Operator SVD with Neural Networks via Nested Low-Rank Approximation

J. Jon Ryu [1] **Xiangxiang Xu** [1] **H. S. Melihcan Erol** [1] **Yuheng Bu** [2] **Lizhong Zheng** [1] **Gregory W. Wornell** [1]

Abstract

Computing eigenvalue decomposition (EVD) of a given linear operator, or finding its leading eigenvalues and eigenfunctions, is a fundamental task in many machine learning and scientific computing problems. For high-dimensional eigenvalue problems, training neural networks to parameterize the eigenfunctions is considered as a promising alternative to the classical numerical linear algebra techniques. This paper proposes a new optimization framework based on the low-rank approximation characterization of a truncated singular value decomposition, accompanied by new techniques called *nesting* for learning the top-L singular values and singular functions in the correct order. The proposed method promotes the desired orthogonality in the learned functions implicitly and efficiently via an unconstrained optimization formulation, which is easy to solve with off-the-shelf gradient-based optimization algorithms. We demonstrate the effectiveness of the proposed optimization framework for use cases in computational physics and machine learning.

1. Introduction

Spectral decomposition techniques, including singular value decomposition (SVD) and eigenvalue decomposition (EVD), are crucial tools in machine learning and data science for handling large datasets and reducing their dimensionality while preserving prominent structures; see, e.g., (Markovsky, 2012; Blum et al., 2020). They break down a matrix (or a linear operator) into its constituent parts, enabling a better understanding of the underlying geometry and relationships within the data. These form the foundation of various low-dimensional embedding algorithms (Schölkopf et al., 1998; Tenenbaum et al., 2000; Roweis & Saul, 2000; Shi & Malik, 2000; Ng et al., 2001; Belkin & Niyogi, 2003; Bengio et al., 2003; Cox & Cox, 2008) and correlation analysis algorithms (Michaeli et al., 2016; Wang et al., 2019) and are widely used in image and signal processing (Andrews & Patterson, 1976; Turk & Pentland, 1991; Wiskott & Sejnowski, 2002; Sprekeler, 2011; Scetbon et al., 2021), natural language processing (Landauer et al., 1998; Goldberg & Levy, 2014), among other fields. Beyond machine learning applications, solving eigenvalue problems is a crucial step in solving partial differential equations (PDEs), such as Schrödinger's equations in quantum chemistry (Hermann et al., 2020; Pfau et al., 2020).

The standard approach to these problems in practice is to perform the matrix spectral decomposition using the standard techniques from numerical linear algebra (Golub & Van Loan, 2013). In machine learning, the size of the matrix is typically given by the size of the data sample or the dimensionality of data. In physical simulation, the underlying matrix scales with the resolution of discretization of a given domain. For finding a few top (or bottom) eigenmodes, in general, iterative subspace methods such as Krylov subspace methods (Saad, 1981) and LOBPCG (Knyazev, 2001) can efficiently find top eigenmodes via repeating matrix-vector products (Golub & Van Loan, 2013). Note that the full eigendecomposition of a $N \times N$ matrix can be performed in $O(N^3)$ time complexity if the matrix can be stored in memory. For large-scale, high-dimensional data, however, the memory, computational, and statistical complexity of matrix decomposition algorithms poses a significant challenge in practice. As the data size (or the resolution of the grid in physical simulation) or the dimensionality of the underlying problem increases, the matrix-based approach becomes easily infeasible as even storing the eigenvectors in memory is too costly.

A promising alternative is to approximate the singular- or eigen-functions using parametric function approximators, assuming that there exists an abstract operator that induces a target matrix to decompose. In other words, we aim to approximate an eigenvector $\hat{\boldsymbol{\phi}}_\ell \in \mathbb{R}^N$ by a single parametric function $\hat{\phi}_\ell \colon \mathcal{X} \to \mathbb{R}$. In Fig. 1, we illustrate the proposed framework NeuralSVD, which is a special instance of the parametric approach, as a schematic diagram.

Compared to the "nonparametric" approach, the parametric approach has several advantages. First, unlike

[1]Department of EECS, MIT, Cambridge, Massachusetts, USA [2]Department of ECE, University of Florida, Gainesville, Florida, USA. Correspondence to: J. Jon Ryu <jongha@mit.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

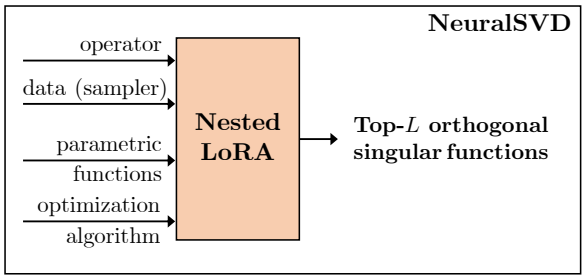


Figure 1: Schematic illustration of NeuralSVD.

