. However, this approach relies on singular value decomposition at every iteration, which is computationally prohibitive for large-scale networks.

3 Differentiable Optimization for Eigendecomposition

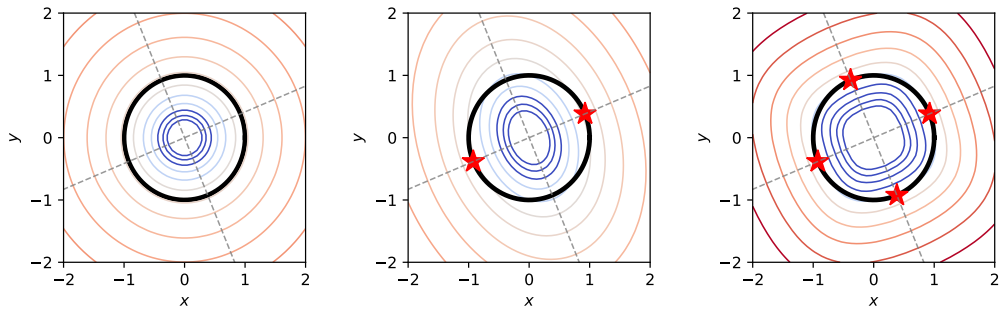
Before introducing the proposed method, let's begin by examining a more straightforward case: determining the eigenvalues of a symmetric matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ through constrained optimization.[1] The eigenvalues of $\boldsymbol{A}$ are denoted as $\lambda_1 \geqslant \lambda_2 \geqslant \cdots \geqslant \lambda_n$. The largest eigenvalue of $\boldsymbol{A}$ can be obtained by directly maximizing the *Rayleigh quotient*:

$$\lambda_1 = \max_{\boldsymbol{u}\in\mathbb{R}^n} \frac{\boldsymbol{u}^{\mathsf{T}}\boldsymbol{A}\boldsymbol{u}}{\boldsymbol{u}^{\mathsf{T}}\boldsymbol{u}}. \tag{6}$$

where the maximum value corresponds to the largest eigenvalue and the associated eigenvector is given by the normalized $\boldsymbol{u}$ at the optimum. This method can be generalized to identifying *the optimal subspace*, which is a common problem in dimensionality reduction and feature selection contexts. The optimal subspace for a symmetric matrix is defined as the subspace spanned by its k largest eigenvectors. This can be found by solving the constrained optimization problem:

$$\max_{\boldsymbol{U}\in\mathbb{U}^{n\times k}} \quad \mathsf{tr}(\boldsymbol{U}^{\mathsf{T}}\boldsymbol{A}\boldsymbol{U}) \tag{7}$$

[1] Although the findings in this paper can be readily extended to the complex space, we specifically focus on the real space throughout our study.



(a) Trace loss (b) Brockett's cost function (c) Convex trace loss

Figure 2: An illustration on disrupting the rotational symmetry of the trace loss. We aim to solve eigenvectors for a 2-dimensional symmetric matrix $\boldsymbol{A} = \begin{pmatrix} 0.8 & 0.2 \\ 0.2 & 0.4 \end{pmatrix}$ with three loss functions. We parameterize an orthonormal matrix $\boldsymbol{U} = \begin{pmatrix} x & y \\ -y & x \end{pmatrix}$, which is subject to the constraint $x^2 + y^2 = 1$. The plot displays the contours of the landscapes of three different loss functions as they vary with x and y: (**a**) trace loss $\mathsf{tr}(\boldsymbol{U}\boldsymbol{A}\boldsymbol{U}^\mathsf{T})$, (**b**) Brockett's cost function $\mathsf{tr}(\boldsymbol{M}\boldsymbol{U}\boldsymbol{A}\boldsymbol{U}^\mathsf{T})$ where $\boldsymbol{M} = \mathsf{diag}(1, 0.2)$ and (**c**) convex loss function $\mathsf{tr}(f(\boldsymbol{U}\boldsymbol{A}\boldsymbol{U}^\mathsf{T}))$ where f is an element-wise exponential function. The feasible area of the constraint is depicted with a black circle. The red stars signify the optima of the loss in the feasible area. The dashed grey lines represent the true eigenvector direction of $\boldsymbol{A}$. We see that, the trace loss results in infinitely many optimal solutions due to its rotational symmetry. In contrast, both Brockett's cost function and the convex loss function reshape the optimization landscape, breaking this symmetry and leading to the identification of the correct eigenvectors.

where the Stiefel manifold $\mathbb{U}^{n\times k} = \{\boldsymbol{U} \in \mathbb{R}^{n\times k} \mid \boldsymbol{U}^\mathsf{T}\boldsymbol{U} = \boldsymbol{I}\}$ is the set of $n \times k$ matrices with orthonormal columns. Maximizing this trace function yields the optimal subspace spanned by the column vectors of $\boldsymbol{U}$.

Rotational Symmetry It's important to recognize, however, that the optimal $\boldsymbol{U}$ of the above optimization problem does NOT represent the eigenvectors of $\boldsymbol{A}$. This is due to the rotational symmetry of the trace function that for any orthonormal matrix $\boldsymbol{C} \in \mathbb{U}^{k\times k}$, the equation $\mathsf{tr}(\boldsymbol{U}^\mathsf{T}\boldsymbol{A}\boldsymbol{U}) = \mathsf{tr}(\boldsymbol{C}^\mathsf{T}\boldsymbol{U}^\mathsf{T}\boldsymbol{A}\boldsymbol{U}\boldsymbol{C})$ always holds. As a result, there are an infinite number of solutions to the optimization problem Eq. (7), all spanning the same subspace as the desired sets of eigenvectors. A visual illustration is provided in Figure 2a.

To accurately obtain the eigenvalues and eigenvectors, it is necessary to refine the traditional trace loss function. This paper introduces two approaches for achieving the correct eigendecomposition. The first method utilizes Brockett's cost function, which applies distinct weights to the diagonal elements of the matrix product $\boldsymbol{U}^\mathsf{T}\boldsymbol{A}\boldsymbol{U}$, effectively differentiating the importance of each eigenvalue. The second method involves applying an element-wise convex function directly to the diagonal elements, resulting in an exact eigendecomposition. We will now elaborate on these methods.

The Brockett's Cost Function The first modification made to the trace loss is Brockett's cost function [3]. It enables the extraction of eigenvectors and eigenvalues by solving the following optimization problem:

$$\max_{\boldsymbol{U}\in\mathbb{U}^{n\times k}} \quad \mathsf{tr}(\boldsymbol{M}\boldsymbol{U}^\mathsf{T}\boldsymbol{A}\boldsymbol{U}). \tag{8}$$

Here $\boldsymbol{M} = \mathsf{diag}(m_1, m_2, \cdots, m_k) \in \mathbb{R}^{k\times k}$ is a diagonal weight matrix with distinct diagonal elements. The matrix $\boldsymbol{M}$ is structured such that all its elements are distinct numbers. Specifically, we can denote these elements as $0 < m_1 < m_2 < \cdots < m_k$. This ordering allows $\boldsymbol{M}$ to assign distinct weights to the diagonal elements of the product $\boldsymbol{U}^\mathsf{T}\boldsymbol{A}\boldsymbol{U}$, effectively disrupting the rotational invariance inherent in the trace loss and thus enabling the determination of eigenvalues and eigenvectors through optimization. In fact, $\boldsymbol{M}$ can be any diagonal matrix with distinct diagonal

elements. A practical choice could be $m_i = i/k$ or $m_i = i$ as suggested by [3]. The next theorem illustrates why this approach can yield the exact eigendecomposition.

Theorem 1 (Trace inequality with weight matrix)**.** *Consider a n-dimensional symmetric matrix $\boldsymbol{A}$. Let $\boldsymbol{M}$ be a k-dimensional diagonal matrix with elements $0 < m_1 < m_2 \cdots < m_k$. For an arbitrary matrix $\boldsymbol{U} \in \mathbb{U}^{n\times k}$ with orthogonal columns, the following inequality always holds:*

$$\sum_{i=1}^{k} m_{k-i+1}\lambda_i \leqslant \mathsf{tr}(\boldsymbol{M}\boldsymbol{U}^\mathsf{T}\boldsymbol{A}\boldsymbol{U}) \leqslant \sum_{i=1}^{k} m_i\lambda_i. \tag{9}$$

The equalities are achieved if and only if $\boldsymbol{U}$ contains the eigenvectors corresponding to the k largest (smallest) eigenvalues of $\boldsymbol{A}$.

This approach can be viewed as an extension of the trace inequalities originally put forward by Von Neumann [40, 31] and further developed by Ruhe [35]. A detailed demonstration is available in the seminal work by Brockett [3]. Additionally, we offer an alternative proof leveraging the Cauchy–Schwarz inequality, which is presented in the Appendix for reference. A more generalized version of this trace inequality is discussed in Liang et al. [26].

The Convex Trace Loss The second method employs a strictly monotonic convex function f, applied element-wise to the diagonal components of $\boldsymbol{U}^\mathsf{T}\boldsymbol{A}\boldsymbol{U}$. This perturbation also disrupts the rotational symmetry inherent in the trace loss. The convex nature of f alters the curvature of the loss landscape, thereby ensuring a unique optimal solution corresponding to the eigenvectors. The convex trace loss function, aimed at extracting the k largest eigenvalues, is expressed as:

$$\max_{\boldsymbol{U}\in\mathbb{U}^{n\times k}} \quad \mathsf{tr} f(\boldsymbol{U}^\mathsf{T}\boldsymbol{A}\boldsymbol{U}) \tag{10}$$

The optimal $\boldsymbol{U}_*$ that achieves the maximum in the above objective are the eigenvectors corresponding to the k largest eigenvalues, and the eigenvalues can be obtained by the diagonal elements of $\boldsymbol{U}_*^\mathsf{T}\boldsymbol{A}\boldsymbol{U}_*$. The next theorem provides a formal validation of this assertion.