the nonparametric approach which relies on the Nyström method (Williams & Seeger, 2000; Bengio et al., 2004) to extrapolate eigenvectors to unseen points, the parametric eigenfunctions can naturally extrapolate without the storage and computational complexity of Nyström; refer to Sec. A.1 for a detailed discussion. Second, given the exceptional ability of neural networks (NNs) to generalize with complex data, such as convolutional neural networks for images, transformers for natural language, and recently developed NN ansatzes for quantum chemistry (Hermann et al., 2020; Pfau et al., 2020), one can anticipate better extrapolation performance than in the nonparametric, matrix approach. If the choice of parametric functions is appropriate to exploit the complex structure of the underlying distribution, we can also expect the parametric approach to scale better in terms of training complexity for large-scale problems than the nonparametric counterpart. Third, in the context of solving PDEs, the parametric approach stands out, notably because it necessitates only a sampler from a specified domain without the need for discretization. This is particularly advantageous as it helps mitigate the potential introduction of undesirable approximation errors.

In this paper, we propose a new optimization framework that can train neural networks to approximate the top-L orthogonal singular- (or eigen-) functions of an operator. The proposed method is based on an unconstrained optimization problem from Schmidt's low-rank approximation theorem (1907) that naturally admits an unbiased gradient estimator. To learn the ordered top-L orthogonal singular basis as the optimal solution simultaneously, we introduce new techniques called *nesting* to break the symmetry so that we can learn the singular functions in the order of singular values; see the high-level illustrations in Fig. 2.

While several frameworks have been proposed in the machine learning community to systematically recover ordered eigenfunctions using neural networks (Pfau et al., 2019; Deng et al., 2022b), these approaches encounter practical optimization challenges, particularly in enforcing the orthonormality of the learned eigenfunctions. Compared to the prior works, our framework can (1) learn the top-L orthogonal singular bases more efficiently for larger L due to the more stable optimization procedure, and (2) perform SVD of a non-self-adjoint operator by design, handling EVD of a self-adjoint operator as a special case. We demonstrate the power of our framework in solving PDEs and representation learning for cross-domain retrieval.

2. Problem Setting and Preliminaries

2.1. Operator SVD

While SVD is typically assumed to be done via EVD, our low-rank approximation framework can directly perform SVD, handling EVD as a special case. We consider two separable Hilbert spaces $\mathcal{F}$ and $\mathcal{G}$ and a linear operator $\mathcal{T}\colon \mathcal{F} \to \mathcal{G}$. We will use the bra-ket notation, which denotes $|f\rangle$ for a function $f(\cdot)$ throughout, as it allows us to describe the proposed method in a succinct way. For most applications, the Hilbert spaces $\mathcal{F}$ and $\mathcal{G}$ are $\mathscr{L}^2$ spaces of square-integrable functions, and a reader thus can read the inner product between two real-valued functions $\langle f|f'\rangle$ as an integral $\int_{\mathcal{X}} f(x)f'(x)\mu(\mathrm{d}x)$ for some underlying measure μ over a domain $\mathcal{X}$. In learning problems, $\mathcal{T}$ is typically an integral kernel operator induced by a kernel function, accompanied by data distributions as the underlying measures. In solving PDEs, $\mathcal{T}$ is given as a differential operator that governs a physical system of interest, where the underlying measure is the Lebesgue measure over a domain.

For a *compact* operator $\mathcal{T}$, it is well known that there exist orthonormal bases $(\phi_i)_{i\geq 1}$ and $(\psi_i)_{i\geq 1}$ with a sequence of non-increasing, non-negative real numbers $(\sigma_i)_{i\geq 1}$ such that $(\mathcal{T}\phi_i)(y) = \sigma_i\psi_i(y)$, $(\mathcal{T}^*\psi_i)(x) = \sigma_i\phi_i(x)$, $i = 1, 2, \ldots$.[1] The function pairs (ϕ_i, ψ_i) are called (left- and right-, resp.) singular functions corresponding to the singular value σ_i. Hence, the compact operator $\mathcal{T}$ can be written as

$$\mathcal{T} = \sum_{i=1}^{\infty} \sigma_i |\psi_i\rangle\langle\phi_i|, \tag{1}$$

for $\sigma_1 \geq \sigma_2 \geq \ldots \geq 0$, which we call the SVD of $\mathcal{T}$. Here, $|\psi\rangle\langle\phi|\colon \mathcal{F} \to \mathcal{G}$ is the operator defined as $(|\psi\rangle\langle\phi|)|f\rangle := (\langle\phi|f\rangle)|\psi\rangle$, which can be understood as the *outer product*. We refer an interested reader to (Weidmann, 2012, Theorem 7.6) for a rigorous treatment of SVD of compact operators.