Theorem 2 (Trace inequality with convex function)**.** *Let $\boldsymbol{A}$ be a given n-dimensional symmetric matrix and let $\boldsymbol{U}$ be a matrix of size $n \times k$ that resides on the Stiefel manifold. Suppose $f : \mathbb{R} \to \mathbb{R}$ be a monotonically increasing, convex function applied element-wise. The following inequalities hold:*

$$k\left(f\left(\frac{1}{k}\mathsf{tr}(\boldsymbol{A})\right)\right) \leqslant \mathsf{tr}\Big(f\big(\boldsymbol{U}^\mathsf{T}\boldsymbol{A}\boldsymbol{U}\big)\Big) \leqslant \sum_{i=1}^{k} f(\lambda_i), \tag{11}$$

The rightmost inequality becomes an equality if and only if $\boldsymbol{U}$ comprises the eigenvectors of $\boldsymbol{A}$ that correspond to the k largest eigenvalues, The leftmost inequality is met with equality when $\boldsymbol{U}$ is such that all diagonal elements of the matrix $\boldsymbol{U}^\mathsf{T}\boldsymbol{A}\boldsymbol{U}$ are equal.

This theorem is established through the application of Jensen's inequality. The detailed proof is provided in the Appendix B.3 for reference. In the above objective, suitable choices of f include: $f(x) = \exp(x)$ on $\mathbb{R}$; $f(x) = x^\alpha$ on $\mathbb{R}^+$ where $\alpha > 1$; and $f(x) = \tan(x)$ on $[0, \pi/2]$. For finding the k smallest eigenvalues, a simple modification can be made by replacing the function f with an element-wise monotonically increasing concave function and then minimizing the trace loss. A visual representation of the optimization process of the three trace losses is presented in Figure 2.

4 The Amortized Eigendecomposition Approach

The proposed amortized eigendecomposition approach aims to modify the eigendecomposition operation within the neural network's computational graph, as illustrated in Eq. (2). This replacement involves two steps: First, the set of eigenvectors form a matrix on the Stiefel manifold reparameterized through computationally efficient operations such as QR decomposition. Then, the loss function is adjusted to ensure that its optimal solutions precisely correspond to the eigendecomposition. The computational graph for this amortized eigendecomposition method is formulated as follows:

$$\underbrace{\boldsymbol{X} \longrightarrow h_{\boldsymbol{\theta}}(\boldsymbol{X})}_{\text{encoder}} \longrightarrow \boldsymbol{A} \longrightarrow \underbrace{f_{\boldsymbol{\omega}}\big(h_{\boldsymbol{\theta}}(\boldsymbol{X}), \boldsymbol{U}, \boldsymbol{\Sigma}\big)}_{\text{model loss}} + \eta\, \underbrace{g\,(\boldsymbol{U}, \boldsymbol{A})}_{\text{eigen loss}} \longleftarrow \underbrace{\boldsymbol{U} \xleftarrow{\texttt{QR}} \boldsymbol{W}}_{\text{reparameterization}}. \tag{12}$$

In the computation graph, the eigendecomposition operation is circumvented and substituted with a more efficient QR operation. The QR operation is employed to reparameterize the orthogonal matrix

Algorithm 1 The conventional eigendecomposition in a neural network outlined in Eq. (2)

Input: Dataset $\boldsymbol{X}$, encoder $h_{\boldsymbol{\theta}}$, task $f_{\boldsymbol{\omega}}$;
Initialize model parameter $\boldsymbol{\theta}$ and $\boldsymbol{\omega}$;
while not converged **do**
  compute $\boldsymbol{A}$ from $h_{\boldsymbol{\theta}}(\boldsymbol{X})$;
  $\boldsymbol{V}, \boldsymbol{\Lambda} = \mathsf{eigh}(\boldsymbol{A})$;
  compute $f_{\boldsymbol{\omega}}(h_{\boldsymbol{\theta}}(\boldsymbol{X}), \boldsymbol{V}, \boldsymbol{\Lambda})$;
  update $\boldsymbol{\theta}$, $\boldsymbol{\omega}$ by gradient descent;
end while

Algorithm 2 The amortized eigendecomposition technique outlined in Eq. (12)

Input: Dataset $\boldsymbol{X}$, encoder $h_{\boldsymbol{\theta}}$, task $f_{\boldsymbol{\omega}}$, and η;
Initialize model parameter $\boldsymbol{\theta}$, $\boldsymbol{\omega}$ and $\boldsymbol{W}$;
while not converged **do**
  compute $\boldsymbol{A}$ from $h_{\boldsymbol{\theta}}(\boldsymbol{X})$;
  $\boldsymbol{U} = \mathsf{QR}(\boldsymbol{W})$, $\boldsymbol{\Sigma} = \mathsf{diag}(\boldsymbol{U}^\top \boldsymbol{A} \boldsymbol{U})$;
  compute $f_{\boldsymbol{\omega}}(h_{\boldsymbol{\theta}}(\boldsymbol{X}), \boldsymbol{U}, \boldsymbol{\Sigma}) + \eta g(\boldsymbol{A}, \boldsymbol{U})$;
  update $\boldsymbol{\theta}$, $\boldsymbol{\omega}$, $\boldsymbol{W}$ by gradient descent;
end while

on the Stiefel Manifold, leading to a substantial acceleration at each iteration. Moreover, instead of forcing the eigendecomposition constraint at each iteration, we relax it to an eigen loss which is jointly optimized with the training loss of the neural network as a nested optimization loop. This training process is outlined in Algorithm 2, where the key difference of our amortized optimization for the eigendecomposition approach is highlighted in red background color.

Reparameterize the Stiefel Manifold There are three prevalent methods for reparameterizing an orthogonal matrix: through the matrix exponential, the Cayley transform, and QR decomposition. Due to the QR decomposition's better numerical stability and efficiency for non-square matrices $\boldsymbol{U}$, we employ it for reparameterizing a matrix with orthonormal columns:

$$\boldsymbol{U} = \mathsf{QR}(\boldsymbol{W}). \tag{13}$$

In this formulation, $\boldsymbol{W}$ is dimensionally consistent with $\boldsymbol{U}$. QR decomposition is more computationally efficient than eigendecomposition and singular value decomposition as shown in Fig. 1. Additionally, the backward computation of the QR decomposition is well-defined and has been efficiently optimized in modern deep learning frameworks, such as PyTorch and JAX. For details on the back-propagation process of the QR decomposition, see [36].

Relaxation with Eigen Loss Previously, we observed that optimizing the Brockett-type or convex trace loss directly enables us to obtain precise eigenvalues and eigenvectors. For any loss function that depends on the eigenvectors or eigenvalues, as specified in Eq.(1), we can transform this loss into a regularized version incorporating the eigen loss. This relaxation allows us to forego the need for explicit eigendecomposition at every iteration, while ultimately achieving equivalent outcomes. We now examine several general scenarios that encompass the majority of cases and explore how to implement this relaxation technique.

In the general case, the model loss in Eq. (1) depends on both the eigenvectors and eigenvalues. This constrained optimization problem can be relaxed by introducing an eigen loss as a regularizer, which is formulated as:

$$\min_{\boldsymbol{\theta}} f\big(h_{\boldsymbol{\theta}}(\boldsymbol{X}), \boldsymbol{V}, \boldsymbol{\Lambda}\big) \xrightarrow[\textit{largest}\ \text{eigenpair}]{(\boldsymbol{V}, \boldsymbol{\Lambda})\ \text{is the}\ k} \min_{\boldsymbol{\theta}, \boldsymbol{W}} f\big(h_{\boldsymbol{\theta}}(\boldsymbol{X}), \boldsymbol{U}, \boldsymbol{\Sigma}\big) - \eta \mathsf{tr}(\boldsymbol{M} \boldsymbol{U}^\top \boldsymbol{A}_{\boldsymbol{\theta}} \boldsymbol{U}). \tag{14}$$

In this reformulation, $\boldsymbol{U} \in \mathbb{U}^{n \times k}$, is reparameterized via a QR operation as shown in Eq. (13). This relaxation circumvents the need for eigendecomposition at each iteration by using the computationally cheaper QR decomposition, while still ensuring that the optimal solution corresponds to the precise eigendecomposition of the matrix $\boldsymbol{A}_{\boldsymbol{\theta}}$.

The second type of optimization problem involves scenarios where the model loss is independent of the eigenvalues and depends solely on the eigenvectors, such as in the latent-space PCA network expressed in Eq. (4). To enhance the efficiency of the solution process, the problem can be reformulated to include a trace penalty term, such as Brockett's cost, which is given by

$$\min_{\boldsymbol{\theta}} f\big(h_{\boldsymbol{\theta}}(\boldsymbol{X}), \boldsymbol{V}\big) \xrightarrow[\text{eigenvectors}]{\boldsymbol{V}\ \text{is}\ k\ \textit{largest}} \min_{\boldsymbol{\theta}, \boldsymbol{W}} f\big(h_{\boldsymbol{\theta}}(\boldsymbol{X}), \boldsymbol{U}\big) - \eta \mathsf{tr}(\boldsymbol{M} \boldsymbol{U}^\top \mathsf{StopGrad}(\boldsymbol{A}_{\boldsymbol{\theta}}) \boldsymbol{U}). \tag{15}$$

In this formulation, a stop gradient operation is applied to $\boldsymbol{A}$, since the eigen loss involves $\boldsymbol{A}$ which relies on the parameter $\boldsymbol{\theta}$. By introducing the stop gradient operation, we prevent this regularization term from propagating gradients back to $\boldsymbol{\theta}$.