2.2. EVD as a Special Case of SVD

In several applications, the operator is self-adjoint (i.e., $\mathcal{T}^* = \mathcal{T}$ with $\mathcal{F} = \mathcal{G}$), and sometimes even positive definite (PD). By the spectral theorem (Weidmann, 2012, Theorem 7.1), a compact self-adjoint operator has the EVD of the form $\mathcal{T} = \sum_{i=1}^{\infty} \lambda_i |\phi_i\rangle\langle\phi_i|$. In this case, the singular values

[1] Compact operators can be informally understood as a benign class of possibly infinite-dimensional operators that behave similarly to finite-dimensional matrices, so that we can consider the notion of SVD as in matrices. A formal definition is not crucial in understanding the manuscript and is thus deferred to Sec. C.1.

of the operator are the absolute values of its eigenvalues, and for each i, the i-th left- and right- singular functions are either identical (if $\lambda_i \geq 0$) or only different by the sign (if $\lambda_i < 0$). Hence, in particular, we can find its EVD by SVD in the case of a positive-definite (PD) operator. We remark in passing that our framework is also applicable for a certain class of non-compact operators; see Sec. 4.1 and Sec. C.6.

2.3. SpIN and NeuralEF

As alluded to earlier, Spectral Inference Networks (SpIN) (Pfau et al., 2019) and Neural Eigenfunctions (NeuralEF) (Deng et al., 2022b) are the most closely related prior works to ours, in the sense that these methods aim to *learn* the top-L orthonormal eigenbasis of a self-adjoint operator by *training* parametric functions. Though there exist other approaches in computational physics that aim to find beyond the top mode or ground state, most, if not all, approaches are based on rather ad-hoc regularization terms and do not have guarantee to recover the top-L ordered eigenfunctions. Hence, we briefly overview SpIN and NeuralEF in the main text, and discuss the two methods in greater details as well as the other line of works in Sec. B.

SpIN and NeuralEF are only applicable for self-adjoint operators, and thus we temporarily assume a self-adjoint operator $\mathcal{T}\colon \mathcal{F} \to \mathcal{F}$ in the rest of this section. SpIN and NeuralEF are both grounded in the principle of maximizing the Rayleigh quotient with orthonormality constraints. However, their optimization frameworks encounter nontrivial complexity issues, as summarized in Table 2. The primary challenge lies in efficiently handling these orthonormality constraints. To achieve fast convergence with off-the-shelf gradient-based optimization algorithms, it is also crucial to estimate gradients in an unbiased manner.

SpIN starts from the following variational characterization of the top-L orthonormal eigenbasis:

$$\begin{aligned} &\underset{\tilde{\phi}_\ell \in \mathcal{F},\, \ell \in [L]}{\text{maximize}} \ \langle \tilde{\phi}_\ell | \mathcal{T} \tilde{\phi}_\ell \rangle \\ &\text{subject to } \ \langle \tilde{\phi}_i | \tilde{\phi}_{i'} \rangle = \delta_{ii'} \quad \forall 1 \leq i, i' \leq L. \end{aligned} \tag{2}$$

Since this formulation only captures the subspace without order, SpIN employs a special gradient masking scheme to learn the eigenfunctions in the correct order. The resulting algorithm involves Cholesky decomposition of $L \times L$ matrix per iteration, which takes $O(L^3)$ complexity in general. Further, to come up with unbiased gradient estimates, SpIN introduces a hyperparameter-sensitive bi-level optimization and necessitates the need to store the Jacobian of the parametric model. As a result, the unfavorable scalability with L, along with memory complexity and implementation challenges, reduces the practical utility of SpIN.

To circumvent the issues with SpIN, NeuralEF adopted and extended an optimization framework of EigenGame (Gemp et al., 2021), which is a game-theoretic formulation for streaming PCA. The underlying optimization problem can be understood as a variant of the *sequential* version of the subspace characterization (2); see Sec. B.2.2 and (15) therein. The resulting optimization, however, still suffers from its biased gradient estimation, and requires the parametric functions to be normalized, i.e., $\|\hat{\phi}_\ell\|_2 = \langle \hat{\phi}_\ell | \hat{\phi}_\ell \rangle^{1/2} = 1$ for every ℓ. While the biased gradient could be alleviated via its simple variant as we explain in Sec. B.2.2, our experiments show that the function normalization step may slow down the convergence in practice.

We note that both SpIN and NeuralEF require each parametric eigenfunction to be parameterized separately, i.e., without shared parameters among them, to ensure that their optimization schemes work. In practice, while using disjoint models is a straightforward choice, it may consume excessive memory if the number of modes L to be retrieved is large or if the model becomes more complex. To address both scenarios, in the next section, we provide two techniques: one suitable for disjoint parameterization (Sec. 3.2.1) and the other for joint parameterization (Sec. 3.2.2).

3. SVD via Nested Low-Rank Approximation

In what follows, we propose a new optimization-based algorithm for SVD with neural networks, based on Schmidt's approximation theorem combined with new techniques called *nesting* for learning the singular functions in order. The resulting framework is significantly conceptually simpler and easier to implement than prior methods, without introducing sophisticated optimization techniques. Further, unlike SpIN and NeuralEF, we can directly perform the SVD of a non-self-adjoint operator. Hereafter, we assume that $\mathcal{T}$ has $\{(\sigma_\ell, |\phi_\ell\rangle, |\psi_\ell\rangle)\}_{\ell=1}^{\infty}$ as its orthonormal singular triplets.

3.1. Learning Subspaces via Low-Rank Approximation

Let L be the number of modes we wish to retrieve. We will use a shorthand notation $\mathbf{f}_{1:\ell}(x) := [f_1(x), \ldots, f_\ell(x)]^\intercal$. Below, we will employ distinct variables $|f\rangle$ and $|g\rangle$ as counterparts to $|\phi\rangle$ and $|\psi\rangle$, respectively, which represent normalized singular functions. The intentional use of separate variables $|f\rangle$ and $|g\rangle$ underscores their role in representing *scaled* singular functions rather than normalized ones within our framework. The importance of this distinction will become apparent in the following subsection.

For the top-L SVD of a given operator $\mathcal{T}$, we consider the *low-rank approximation* (LoRA) objective defined as

$$\begin{aligned} \mathcal{L}_{\mathsf{LoRA}}(\mathbf{f}_{1:L}, \mathbf{g}_{1:L}) &:= \mathcal{L}_{\mathsf{LoRA}}(\mathbf{f}_{1:L}, \mathbf{g}_{1:L}; \mathcal{T}) \\ &:= -2 \sum_{\ell=1}^{L} \langle g_\ell | \mathcal{T} f_\ell \rangle + \sum_{\ell=1}^{L} \sum_{\ell'=1}^{L} \langle f_\ell | f_{\ell'} \rangle \langle g_\ell | g_{\ell'} \rangle. \end{aligned} \tag{3}$$

This objective can be derived as the approximation error of $\mathcal{T}$ via a low-rank expansion $\sum_{\ell=1}^{L} |f_\ell\rangle\langle g_\ell|$ measured in the squared Hilbert–Schmidt norm, for a compact operator $\mathcal{T}$. We defer its derivation to Sec. C.1. By Schmidt's LoRA theorem (Schmidt, 1907), which is the operator counterpart of Eckart & Young (1936) for matrices, $(\mathbf{f}^\star, \mathbf{g}^\star)$ corresponds to the rank-L approximation of $\mathcal{T}$. The proof of the following theorem can be found in Sec. C.2.

Theorem 3.1. *Assume that $\mathcal{T}\colon \mathcal{F} \to \mathcal{G}$ is compact. Let $((f_\ell^\star, g_\ell^\star))_{\ell=1}^{L} \in (\mathcal{F} \times \mathcal{G})^L$ be a global minimizer of $\mathcal{L}_{\mathsf{LoRA}}(\mathbf{f}_{1:L}, \mathbf{g}_{1:L})$. If $\sigma_L > \sigma_{L+1}$, then*

$$\sum_{\ell=1}^{L} |g_\ell^\star\rangle\langle f_\ell^\star| = \sum_{\ell=1}^{L} \sigma_\ell |\psi_\ell\rangle\langle\phi_\ell|.$$

In cases of degeneracy, i.e., when multiple singular functions share the same singular value, a minimizer will still recover a subspace spanned by the singular functions associated with that particular singular value. Throughout, we will assume such strict spectral gap assumptions for the sake of simple exposition.

3.2. Nesting for Learning Ordered Singular Functions

While the LoRA characterization of the spectral subspaces is favorable in practice due to its unconstrained nature, a global minimizer only characterizes the top-L singular *subspaces*; note that $(\mathsf{Q}\mathbf{f}^\star, \mathsf{Q}\mathbf{g}^\star)$ for any orthogonal matrix $\mathsf{Q} \in \mathbb{R}^{L\times L}$ is also a global minimizer. We thus require an additional technique to find the singular functions and singular values *in order* by breaking the symmetry in the objective $\mathcal{L}_{\mathsf{LoRA}}(\mathbf{f}_{1:L}, \mathbf{g}_{1:L})$.

The idea for learning the ordered solution is as follows. Suppose that we can find a common global minimizer $(\mathbf{f}_{1:L}^\star, \mathbf{g}_{1:L}^\star)$ of the objectives $\mathcal{L}_{\mathsf{LoRA}}(\mathbf{f}_{1:\ell}, \mathbf{g}_{1:\ell})$ for $1 \le \ell \le L$. Then, from the optimality in Theorem 3.1, $\sum_{i=1}^{\ell} |g_i^\star\rangle\langle f_i^\star|$ must be the rank-ℓ approximation of $\mathcal{T}$ for each $\ell \in [L]$, which is $\sum_{i=1}^{\ell} \sigma_i |\psi_i\rangle\langle\phi_i|$. By telescoping, we then have $|g_\ell^\star\rangle\langle f_\ell^\star| = \sigma_\ell |\psi_\ell\rangle\langle\phi_\ell|$ for each $\ell \in [L]$, which is the desired solution. Since the optimization is performed with a certain nested structure, we call this idea *nesting*.