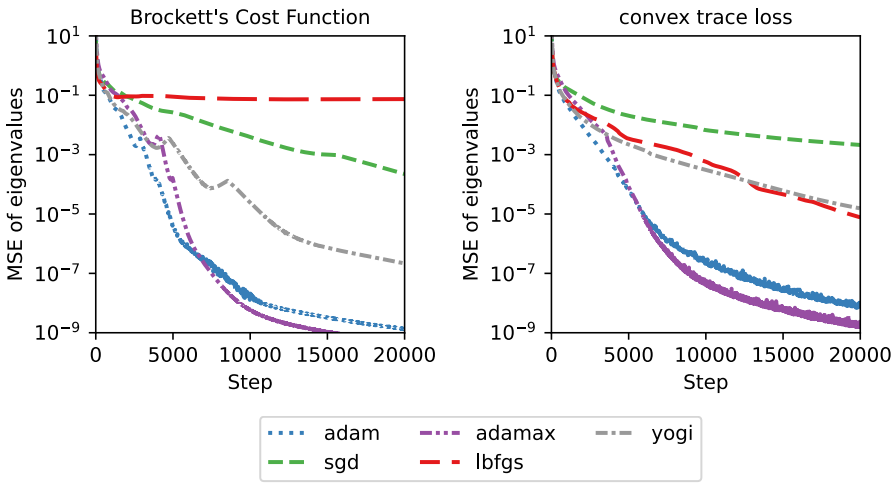


Figure 3: Convergence analysis on finding 50 largest eigenvalues on random 1000×1000-dimensional symmetric matrices. (a): Convergence curves using Brockett's cost and convex trace loss ($f(x) = x^{1.5}$). (b) The fine-tuning convergence on a series of similar matrices.

Besides the above relaxations, further simplification is possible if the loss function conforms to the structure of Brockett's cost function or a convex trace function. That is, when the model loss $f_{\boldsymbol{\omega}}$ is a non-uniform linear combination of the eigenvalues, or it is monotonic and convex (or concave) to the eigenvalues, the trace penalty term is unnecessary. The loss function will inherently converge to the correct eigendecomposition. This principle is exemplified in the context of the nuclear norm regularization problem, which will be illustrated later. This can be formulated as:

$$\min_{\boldsymbol{\theta}} f\big(h_{\boldsymbol{\theta}}(\boldsymbol{X}), \boldsymbol{V}, \boldsymbol{\Lambda}\big) \xrightarrow[\text{or a non-uniform linear combination of } \boldsymbol{\Lambda}]{f \text{ is monotonic and concave w.r.t } \boldsymbol{\Lambda}} \min_{\boldsymbol{\theta}, \boldsymbol{W}} f\big(h_{\boldsymbol{\theta}}(\boldsymbol{X}), \boldsymbol{U}, \boldsymbol{\Sigma}\big). \tag{16}$$

5 Experiments

In this section, we present an evaluation of our approach, focusing on four specific tasks. Firstly, we demonstrate the convergence properties of our method empirically. Next, we measure the efficacy and efficiency of our amortized eigendecomposition technique over nuclear norm regularized auto-encoder and latent-space PCA using the MNIST dataset. Lastly, we assess the effectiveness of our approach in the context of graph adversarial learning tasks. We implement our approach with the deep learning framework JAX [2]. All the experiments of our approach are conducted on a single NVIDIA A100 GPU with 40GB memory. The two fundamental questions we investigate are as follows:

- *Does our approach accurately identify the eigendecomposition and singular value decomposition?*
- *How does the efficiency of our method compare to that of traditional techniques?*

5.1 Convergence

In this experiment, we evaluate the numerical error and convergence speed of our algorithm applied to solving eigendecomposition. We randomly generate ten symmetric matrices of size 1000×1000. The first 50 eigenvalues of these matrices range from 1 to 50, while the remaining eigenvalues lie between 0 and 1. Our objective is to compute the 50 largest eigenvalues by minimizing Brockett's cost function and convex trace loss (we adopt $f(x) = x^{1.5}$ as the convex function). To achieve this, we employ several optimization algorithms, including Adam [20], Adamax [20], Yogi [46], SGD, and L-BFGS [28]. These algorithms are provided by the Optax and JAX-opt libraries [7]. We measure the mean square error (MSE) of the eigenvalues to the number of training iterations. The results are illustrated in Figure 3.

Table 1: Evaluation of execution times per iteration on three tasks.

Task	Dimension	Backbone time (s/iter) t_0	Backbone+ eigh/svd time (s/iter) t_1	Backbone+ our method time (s/iter) t_2	Speed-up $\frac{t_1-t_0}{t_2-t_0}$
Nuclear norm regularization	128×128	5.275E-2	8.323E-2	6.025E-2	$4.06\times$
	256×256	5.600E-2	1.209E-1	6.080E-2	$13.5\times$
	512×512	7.186E-2	2.616E-1	7.366E-2	$105.4\times$
Latent-space PCA	256×2	4.178E-3	1.446E-2	1.117E-2	$1.47\times$
	512×2	6.792E-3	2.918E-2	2.224E-2	$1.45\times$
	1028×2	1.434E-2	7.018E-2	5.467E-2	$1.39\times$
Low-rank GCN	2708×16	1.021E-3	1.769E-2	1.732E-3	$23.4\times$
	3312×16	1.367E-3	2.825E-2	2.498E-3	$23.7\times$
	19717×16	1.931E-2	4.941E+0	2.731E-2	$615.2\times$

This result demonstrates that both loss functions are capable of identifying the correct eigenvalues with a small numerical error of 10^{-9}. However, there is a noticeable difference in convergence speed. For both trace loss functions, the Adam and Adamax optimizers outperform the others, achieving faster convergence rates. Brockett's cost function, which introduces a linear combination of the trace elements, is more numerically stable compared to the convex trace loss, resulting in faster convergence. This experiment validates the efficiency of the differentiable optimization framework for computing the k largest eigenvalues.

5.2 Nuclear Norm Regularization

In this experiment, we apply the amortized eigendecomposition approach to the nuclear norm regularization problem, as outlined in Eq. (3). The experimental framework entails training an auto-encoder on the MNIST dataset by minimizing the reconstruction loss with a nuclear norm regularizer applied to the weight matrix $\boldsymbol{\theta} \in \mathbb{R}^{n\times k}$ of the encoder's last layer. We employ a relaxation technique to the original problem defined in Eq. (16), which can be expressed as:

$$\min_{\boldsymbol{\omega},\boldsymbol{\theta}} \left\| \mathsf{Dec}_{\boldsymbol{\omega}}\big(\mathsf{Enc}_{\boldsymbol{\theta}}(\boldsymbol{X})\big) - \boldsymbol{X} \right\|_{\mathrm{F}} + \eta \|\boldsymbol{\theta}\|_* \longrightarrow \min_{\boldsymbol{\omega},\boldsymbol{\theta},\boldsymbol{W}} \left\| \mathsf{Dec}_{\boldsymbol{\omega}}\big(\mathsf{Enc}_{\boldsymbol{\theta}}(\boldsymbol{X})\big) - \boldsymbol{X} \right\|_{\mathrm{F}} + \eta \sum_{i=1}^{k} \left\| \boldsymbol{\theta}\boldsymbol{u}_i \right\|_2 \quad (17)$$

where $\boldsymbol{u}_i$'s are the orthogonal column vectors of $\boldsymbol{U}$, which are parameterized by $\boldsymbol{W}$ through Eq. (13). The architectures of the encoder and the decoder are constructed as a 2-layer MLP with hidden layer dimensions of D = 128, 256, and 512. For comparison, we also implement the approach based on singular value decomposition (using the svd function). It should be noted that in the current versions of JAX, both the eigh and svd functions are limited to operations on full matrices.

The average execution time per iteration for the baseline backbone with only reconstruction loss, the backbone with svd, i.e. LHS of Eq. (17), and the backbone utilizing amortized eigendecomposition, i.e. RHS of Eq. (17) are reported in Table 1. We denote these execution times as t_0, t_1, and t_2 respectively, and define the speed-up ratio for our approach relative to the svd as:

$$\text{speed-up} = \frac{t_1 - t_0}{t_2 - t_0}. \quad (18)$$

This ratio represents the improvement in execution speed of our eigendecomposition method compared to the standard svd, relative to the baseline backbone performance.