We remark that, unlike most existing methods that aim to directly learn ortho-*normal* eigenfunctions, the global optimum with (nested) LoRA characterizes the correct singular functions *scaled* by the singular value σ_ℓ, as alluded to earlier. Using this property, one can estimate σ_ℓ by computing the product of norms $\|f_\ell^\star\| \cdot \|g_\ell^\star\|$; see Sec. D.5 for the detail.

Below, we introduce two different versions that implement this idea: *sequential nesting*, which is ideal when each eigenfunction is parameterized by disjoint neural networks, and *joint nesting*, which can be used even when they may share parameters.

3.2.1. Sequential Nesting

Sequential nesting is based on the following observation: if $(\mathbf{f}_{1:\ell-1}, \mathbf{g}_{1:\ell-1})$ already captures the top-$(\ell-1)$ singular subspaces as a minimizer of $\mathcal{L}_{\mathsf{LoRA}}(\mathbf{f}_{1:\ell-1}, \mathbf{g}_{1:\ell-1})$, minimizing $\mathcal{L}_{\mathsf{LoRA}}(\mathbf{f}_{1:\ell}, \mathbf{g}_{1:\ell})$ for (f_ℓ, g_ℓ) finds the ℓ-th singular functions. Its proof can be found in Sec. C.3. Formally:

Theorem 3.2. *Assume that $\mathcal{T}\colon \mathcal{F} \to \mathcal{G}$ is compact. Pick any $\ell \ge 1$. Let $(\mathbf{f}_\ell^\star, \mathbf{g}_\ell^\star) \in \mathcal{F} \times \mathcal{G}$ be a global minimizer of $\mathcal{L}_{\mathsf{LoRA}}(\mathbf{f}_{1:\ell}, \mathbf{g}_{1:\ell})$, where $\sum_{i=1}^{\ell-1} |g_i\rangle\langle f_i| = \sum_{i=1}^{\ell-1} \sigma_i |\psi_i\rangle\langle\phi_i|$. If $\sigma_\ell > \sigma_{\ell+1}$, then $|g_\ell^\star\rangle\langle f_\ell^\star| = \sigma_\ell |\psi_\ell\rangle\langle\phi_\ell|$.*

We can implement this idea by simultaneously updating the iterate $(f_\ell^{(t)}, g_\ell^{(t)})$ at time step $t \ge 1$ for each $\ell \in [L]$, to minimize $\mathcal{L}_{\mathsf{LoRA}}(\mathbf{f}_{1:\ell}^{(t)}, \mathbf{g}_{1:\ell}^{(t)})$, treating $(\mathbf{f}_{1:\ell-1}^{(t)}, \mathbf{g}_{1:\ell-1}^{(t)})$ as a good proxy to the global optimum. That is, for each $\ell \in [L]$,

$$\begin{aligned}&(f_\ell^{(t+1)}, g_\ell^{(t+1)}) \\ &\leftarrow \mathsf{GradOpt}((f_\ell^{(t)}, g_\ell^{(t)}), \partial_{(f_\ell, g_\ell)} \mathcal{L}_{\mathsf{LoRA}}(\mathbf{f}_{1:\ell}^{(t)}, \mathbf{g}_{1:\ell}^{(t)})).\end{aligned} \tag{4}$$

Here, $\mathsf{GradOpt}(\boldsymbol{\theta}, \mathbf{g})$ denotes a gradient-based optimization algorithm that returns the next iterate based on the current iterate $\boldsymbol{\theta}$ and the gradient $\mathbf{g}$.

Suppose that each model pair (f_ℓ, g_ℓ) is parameterized via L separate models with (disjoint) parameters $\theta = (\theta_\ell)_{\ell=1}^{L}$. In this case, the ℓ-th eigenfunction can be updated independently from the ℓ'-th eigenfunctions for $\ell' > \ell$ via the sequential nesting (4). Hence, while all $(\mathbf{f}_{1:L}, \mathbf{g}_{1:L})$ are optimized simultaneously, the optimization is *inductive* in the sense that the modes can be learned in the order of the singular values. As a shorthand notation, let

$$\mathcal{L}_\ell := \mathcal{L}_{\mathsf{LoRA}}(\mathbf{f}_{1:\ell}, \mathbf{g}_{1:\ell}).$$

The gradient in (4) can be directly implemented by updating each θ_ℓ with the gradient

$$\partial_{\theta_\ell} \mathcal{L}_\ell = \langle \partial_{\theta_\ell} f_\ell |\, \partial_{f_\ell} \mathcal{L}_\ell \rangle + \langle \partial_{\theta_\ell} g_\ell |\, \partial_{g_\ell} \mathcal{L}_\ell \rangle,$$

$$\text{where } |\, \partial_{f_\ell} \mathcal{L}_\ell \rangle = 2\Big\{ -|\mathcal{T}^* g_\ell\rangle + \sum_{i=1}^{\ell} |f_i\rangle\langle g_i | g_\ell\rangle \Big\}, \tag{5}$$

and $|\, \partial_{g_\ell} \mathcal{L}_\ell \rangle$ can be similarly computed by a symmetric expression. Note that $|\, \partial_{\theta_\ell} f_\ell \rangle$ should be understood as a vector-valued function of dimension $|\theta_\ell|$, i.e., the number of parameters in θ_ℓ. This gradient can be computed in a vectorized manner over $\ell \in [L]$.