5.3 Latent-space Principle Component Analysis

We investigate the effectiveness of our approach for the latent-space PCA method, as described in Eq. (4), using the MNIST dataset. Computing the eigenvectors of the large-scale covariance matrix in each iteration significantly increases the computation overhead, while our method amortizes this cost by jointly minimizing an additional loss. Moreover, we also aim to ensure that the first two eigenvalues are significantly larger than the subsequent ones. In the following objective function, the

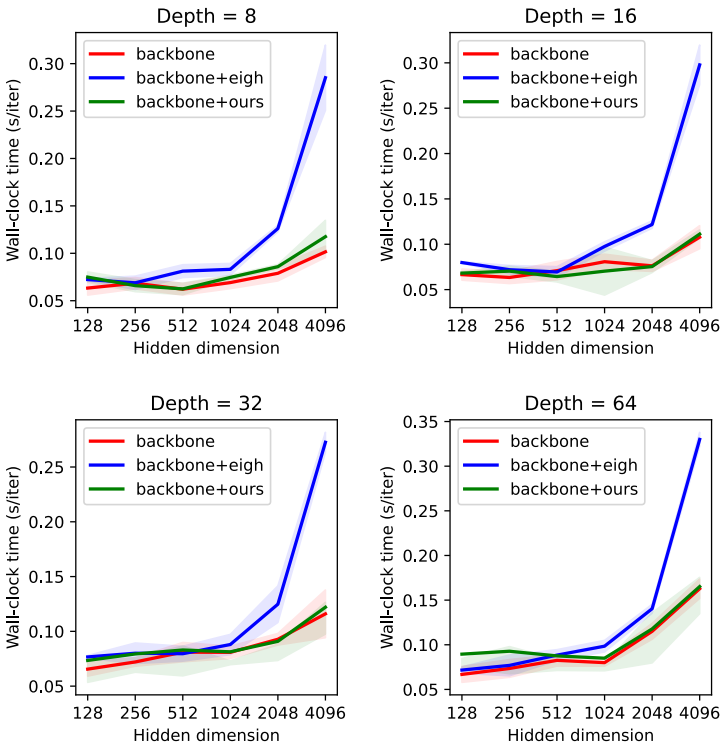


Figure 4: A comparison of scaling in the latent PCA task using the Celeb-A-HQ (256x256) dataset. The backbone autoencoders used in this study consist entirely of fully-connected layers with ReLU activation, all maintaining the same dimensions. Between the encoder and decoder, we applied both an eigen solver from the JAX `eigh` function and our amortized eigendecomposition method. We varied the depth of the autoencoders across 8, 16, 32, and 64 layers, and explored dimensionalities of 128, 256, 512, 1024, 2048, and 4096. The results present the average execution time per iteration over 100 runs. Notably, the largest model tested, featuring an autoencoder with 64 layers and a dimension of 4096, comprises up to 1.0 billion parameters.

additional term resembles Brockett's cost function while the trace of the covariance matrix ensures homogeneity:

$$\min_{\boldsymbol{\omega},\boldsymbol{\theta},\boldsymbol{W}} \left\| \mathsf{Dec}_{\boldsymbol{\omega}}\left(\boldsymbol{U}\boldsymbol{U}^{T}\mathsf{Enc}_{\boldsymbol{\theta}}(\boldsymbol{X})\right) - \boldsymbol{X} \right\|_{\mathrm{F}} - \eta \frac{\mathsf{tr}(\boldsymbol{M}\boldsymbol{U}^{\mathsf{T}}\mathrm{cov}(h_{\boldsymbol{\theta}}(\boldsymbol{X}))\boldsymbol{U})}{\mathsf{tr}(\mathrm{cov}(h_{\boldsymbol{\theta}}(\boldsymbol{X})))}, \tag{19}$$

where cov represents the covariance function. The architecture for both the encoder and decoder mirrors that of the nuclear norm regularization model, with the exception that there is a linear projection aligning with the direction of the principal components. The average execution times are reported in Table 1. More results and analysis are provided in Appendix A.2. The experimental results demonstrate that our approach achieves an average training speed improvement of 40% compared to the conventional eigendecomposition approach.

Additionally, we conducted a scalability study, with the results presented in Figure 4. This study examined the scaling behavior of latent PCA on the Celeb-A-HQ (256x256) dataset [29] by varying both the depth and width of the backbone autoencoder, with average execution time per iteration reported. The largest model in our tests, a 64-layer autoencoder with a 4096-dimensional latent space, contains over 1 billion parameters. From these results, we draw two main conclusions. First, our amortized eigen loss substantially reduces the eigendecomposition training time without significantly increasing the computational load of the backbone, as evidenced by the close alignment of the red (backbone) and green (backbone + our approach) lines. In contrast, the traditional eigendecomposition approach (blue line) scales steeply with increasing dimensionality, whereas our approach exhibits a much slower growth rate. Second, eigendecomposition emerges as the primary computational bottleneck within these neural network architectures, while the fully-connected layer computation

remains minor, particularly at large widths (>2000). This is reflected in the widening gap between the `backbone` (red line) and `backbone + eigh` (blue line) as dimensionality increases. Notably, increasing the depth of the backbone while keeping the hidden dimension constant results in minimal change in execution time, indicating that the cost of fully-connected layers is small relative to eigendecomposition—further underscoring the results shown in Figure 1.

5.4 Adversarial Attacks on Graph Convolutional Networks

In this study, we explore the robustness of graph convolutional networks (GCNs) [21] by implementing adversarial attacks on graph structures. The objective of this problem is described in Eq. (5). Our approach simplifies the attainment of a low-rank structure by optimizing:

$$\min_{\boldsymbol{\theta}, \boldsymbol{W}} \quad \left\|\mathsf{GCN}_{\boldsymbol{\theta}}(\boldsymbol{X}, \widehat{\boldsymbol{L}}) - \boldsymbol{Y}\right\|_2 - \eta \mathsf{tr}(\boldsymbol{M}\boldsymbol{U}^\mathsf{T}\boldsymbol{L}\boldsymbol{U}), \tag{20}$$

where $\widehat{\boldsymbol{L}} = \boldsymbol{U}^\mathsf{T}\mathsf{diag}(\boldsymbol{U}^\mathsf{T}\boldsymbol{L}\boldsymbol{U})\boldsymbol{U}$, with $\boldsymbol{U}$ being parameterized by Eq. (13). The motivation for this formulation is that at optimum, the columns of $\boldsymbol{U}$ correspond to the top-k eigenvectors of $\boldsymbol{L}$. Then $\widehat{\boldsymbol{L}}$ becomes the best rank k approximation of $\boldsymbol{L}$ under the Frobenius norm which corresponds to the terms $\left\|\widehat{\boldsymbol{L}}\right\|_*, \left\|\widehat{\boldsymbol{L}} - \boldsymbol{L}\right\|_\mathrm{F}^2$ in Eq. (5). This formulation allows the GNN to operate on a low-rank graph $\widehat{\boldsymbol{L}}$, which has been shown to enhance robustness against adversarial attacks on the graph structure.

Our architecture consists of a three-layer GCN, which is utilized for semi-supervised node classification tasks on several citation networks, namely Cora, Citeseer, and Pubmed. Each layer has a hidden dimension of 32. The dropout rates are set to 0.4 for Cora and Citeseer, and to 0.1 for Pubmed, to prevent overfitting. For optimization, we employ the Adam algorithm with a learning rate of 10^{-3}.

The adversarial attacks are executed by perturbing the graph structure through the random addition and deletion of edges in the adjacency matrix. We quantify the extent of these perturbations using a *contamination rate*, which is defined as the ratio of the altered edge count to the total node pairs.

Additionally, we propose a graph modification based on the original objective, as detailed in Eq. (5). The graph Laplacian is represented by a symmetric matrix, on which the eigh function is applied to compute the eigenvalues. The corresponding execution times are documented in Table 1. For a more comprehensive experiment results and analysis of our findings, please refer to the Appendix A.3.

6 Discussion and Conclusion

In this study, we address a class of deep learning problems that incorporate eigendecomposition within their constraints or objective functions. Such problems are prevalent in applications like nuclear-norm regularized denoising, network compression, graph structure learning, and whitening normalization. The traditional approach requires performing eigendecomposition or singular value decomposition within the computational graph, which becomes the bottleneck in the training process. To circumvent this computation overhead, we introduce an amortized eigendecomposition framework integrating a relaxation eigen loss into the learning objective, which relies on a set of orthonormal vectors. These vectors are reparameterized efficiently through QR decomposition, thereby substantially reducing the computational cost of eigendecomposition in each iteration. Furthermore, the differentiable nature of QR decomposition allows for its seamless incorporation into the neural network training workflow. Our experimental results demonstrate that, when applied to network tasks, our algorithm not only accelerates the process but also maintains the precision of the conventional method with eigendecomposition. The method proves particularly beneficial as a differentiable top k eigensolver in environments where the backward gradient computation for top k eigendecomposition, such as in JAX or PyTorch, is not well-supported. Consequently, our approach provides a viable alternative for integrating top k eigendecomposition into neural networks.

Limitations. While our method excels in scenarios where eigendecomposition operations on large matrices are embedded within neural networks, it is important to note that when employed as a pure numerical eigensolver, it does not offer any speed or precision advantages over conventional methods. Thus, its suitability is specifically aligned with applications where eigendecomposition is a substantial component of the neural network training process.

References

[1] P-A Absil, Robert Mahony, and Rodolphe Sepulchre. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2008.

[2] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[3] Roger W Brockett. Dynamical systems that sort lists, diagonalize matrices, and solve linear programming problems. *Linear Algebra and its applications*, 146:79–91, 1991.

[4] HanQin Cai, Jialin Liu, and Wotao Yin. Learned robust pca: A scalable deep unfolding approach for high-dimensional outlier detection. *Advances in Neural Information Processing Systems*, 34:16977–16989, 2021.

[5] Emmanuel J Candès and Terence Tao. The power of convex relaxation: Near-optimal matrix completion. *IEEE transactions on information theory*, 56(5):2053–2080, 2010.

[6] Tsung-Han Chan, Kui Jia, Shenghua Gao, Jiwen Lu, Zinan Zeng, and Yi Ma. Pcanet: A simple deep learning baseline for image classification? *IEEE transactions on image processing*, 24(12):5017–5032, 2015.