3.2.2. Joint Nesting

As alluded to earlier, in the case of a shared parameterization, the sequential nesting (4) may exhibit behavior that differs from its inductive nature with the shared parameterization.[2] For example, for a shared model, imperfect

[2] We can still apply sequential nesting even when the functions are parameterized by a shared model; see Sec. D.6 for a discussion.

functions (f_ℓ, g_ℓ) for some $\ell \in [L]$ may affect the already perfectly matched singular functions, say, (f_1, g_1), unlike the disjoint parameterization case.

Interestingly, there is an alternative way to implement the idea of nesting that works for a shared parameterization with a guarantee. The key observation is that the ordered singular values and functions $\{(\sigma_\ell, \phi_\ell, \psi_\ell)\}_{\ell=1}^L$ can be characterized as the global minimizer of a single objective function, by taking a weighted sum of $\{\mathcal{L}_\ell = \mathcal{L}_{\mathsf{LoRA}}(\mathbf{f}_{1:\ell}, \mathbf{g}_{1:\ell})\}_{\ell=1}^L$ with positive weights. That is, define, for any positive weights $\mathbf{w} = (w_1, \dots, w_L) \in \mathbb{R}^L_{>0}$,

$$\mathcal{L}_{\mathsf{jnt}}(\mathbf{f}_{1:L}, \mathbf{g}_{1:L}; \mathbf{w}) := \sum_{\ell=1}^L w_\ell \mathcal{L}_{\mathsf{LoRA}}(\mathbf{f}_{1:\ell}, \mathbf{g}_{1:\ell}). \quad (6)$$

Theorem 3.3. *Assume that $\mathcal{T}\colon \mathcal{F} \to \mathcal{G}$ is compact. Let $((f_\ell^\star, g_\ell^\star))_{\ell=1}^L \in (\mathcal{F} \times \mathcal{G})^L$ be a global minimizer of $\mathcal{L}_{\mathsf{jnt}}(\mathbf{f}_{1:L}, \mathbf{g}_{1:L}; \mathbf{w})$. For any positive weights $\mathbf{w} \in \mathbb{R}^L_{>0}$, if the top-$(L+1)$ singular values are all distinct, $|g_\ell^\star\rangle\langle f_\ell^\star| = \sigma_\ell |\psi_\ell\rangle\langle\phi_\ell|$ for each $\ell \in [L]$.*[3]

See Sec. C.4 for its proof. The proof readily follows from observing that the joint objective $\mathcal{L}_{\mathsf{jnt}}(\mathbf{f}_{1:\ell}, \mathbf{g}_{1:\ell})$ is minimized if and only if $\mathcal{L}_{\mathsf{LoRA}}(\mathbf{f}_{1:\ell}, \mathbf{g}_{1:\ell})$ is minimized for each $\ell \in [L]$, i.e., $(\mathbf{f}_{1:\ell}, \mathbf{g}_{1:\ell})$ characterizes the top-ℓ singular subspaces for each $\ell \in [L]$. Any positive weights guarantee consistency, but we empirically found that the uniform weights $\mathbf{w} = (\frac{1}{L}, \dots, \frac{1}{L})$ work well in practice.

Since the joint nesting is based on a single objective function (6), the optimization is as simple as

$$(\mathbf{f}_{1:L}^{(t+1)}, \mathbf{g}_{1:L}^{(t+1)}) \quad (7)$$
$$\leftarrow \mathsf{GradOpt}((\mathbf{f}_{1:L}^{(t)}, \mathbf{g}_{1:L}^{(t)}), \partial_{(\mathbf{f}_{1:L}, \mathbf{g}_{1:L})} \mathcal{L}_{\mathsf{jnt}}(\mathbf{f}_{1:L}^{(t)}, \mathbf{g}_{1:L}^{(t)}; \mathbf{w})).$$