[7] DeepMind, Igor Babuschkin, Kate Baumli, Alison Bell, Surya Bhupatiraju, Jake Bruce, Peter Buchlovsky, David Budden, Trevor Cai, Aidan Clark, Ivo Danihelka, Antoine Dedieu, Claudio Fantacci, Jonathan Godwin, Chris Jones, Ross Hemsley, Tom Hennigan, Matteo Hessel, Shaobo Hou, Steven Kapturowski, Thomas Keck, Iurii Kemaev, Michael King, Markus Kunesch, Lena Martens, Hamza Merzic, Vladimir Mikulik, Tamara Norman, George Papamakarios, John Quan, Roman Ring, Francisco Ruiz, Alvaro Sanchez, Laurent Sartran, Rosalia Schneider, Eren Sezener, Stephen Spencer, Srivatsan Srinivasan, Miloš Stanojević, Wojciech Stokowiec, Luyu Wang, Guangyao Zhou, and Fabio Viola. The DeepMind JAX Ecosystem, 2020.

[8] Guillaume Desjardins, Karen Simonyan, Razvan Pascanu, et al. Natural neural networks. *Advances in neural information processing systems*, 28, 2015.

[9] Jed A Duersch, Meiyue Shao, Chao Yang, and Ming Gu. A robust and efficient implementation of lobpcg. *SIAM Journal on Scientific Computing*, 40(5):C655–C676, 2018.

[10] Negin Entezari, Saba A Al-Sayouri, Amirali Darvishzadeh, and Evangelos E Papalexakis. All you need is low (rank) defending against adversarial attacks on graphs. In *Proceedings of the 13th international conference on web search and data mining*, pages 169–177, 2020.

[11] Farzan Farnia, Jesse M. Zhang, and David Tse. Generalizable adversarial training via spectral normalization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[12] Shuhang Gu, Qi Xie, Deyu Meng, Wangmeng Zuo, Xiangchu Feng, and Lei Zhang. Weighted nuclear norm minimization and its applications to low level vision. *International journal of computer vision*, 121:183–208, 2017.

[13] Shuhang Gu, Lei Zhang, Wangmeng Zuo, and Xiangchu Feng. Weighted nuclear norm minimization with application to image denoising. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2862–2869, 2014.

[14] Cho-Jui Hsieh and Peder Olsen. Nuclear norm minimization via active subspace selection. In *International Conference on Machine Learning*, pages 575–583. PMLR, 2014.

[15] William W Hsieh. Nonlinear principal component analysis by neural networks. *Tellus A*, 53(5):599–615, 2001.

[16] Lei Huang, Dawei Yang, Bo Lang, and Jia Deng. Decorrelated batch normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 791–800, 2018.

[17] M Jaderberg, A Vedaldi, and A Zisserman. Speeding up convolutional neural networks with low rank expansions. In *BMVC 2014-Proceedings of the British Machine Vision Conference 2014*. British Machine Vision Association, 2014.

[18] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. Graph structure learning for robust graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 66–74, 2020.

[19] Anton Johansson, Niklas Engsner, Claes Strannegård, and Petter Mostad. Improved spectral norm regularization for neural networks. In *Modeling Decisions for Artificial Intelligence: 20th International Conference, MDAI 2023, Umeå, Sweden, June 19–22, 2023, Proceedings*, page 181–201, Berlin, Heidelberg, 2023. Springer-Verlag.

[20] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[21] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

[22] Andrew V Knyazev. Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method. *SIAM journal on scientific computing*, 23(2):517–541, 2001.

[23] Sandeep Kumar, Jiaxi Ying, José Vinícius de M Cardoso, and Daniel P Palomar. A unified framework for structured graph learning via spectral constraints. *Journal of Machine Learning Research*, 21(22):1–60, 2020.

[24] Rongjie Lai and Stanley Osher. A splitting method for orthogonality constrained problems. *Journal of Scientific Computing*, 58:431–449, 2014.

[25] Shen Li and Bryan Hooi. Neural pca for flow-based representation learning. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 3229–3235. International Joint Conferences on Artificial Intelligence Organization, 7 2022.

[26] Xin Liang, Li Wang, Lei-Hong Zhang, and Ren-Cang Li. On generalizing trace minimization principles. *Linear Algebra and its Applications*, 656:483–509, 2023.

[27] Zhouchen Lin, Minming Chen, and Yi Ma. The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. *arXiv preprint arXiv:1009.5055*, 2010.

[28] Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

[29] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[30] Marc Masana, Joost van de Weijer, Luis Herranz, Andrew D. Bagdanov, and Jose M. Alvarez. Domain-adaptive deep network compression. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[31] Leon Mirsky. A trace inequality of john von neumann. *Monatshefte für mathematik*, 79(4):303–306, 1975.

[32] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral Normalization for Generative Adversarial Networks. 2 2018.

[33] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[34] Michal Rolinek, Dominik Zietlow, and Georg Martius. Variational autoencoders pursue pca directions (by accident). In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12406–12415, 2019.

[35] Axel Ruhe. Perturbation bounds for means of eigenvalues and invariant subspaces. *BIT Numerical Mathematics*, 10(3):343–354, 1970.

[36] Matthias Seeger, Asmus Hetzel, Zhenwen Dai, Eric Meissner, and Neil D Lawrence. Auto-differentiating linear algebra. *arXiv preprint arXiv:1710.08717*, 2017.

[37] Aliaksandr Siarohin, Enver Sangineto, and Nicu Sebe. Whitening and coloring batch transform for gans. *arXiv preprint arXiv:1806.00420*, 2018.

[38] Kai Sun, Jiangshe Zhang, Hongwei Yong, and Junmin Liu. Fpcanet: Fisher discrimination for principal component analysis network. *Knowledge-Based Systems*, 166:108–117, 2019.

[39] Lloyd N Trefethen and David Bau. *Numerical linear algebra*. SIAM, 2022.

[40] John von Neumann. Some matrix-inequalities and metrization of matrix-space. *John von Neumann Collected Works*, 1937.

[41] Wei Wang, Zheng Dang, Yinlin Hu, Pascal Fua, and Mathieu Salzmann. Backpropagation-friendly eigendecomposition. *Advances in Neural Information Processing Systems*, 32, 2019.

[42] Wei Wang, Yan Huang, Yizhou Wang, and Liang Wang. Generalized autoencoder: A neural network framework for dimensionality reduction. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 490–497, 2014.

[43] Zaiwen Wen and Wotao Yin. A feasible method for optimization with orthogonality constraints. *Mathematical Programming*, 142(1):397–434, 2013.

[44] John Wright, Arvind Ganesh, Shankar Rao, Yigang Peng, and Yi Ma. Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization. *Advances in neural information processing systems*, 22, 2009.

[45] Chenggang Yan, Zhisheng Li, Yongbing Zhang, Yutao Liu, Xiangyang Ji, and Yongdong Zhang. Depth image denoising using nuclear norm and learning graph model. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 16(4):1–17, 2020.

[46] Manzil Zaheer, Sashank Reddi, Devendra Sachan, Satyen Kale, and Sanjiv Kumar. Adaptive methods for nonconvex optimization. *Advances in neural information processing systems*, 31, 2018.

[47] Yanqiao Zhu, Weizhi Xu, Jinghao Zhang, Qiang Liu, Shu Wu, and Liang Wang. Deep graph structure learning for robust representations: A survey. *arXiv preprint arXiv:2103.03036*, 14:1–1, 2021.

Appendix A More Experimental Results

A.1 Nuclear Norm Regularization

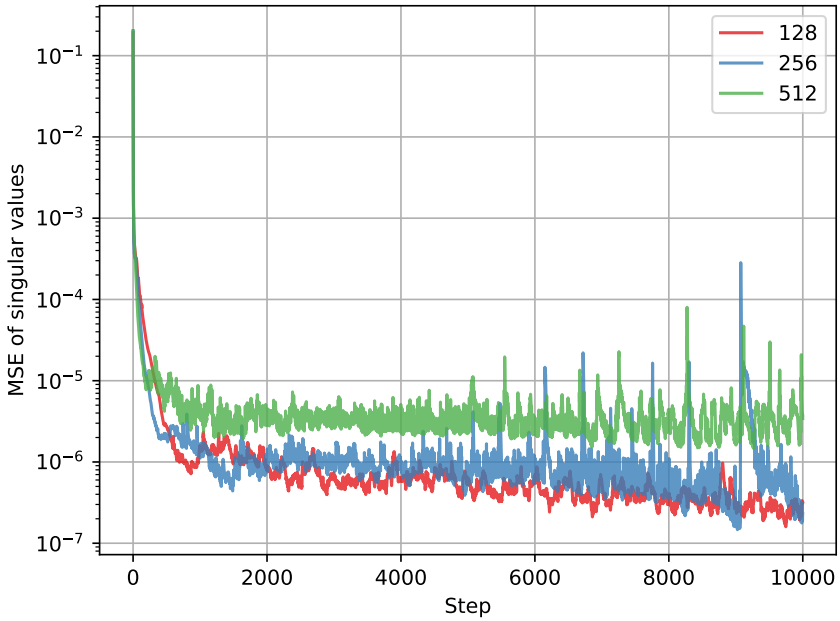


Figure 5: The convergence curve of the singular values.

In this experiment, we present the convergence analysis of our amortized eigendecomposition approach concerning the accurate estimation of singular values. We tackle the nuclear norm regularization problem outlined in Eq. (17). In each iteration, we compare the diagonal elements of $\boldsymbol{U}^\top \boldsymbol{\theta}^\top \boldsymbol{\theta} \boldsymbol{U}$ with the exact singular values of $\boldsymbol{\theta}$. We conducted tests on $\boldsymbol{\theta}$ of sizes 128×128, 256×256 and 512×512. The mean square error (MSE) is depicted in Figure 5.