Even though the joint nesting can be implemented directly using an autograd package with (6), overall training can be nearly twice as fast via manual gradient computation. By the chain rule, the gradient can be computed as

$$\partial_\theta \mathcal{L}_{\mathsf{jnt}} = \sum_{\ell=1}^L \{\langle \partial_\theta f_\ell |\, \partial_{f_\ell} \mathcal{L}_{\mathsf{jnt}}\rangle + \langle \partial_\theta g_\ell |\, \partial_{g_\ell} \mathcal{L}_{\mathsf{jnt}}\rangle\}, \text{ where}$$

$$|\partial_{f_\ell} \mathcal{L}_{\mathsf{jnt}}\rangle = 2\Big\{-\mathsf{m}_\ell |\mathcal{T}^* g_\ell\rangle + \sum_{i=1}^L \mathsf{M}_{i\ell} |f_i\rangle\langle g_i | g_\ell\rangle\Big\} \quad (8)$$

and $|\,\partial_{g_\ell} \mathcal{L}_{\mathsf{jnt}}\rangle$ is similarly computed. Here, we define the vector mask as $\mathsf{m}_\ell := \sum_{i=\ell}^L w_i$ and the matrix mask as $\mathsf{m}_{\ell\ell'} = \mathsf{m}_{\max\{\ell,\ell'\}}$; see Sec. C.5 for a formal derivation. Lastly, setting $\mathsf{m}_\ell \leftarrow 1$ and $\mathsf{m}_{i\ell} \leftarrow \mathbb{1}\{i \le \ell\}$ in (8) recovers the sequential nesting gradient (5). Therefore, both versions of nesting can be implemented in a unified way via (8).

[3] Again, the strict spectral gap is assumed for simplicity; when there exist a degeneracy, the optimally learned functions should recover the orthonormal eigenbasis of the corresponding subspace.

Remark 3.4 (Comparison to sequential nesting). In general, joint nesting may be less effective than sequential nesting with disjoint parameterization, as learning the top modes is affected by badly initialized latter modes, potentially slowing down the convergence. This is empirically demonstrated in Sec. 4.1. For the case of joint parameterization, however, we also empirically observe that joint nesting can outperform sequential nesting, as expected; see Sec. 4.2. Hence, we suggest users choose the version of nesting depending on the form of parameterization. We provide an additional remark in Sec. 5.

3.3. NeuralSVD: Nested LoRA with Neural Networks

When combined with NN eigenfunctions, we call the overall approach *NeuralSVD$_{seq}$* and *NeuralSVD$_{jnt}$* based on the version of nesting, or *NeuralSVD* for simplicity. While the parametric approach can work with any parametric functions, we adopt the term *neural* given that NNs represent a predominant class of powerful parametric functions.

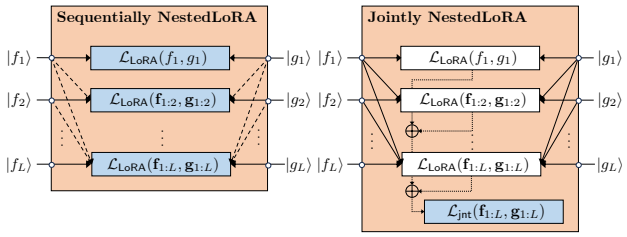
In practice, we will need to use minibatch samples for optimization. We explain how to implement the gradient updates of NestedLoRA based on the expression (8) in a greater detail in Sec. D with PyTorch code snippets. We have open-sourced a PyTorch implementation of our method, along with our implementations of SpIN and NeuralEF with a unified I/O interface for a fair comparison.[4]

We emphasize that, to apply NeuralSVD (and other existing methods), we only need to know how to evaluate a quadratic form $\langle f | \mathcal{T} g\rangle$ and inner products such as $\langle f | f'\rangle$. Since we consider $\mathscr{L}^2$ spaces for most applications, and the quadratic forms and inner products can be estimated via importance sampling or given data in an unbiased manner; see a detailed discussion on importance sampling to Sec. D.3. After all, the gradients described above can be estimated without bias, and we can thus use any off-the-shelf stochastic optimization method with minibatch to solve the optimization problem. Given a minibatch of size B, we can compute the minibatch objective and gradient only with matrix-vector products, and the complexity is $O(B^2L + BL^2)$.

4. Example Applications and Experiments

In this section, we illustrate two example use cases and present experimental results: *differential operators* in computational physics, and *canonical dependence kernels* in machine learning, which will be defined in Sec. 4.2. We experimentally demonstrate the correctness of NeuralSVD and its ability to learn ordered eigen- or singular-functions and show the superior performance of our method compared to the existing methods. We focus on rather small-scale problems that suffice with simple multi-layer perceptrons

[4] https://github.com/jongharyu/neural-svd



(a) Sequential nesting. *(b)* Joint nesting.

Figure 2: Schematic illustrations of the nesting techniques; recall Fig. 1. The operator and data (and weights for the joint nesting) are omitted for simplicity. In both cases, gradients are computed and backpropagated from the blue boxes. Gradients cannot be backpropagated through the dashed lines in (a); see Sec. 3.2.

(MLPs) for extensive numerical evaluation of our method against the existing parametric methods. All the training details can be found in Sec. E.