The results indicate that initially, the singular values obtained by our approach do not match the correct singular values within a small error. However, within just a few iterations, our approach rapidly converges to the correct values, demonstrating the effectiveness of the loss function in accurately estimating singular values.

A.2 Latent-space Principle Component Analysis

In this task, we evaluate the latent-space PCA method as outlined in Eq. (19) using the MNIST dataset. The architecture for both the encoder and decoder mirrors that of the nuclear norm regularization model, with the exception that there is a linear projection $\boldsymbol{U}\boldsymbol{U}^\top$ aligning with the direction of the principal components. We aim to extract the first two principal components in the latent space.

The experimental outcomes are depicted in Figure 6. Specifically, Figure 6 (a) displays the convergence trajectories for both the reconstruction loss and the eigenvalue loss as defined in Eq. (19). It is observed that the reconstruction loss curves for the conventional eigh function and our amortized eigendecomposition strategy are nearly indistinguishable. However, for the eigen-loss, our method initially registers lower values compared to the eigh function but eventually converges to equivalent values. This demonstrates the efficacy of the amortized optimization approach.

Figure 6(b) illustrates the distribution of features in the latent space along the two principal components for cases where $\eta = 0$ and $\eta = 1$. The eigenvalues correspond to the total variance across the feature dimensions. Without regularization (when $\eta = 0$), the scales of the principal component axes

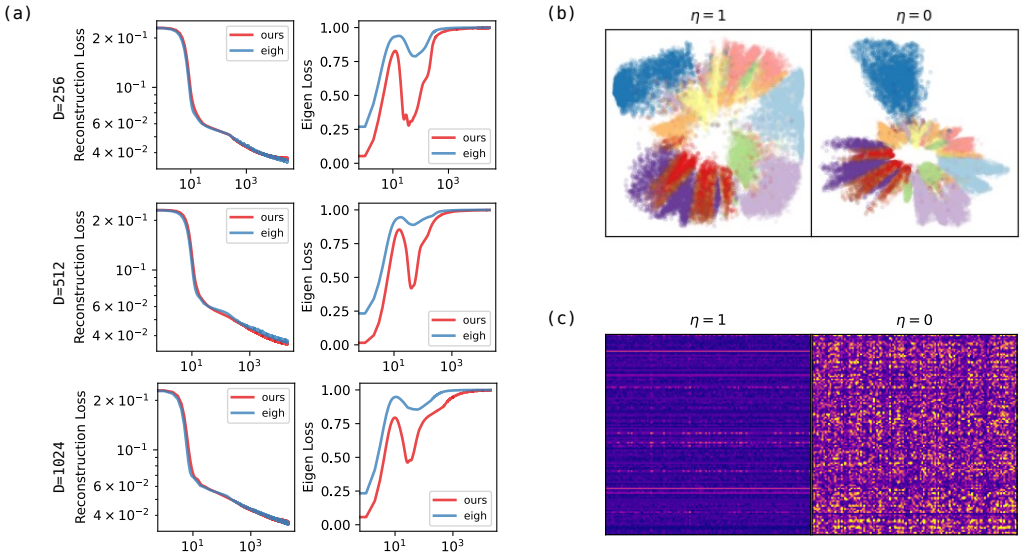


Figure 6: Experimental results of Latent-space PCA on MNIST dataset. (a) Convergence curves. First column: Convergence curve of reconstruction loss. Second column: Convergence curve of eigen loss. (b) Principle components in latent space: The two principle components of features in latent space for $\eta = 0$ and $\eta = 1$. (c) The sparsity of the network structure. The weight matrices of the second layer of the encoder. The color indicates the scale of the absolute values of the weight matrix ranging from 0 to 1.

can vary significantly. However, when regularization is applied (with $\eta = 0$), the distribution of the embedding becomes more uniform, indicating a more even spread across the principal components.

The regularization applied to the eigenvalues also influences the sparsity of the network's architecture. This effect is depicted in Figure 6(c), which shows the weight matrix of the last layer in the encoder. When $\eta = 0$, the network prioritizes learning weights that optimally reconstruct the images, resulting in a relatively dense weight configuration. Conversely, when $\eta = 0$, the eigenvalues corresponding to the principal components are encouraged to concentrate on the first few, leading to a decline in variance for the subsequent components. This, in turn, promotes sparsity in the weight matrix, as the network assigns less importance to the less variable components.

A.3 Adversarial Attacks on Graph Convolutional Networks

This study investigates the robustness of graph convolutional networks (GCNs) [21] by conducting adversarial attacks on graph structures. These attacks involve perturbing the graph structure by randomly adding or removing edges in the adjacency matrix. To measure the magnitude of these perturbations, we introduce a *contamination rate*, which represents the ratio of the number of modified edges to the total number of node pairs.

For each adversarial scenario, we commence by randomly determining a *contamination rate* within the range of [0, 0.02] for the Cora and Citeseer datasets, and [0, 0.001] for the PubMed dataset. This rate dictates the proportion of edges to be randomly added or removed from the graph. A greater contamination rate signifies a more severe attack. The nodes are partitioned randomly into training, validation, and test sets with respective proportions of 60%, 20%, and 20%. The classification accuracy on the test set is then recorded.

The robustness of the GCN when subjected to graph contamination is illustrated in Figure 7. Each data point on the graph corresponds to a specific attack instance and its resultant test accuracy. Additionally, we have applied a polynomial regression with a 5th-degree basis to model the general trend of the relationship between the contamination rate and the test accuracy. The findings indicate that the graph convolutional network exhibits enhanced robustness when convolutions are applied to graph signals of a lower rank.

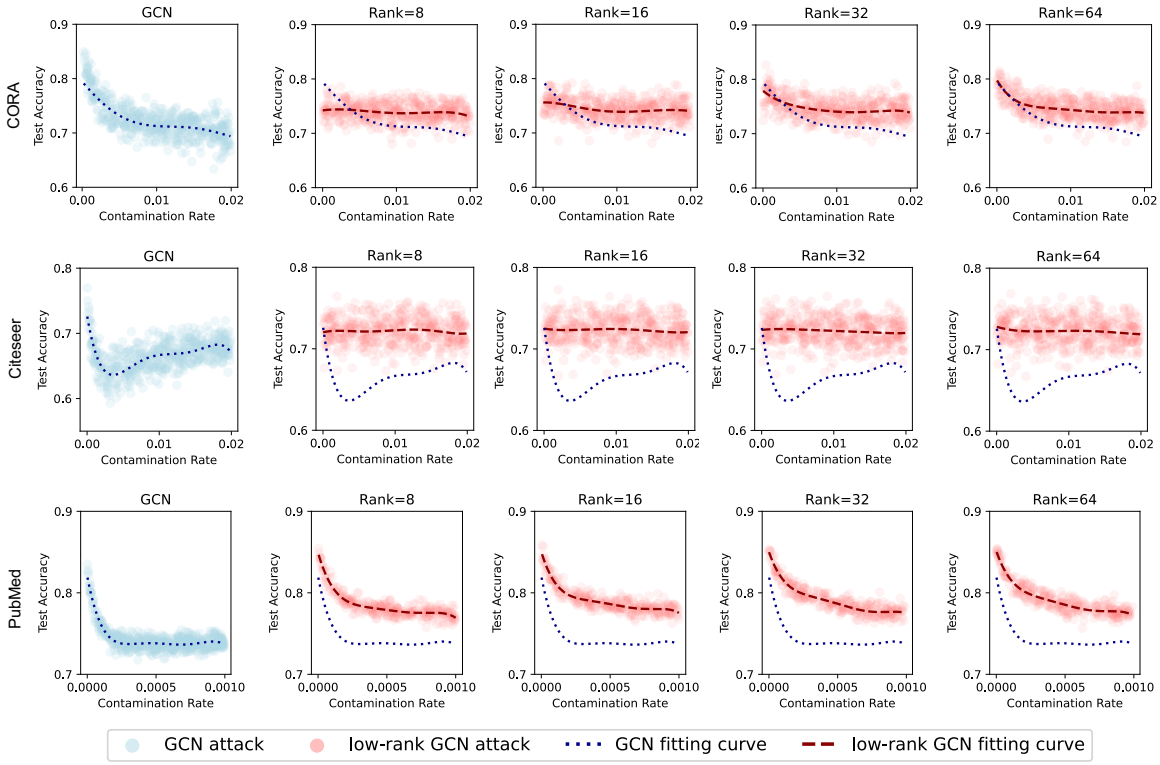


Figure 7: Experimental results of graph structure adversarial attacking

Appendix B Proofs

In this section, we provide the proofs for our theoretical findings.

B.1 Preliminaries

Lemma 1 (Jensen's inequality for convex functions)**.** *For a convex function f, given n real numbers $x_1, \ldots, x_n$ and non-negative weights $w_1, \ldots, w_n$ such that $\sum_{i=1}^{n} w_i = 1$, Jensen's inequality is expressed as:*

$$f\left(\sum_{i=1}^{n} w_i x_i\right) \geqslant \sum_{i=1}^{n} w_i f\left(x_i\right). \tag{21}$$

The equality holds if and only if one of the following conditions is satisfied:

1. *f is linear;*
2. *x_i's are equal;*
3. *one of w_i's is one and the rest are zero.*

B.2 Proof of Theorem 1

Theorem 1 (Trace inequality with weight matrix) *Consider a n-dimensional symmetric matrix $\boldsymbol{A}$. Let $\boldsymbol{M}$ be a k-dimensional diagonal matrix with elements $0 < m_1 < m_2 \cdots < m_k$. For an arbitrary matrix $\boldsymbol{U} \in \mathbb{U}^{n\times k}$ with orthogonal columns, the following inequality always holds:*

$$\sum_{i=1}^{k} m_{k-i+1}\lambda_i \leqslant \mathsf{tr}(\boldsymbol{M}\boldsymbol{U}^{\mathsf{T}}\boldsymbol{A}\boldsymbol{U}) \leqslant \sum_{i=1}^{k} m_i\lambda_i. \tag{22}$$

The equalities are achieved if and only if $\boldsymbol{U}$ contains the eigenvectors corresponding to the k largest (smallest) eigenvalues of $\boldsymbol{A}$.

Proof. We can consider the case where $\boldsymbol{A}$ has all nonnegative eigenvalues $\varepsilon_j \geqslant 0$, without loss of generality, as we can always shift the diagonal elements of $\boldsymbol{A}$ to meet this condition:

$$\mathsf{tr}\left(\boldsymbol{M}\boldsymbol{U}^{\mathsf{T}}\boldsymbol{A}\boldsymbol{U}\right) = \mathsf{tr}\left(\boldsymbol{M}\boldsymbol{U}^{\mathsf{T}}\left(\boldsymbol{A}+\alpha\boldsymbol{I}\right)\boldsymbol{U}\right) - \alpha\sum_{i=1}^{k} m_i \tag{23}$$

Let u_{ij} be the i,j-th entry of $\boldsymbol{U}$. The Right-hand-side (RHS) of the leftmost inequality can be expressed as,

$$\text{RHS} = \mathsf{tr}(\boldsymbol{M}\boldsymbol{U}^{\mathsf{T}}\boldsymbol{A}\boldsymbol{U}) = \sum_{i=1}^{k}\sum_{l=1}^{n}\sum_{j=1}^{n} m_i a_{jl} u_{ji} u_{li} \tag{24}$$

We denote the actual eigenvectors of $\boldsymbol{A}$ as $\boldsymbol{V}$ with entries $\{\nu_{jl}\}$, and $\{\lambda_s\}, s = 1, \ldots, n$ as the eigenvalues, we have:

$$a_{jl} = \sum_{s=1}^{n} \lambda_s \nu_{js} \nu_{ls} \tag{25}$$

Substituting this into (24), we obtain:

$$\text{RHS} = \sum_{i=1}^{k}\sum_{l=1}^{n}\sum_{j=1}^{n} m_i \left(\sum_{s=1}^{n} \lambda_s \nu_{js}\nu_{ls}\right) u_{ji}u_{li} \tag{26}$$

$$= \sum_{i=1}^{k}\sum_{s=1}^{n} m_i\lambda_s \sum_{j=1}^{n} \nu_{js}u_{ji} \sum_{l=1}^{n} \nu_{ls}u_{li} \tag{27}$$

$$= \sum_{i=1}^{k}\sum_{s=1}^{n} m_i\lambda_s \left(\boldsymbol{\nu}_s^{\mathsf{T}}\boldsymbol{u}_i\right)\boldsymbol{\nu}_s^{\mathsf{T}}\boldsymbol{u}_i \tag{28}$$

We define

$$g_{si} = \boldsymbol{\nu}_s^\mathsf{T} \boldsymbol{u}_i \tag{29}$$

then the RHS becomes,

$$\text{RHS} = \sum_{i=1}^{k} \sum_{s=1}^{n} m_i \lambda_s g_{si}^2 \tag{30}$$

and,

$$0 \leqslant g_{si}^2 \leqslant 1. \tag{31}$$

It's worth noting that for any s,

$$\sum_{s=1}^{n} g_{si}^2 = \sum_{s=1}^{n} \boldsymbol{u}_i^\mathsf{T} \boldsymbol{\nu}_s \boldsymbol{\nu}_s^\mathsf{T} \boldsymbol{u}_i \tag{32}$$

$$= \mathsf{tr}\left(\boldsymbol{u}_i^\mathsf{T} \boldsymbol{V} \boldsymbol{V}^\mathsf{T} \boldsymbol{u}_i\right) = 1 \tag{33}$$

Thus we have,

$$\text{RHS} = \sum_{i=1}^{k} m_i \left(\sum_{s=1}^{n} \lambda_s g_{si}^2\right) \left(\sum_{s=1}^{n} g_{si}^2\right) \tag{34}$$

Applying the Cauchy-Schwartz inequality, we get,

$$\text{RHS} \geqslant \sum_{i=1}^{k} m_i \left(\sum_{s=1}^{n} \sqrt{\lambda_s} g_{si}^2\right)^2 \tag{35}$$

The equality holds if and only if for all s and a fixed l,

$$\frac{\lambda_s g_{si}^2}{g_{si}^2} = \text{const.} \tag{36}$$

or

$$g_{si}^2 = 0 \tag{37}$$

This is only true when g_{si} are either 0 or 1, with at most one non-zero value for each s and i. In other words, $\boldsymbol{G} = \{g_{si}\}$ must be a permutation matrix. This implies that $\boldsymbol{U}$ is equivalent to the eigenvectors of $\boldsymbol{V}$, albeit with a possible permutation.

Thus this inequality reaches a minimum if we rearrange the indices of the eigenvalues:

$$\text{RHS} \geqslant \sum_{i=1}^{k} m_i \lambda_{\sigma(i)} \geqslant \sum_{i=1}^{k} m_{k-i+1} \lambda_i \tag{38}$$

The last inequality is a result of the rearrangement inequality. This completes the proof of the first inequality.

For the upper bound of Eq. (22), we apply the Cauchy-Schwartz inequality to Eq. (30) and obtain,

$$\text{LHS} \leqslant \sum_{i=1}^{k} m_i \sqrt{\sum_{s=1}^{n} \lambda_s^2 g_{si}^2 \sum_{s=1}^{n} g_{si}^2} \tag{39}$$

$$= \sum_{i=1}^{k} m_i \sqrt{\sum_{s=1}^{n} \lambda_s^2 g_{si}^2} \tag{40}$$

$$\leqslant \sum_{i=1}^{k} m_i \lambda_{\sigma(s)}. \tag{41}$$

Here again, the Cauchy-Schwartz inequality achieves the maximum when $\boldsymbol{U}$ is equivalent to the eigenvectors of $\boldsymbol{V}$, up to a possible permutation. By applying the rearrangement inequality, we further obtain the final bound:

$$\text{LHS} \leqslant \sum_{i=1}^{k} m_i \lambda_i. \tag{42}$$

This yields the desired upper bound for the second inequality of the main result. □

B.3 Proof of Theorem 2

Theorem 2 (Trace inequality with convex function) *Let $\boldsymbol{A}$ be a given n-dimensional symmetric matrix and let $\boldsymbol{U}$ be a matrix of size $n \times k$ that resides on the Stiefel manifold. Suppose $f : \mathbb{R} \to \mathbb{R}$ be a monotonically increasing, convex function applied element-wise. The following inequalities hold:*

$$k\left(f\left(\frac{1}{k}\mathsf{tr}(\boldsymbol{A})\right)\right) \leqslant \mathsf{tr}\Big(f\big(\boldsymbol{U}^{\mathsf{T}}\boldsymbol{A}\boldsymbol{U}\big)\Big) \leqslant \sum_{i=1}^{k} f(\lambda_i), \tag{43}$$

The rightmost inequality becomes an equality if and only if $\boldsymbol{U}$ comprises the eigenvectors of $\boldsymbol{A}$ that correspond to the k largest eigenvalues, The leftmost inequality is met with equality when $\boldsymbol{U}$ is such that all diagonal elements of the matrix $\boldsymbol{U}^{\mathsf{T}}\boldsymbol{A}\boldsymbol{U}$ are equal.

Proof. **The rightmost inequality** We first focus on the rightmost inequality, which provides an upper bound for the convex trace loss.

To establish the rightmost inequality, let us consider the left-hand side (LHS) of the equation. Let's denote the diagonal elements of $\boldsymbol{A}$ as $\{a_{jj}\}$. Consider the eigendecomposition of $\boldsymbol{A}$ as in Eq. (25), we can rewrite a_{jj} in terms of the eigenvalues and eigenvectors:

$$a_{jl} = \sum_{i=1}^{k} \lambda_i \nu_{ji} \nu_{li} \tag{44}$$

where ν_{ji} is the (j, i) entry of its eigenvectors $\boldsymbol{V}$. The diagonal elements of the matrix product $\boldsymbol{U}^{\mathsf{T}}\boldsymbol{A}\boldsymbol{U}$ can then be written as:

$$\Big(\boldsymbol{U}^{\mathsf{T}}\boldsymbol{A}\boldsymbol{U}\Big)_{ss} = \sum_{l=1}^{n}\sum_{j=1}^{n} a_{jl} u_{ls} u_{js} \tag{45}$$

Substituting this result back into the LHS, we have

$$\begin{aligned}
RHS &= \sum_{s=1}^{n} f\left(\Big(\boldsymbol{U}^{\mathsf{T}}\boldsymbol{A}\boldsymbol{U}\Big)_{ss}\right) && (46)\\
&= \sum_{s=1}^{n} f\left(\sum_{l=1}^{n}\sum_{j=1}^{n} a_{jl} u_{ls} u_{js}\right) && (47)\\
&= \sum_{s=1}^{n} f\left(\sum_{l=1}^{n}\sum_{j=1}^{n}\sum_{i=1}^{k} \lambda_i \nu_{ji} \nu_{li} u_{ls} u_{js}\right) && (48)\\
&= \sum_{s=1}^{n} f\left(\sum_{i=1}^{k} \lambda_i \sum_{l=1}^{n}\sum_{j=1}^{n} \nu_{ji} \nu_{li} u_{ls} u_{js}\right) && (49)
\end{aligned}$$

We can express the summation of the elements in vector form as follows:

$$\begin{aligned}
\sum_{i=1}^{k}\sum_{l=1}^{n}\sum_{j=1}^{n} \nu_{ji} \nu_{li} u_{ls} u_{js} &= \sum_{i=1}^{k} \boldsymbol{u}_s^{\mathsf{T}} \boldsymbol{v}_i \boldsymbol{v}_i^{\mathsf{T}} \boldsymbol{u}_s && (50)\\
&= \mathsf{tr}(\boldsymbol{u}_s^{\mathsf{T}} \boldsymbol{V}\boldsymbol{V}^{\mathsf{T}} \boldsymbol{u}_s) = 1 && (51)
\end{aligned}$$

For all indices j, l and s, all the addends satisfy:

$$0 \leqslant \sum_{l=1}^{n}\sum_{j=1}^{n} \nu_{ji}\nu_{li}u_{ls}u_{js} \leqslant 1, \tag{52}$$

which allows us to treat $\sum_{l=1}^{n}\sum_{j=1}^{n} \nu_{ji}\nu_{li}u_{ls}u_{js}$ as a set of weights. By applying Jensen's inequality to the equation previously labeled as Eq. (49), we obtain:

$$LHS \leqslant \sum_{i=1}^{k} f\left(\lambda_i\right) \sum_{s=1}^{n}\sum_{l=1}^{n}\sum_{j=1}^{n} \nu_{ji}\nu_{li}u_{ls}u_{js} \tag{53}$$

Equality holds if and only if all the eigenvalues $\{\lambda_i\}$ are identical, which is generally not the case for an arbitrary symmetric matrix $\boldsymbol{A}$, or for each index s, there exists an index i such that,

$$(\boldsymbol{u}_s^{\mathsf{T}}\boldsymbol{v}_i)^2 = 1, \tag{54}$$

with all other terms being zero. Therefore, $\boldsymbol{U}$ must be a permutation of $\boldsymbol{V}$:

$$\boldsymbol{u}_s = \boldsymbol{v}_{\sigma(i)} \tag{55}$$

Substituting this result back into Eq. (53), we obtain:

$$\sum_{i=1}^{k} f\left(\lambda_i\right) \sum_{s=1}^{n}\sum_{l=1}^{n}\sum_{j=1}^{n} \nu_{ji}\nu_{li}u_{ls}u_{js} \leqslant \sum_{i=1}^{k} f\left(\lambda_{\sigma(i)}\right) \leqslant \sum_{i=1}^{k} f\left(\lambda_i\right) \tag{56}$$

With this, we conclude the derivation of the first inequality.

The leftmost inequality. For the leftmost inequality,

$$\begin{aligned}
RHS &= \sum_{i=1}^{k} f\left(\sum_{l=1}^{n}\sum_{j=1}^{n} a_{jl}u_{li}u_{ji}\right) && (57)\\
&= k\sum_{i=1}^{k} \frac{1}{k} f\left(\sum_{l=1}^{n}\sum_{j=1}^{n} a_{jl}u_{li}u_{ji}\right) && (58)\\
&\geqslant kf\left(\frac{1}{k}\sum_{i=1}^{k}\sum_{l=1}^{n}\sum_{j=1}^{n} a_{jl}u_{li}u_{ji}\right) \qquad \text{(Jensen)} && (59)\\
&= kf\left(\frac{1}{k}\mathsf{tr}\left(\boldsymbol{U}^{\mathsf{T}}\boldsymbol{A}\boldsymbol{U}\right)\right) && (60)\\
&= kf\left(\frac{1}{k}\mathsf{tr}\left(\boldsymbol{A}\right)\right) && (61)
\end{aligned}$$

the equality holds if and only if $\sum_j u_{ij}^* u_{ij} a_{jj}$ equals for all i. □

A similar result for finding the k smallest eigenvalues can be obtained. The next corollary states the results

Corollary (Trace inequality with concave function) *Let $\boldsymbol{A}$ be a given n-dimensional symmetric matrix and let $\boldsymbol{U}$ be a matrix of size $n \times k$ that resides on the Stiefel manifold. Suppose $f : \mathbb{R} \to \mathbb{R}$ be a monotonically increasing, concave function applied element-wise. The following inequalities hold:*

$$\sum_{i=1}^{k} f(\lambda_i) \leqslant \mathsf{tr}\Big(f(\boldsymbol{U}^{\mathsf{T}}\boldsymbol{A}\boldsymbol{U})\Big) \leqslant k\left(f\left(\frac{1}{k}\mathsf{tr}(\boldsymbol{A})\right)\right), \tag{62}$$

The rightmost inequality becomes an equality if and only if $\boldsymbol{U}$ comprises the eigenvectors of $\boldsymbol{A}$ that correspond to the k smallest eigenvalues.

Appendix C Further Extensions

C.1 Related Work

Eigensolver in numerical linear algebra Large-scale eigensolvers in numerical linear algebra usually involve two stages [39]. In the first stage, depending on whether the matrix is symmetric or not, either Arnoldi or Lanczos iteration is applied to reduce it into a matrix of smaller scale. Then, classical iteration methods such as power, inverse, and Rayleigh quotient iterations are applied to extract the eigensystem of the small-scale matrix. Compared to iterative solvers, our method is more easy to be integrated into the network training procedure.

Eigendecomposition via manifold optimization In the optimization community, especially manifold optimization [1], identifying the eigen-subspace or solving for eigendecomposition is often achieved by optimizing the original trace loss function in Eq. 6 or the Brockett's cost function in Eq. 8 respectively. Instead of parametrizing the matrix $\boldsymbol{U}$ via QR decomposition, one usually solves the constraint optimization via gradient descent with retraction. Namely, in each iteration, one firstly performs one step of gradient descent from a $\boldsymbol{U}$ on the Stiefel manifold and then uses a retraction to correct it back to the manifold. Some convergence guarantees to the critical point and linear convergence rate can be established under certain assumptions [1, 43]. Various methods exist, such as [24] introducing the splitting method and [43] proposing a Crank-Nicolson-like update scheme to preserve the orthogonal constraints. No matter which method is used, the computation bottleneck lies in preserving the orthogonal constraints similar to our amortized method.

Spectral normalization Spectral normalization is frequently added to improve the generalization ability of the neural network, i.e. CNN [11, 19] and GAN [32]. Compared to nuclear-norm regularization, spectral regularization only penalizes the largest absolute value of the eigenvalue so that the network outputs depend continuously on the input. Therefore no orthogonal constraint is enforced and the spectral norm can be more efficiently estimated based on the power iteration on a randomly initialized vector as in [32, 11].

C.2 Extensions on the Amortized Eigendecomposition Approach

Singular Value Decomposition Problem The singular value decomposition of a rectangular matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ can be achieved by solving the following objective:

$$\max_{\boldsymbol{U}\in\mathbb{U}^{n\times k}} \|\boldsymbol{A}\boldsymbol{U}\boldsymbol{M}\|_{\mathrm{F}} \tag{63}$$

where $\|\cdot\|_{\mathrm{F}}$ represent the Frobenius norm. The i-th largest singular values can be obtained by $\sigma_i = \|\boldsymbol{A}\boldsymbol{u}_i\|_2$ with $\boldsymbol{u}_i$ the i-th column of optimal $\boldsymbol{U}$.

Generalized Eigendecomposition Problem Generalized eigendecomposition is an extension of conventional eigendecomposition to a pair of matrices, rather than just a single matrix. The generalized eigendecomposition can be formulated as follows:

$$\boldsymbol{A}\boldsymbol{u} = \varepsilon\boldsymbol{B}\boldsymbol{u} \tag{64}$$

where $\boldsymbol{A}$ and $\boldsymbol{B}$ are square matrices of same dimensions in the complex field $\mathbb{C}^{n\times n}$. The vector $\boldsymbol{u}$ represents the eigenvector, and the scalar ε denotes the corresponding eigenvalue. Together, $(\varepsilon, \boldsymbol{u})$ constitute the eigenpair of the generalized eigenvalue problem of the matrix pair $(\boldsymbol{A}, \boldsymbol{B})$. In a manner akin to prior problems, the resolution to this generalized version can be achieved through the optimization of the following objective function:

$$\max_{\boldsymbol{U}\in\mathbb{U}^{n\times k}} \sum_{i=1}^{k} m_i \frac{\boldsymbol{u}_i^{\mathsf{T}}\boldsymbol{A}\boldsymbol{u}_i}{\boldsymbol{u}_i^{\mathsf{T}}\boldsymbol{B}\boldsymbol{u}_i}. \tag{65}$$

where m_i's are the weights as defined previously and $\boldsymbol{u}_i$'s are the orthonormal column vectors of $\boldsymbol{U}$.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims consist of theoretical analysis of the loss functions which is stated in Thm. 1 and Thm. 2 and verifications of our method in numerical experiments are done in Sec. 5.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We show in the discussion section.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: All the assumptions are stated in Thm. 1 and Thm. 2 respectively and the proof is put in Sec. B.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: This is included in this paper in Sec. 5.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: Our code will be released upon acceptance.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: This is included in this paper in the Experiment section and Appendix A.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: The statistical significance may not be the primary focus. The evaluation and validation of our method rely on other metrics, such as accuracy and convergence rate.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We have indicated the type of GPU in the Experiment section.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: this paper is conducted with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: This paper is a pure theoretic paper instead of an application, thus no societal impacts is involved.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [No]

    Justification: Not relevant to our research.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have done this. CC-BY 4.0

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We have done this.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.