4.1. Analytical Operators

In many application scenarios, an operator $\mathcal{T}$ is given in an analytical form. In machine learning, there exists a variety of *kernel-based methods*, which assumes a certain kernel function $k(x, y)$ defined in a closed form. In this case, the underlying operator is the so-called *integral kernel operator* $\mathcal{K}$, which is defined as $(\mathcal{K}f)(y) := \mathbb{E}_{p(x)}[k(x, y)f(x)]$. In computational physics, a certain class of important PDEs can be reduced to eigenvalue problems, where we can directly apply our framework to solve them. In this case, the operator involves a differential operator, such as Laplacian ∇^2, as will become clear below. We will provide a numerical demonstration of NeuralSVD for the latter scenario.

A representative example of such PDE is a time-independent Schrödinger equation (TISE) (Griffiths & Schroeter, 2018)

$$\mathcal{H}|\psi\rangle = E|\psi\rangle.$$

Here, $\mathcal{H}$ is the Hamiltonian that characterizes a given physical system, $|\psi\rangle$ denotes an eigenfunction, and $E \in \mathbb{R}$ the corresponding eigen-energy. Recall that to perform EVD in our SVD framework, we only need to identify $\mathbf{g}$ to $\mathbf{f}$. Since bottom modes are typically of physical interest, we can aim to find the eigenfunctions of the *negative* Hamiltonian $-\mathcal{H}$.

We consider two simple yet representative examples of TISEs that have closed-form solutions for extensive quantitative evaluations. The first example is a 2D hydrogen atom, the corresponding operator of which is compact. With the second example of a 2D harmonic oscillator, in which the operator of interest is *not* compact, we demonstrate that our framework is still applicable. In both cases, we used simple MLPs with multi-scale random Fourier features as the parametric eigenfunctions (Wu et al., 2023).

Experiment 1: 2D Hydrogen Atom. We first consider a hydrogen atom confined over a 2D plane. By solving the associated TISE, we aim to learn a few bottom eigenstates and their respective eigenenergies. The detailed problem setting, including the underlying PDE, can be found in Sec. E. Ignoring irrelevant constants, the true eigenvalues (after negating the sign) are known as $\lambda_{n,l} = (2n + 1)^{-2}$ for $n \geq 0$ and $-n \leq l \leq n$. That is, for each n, there exist $2n + 1$ degenerate states. In our experiment, we aimed to learn $L = 16$ eigenstates that cover the first four degenerate eigensubspaces. We trained SpIN, NeuralEF, NeuralSVD$_{\text{seq}}$, and NeuralSVD$_{\text{jnt}}$ with the same architecture and training procedure with different batch sizes 128 and 512.[5] Here, we found that the original version of NeuralEF performed much worse than NeuralSVD, and we thus implemented and reported the result of a variant of NeuralEF with unbiased gradient estimates, whose definition can be found in Sec. B.2.2.

Fig. 3 shows the learned eigenfunctions from SpIN (128), NeuralEF (512), and NeuralSVD$_{\text{seq}}$ (512), where the numbers in the parentheses indicate used batch sizes. For comparison, we present the true eigenfunctions with a choice of canonical directions to plot the degenerate subspaces (last row). Note that SpIN and NeuralEF do not match the ground truths even after the rotation in several modes. Further, the learned functions (before rotation) are not orthogonal as visualized in the rightmost column. In contrast, NeuralSVD$_{\text{seq}}$ can reliably match the correct eigenfunctions, with almost perfect orthogonality. Fig. 4 report several quantitative measures to evaluate the fidelity of learned eigenfunctions; see Sec. E.1.3 for the definitions of the measures. The results show that both NeuralSVD$_{\text{jnt}}$ and NeuralSVD$_{\text{seq}}$ outperform SpIN and NeuralEF by an order of magnitude.

Note that though NeuralSVD$_{\text{jnt}}$ can recover the eigenfunctions reasonably well, it performs worse than NeuralSVD$_{\text{seq}}$ as expected; see Remark 3.4. We also remark that the computational and memory complexity of NeuralEF and NeuralSVD are almost the same, while SpIN takes much longer time and consumes more memory due to the Cholesky decomposition and the need for storing the Jacobian; we refer an interested reader to Sec. B.2.1 for the detail of SpIN.

In the Appendix, we demonstrate the advantages of NeuralSVD compared to standard numerical linear algebra techniques; see Sec. A.3 for its comparison to a matrix-free method, and Sec. A.2 for the effectiveness of nesting.

Experiment 2: 2D Harmonic Oscillator. We now consider finding the eigenstates of a 2D harmonic oscillator, whose eigenstate is characterized by a pair of nonegative integers (n, l) for $n \geq 0$ and $0 \leq l \leq n$ with (negative) eigenenergy

[5] As an exception, a smaller network and batch size 128 was used for SpIN due to its large memory requirement $O(L \times$ (model size)) for maintaining copies of Jacobian for each mode.

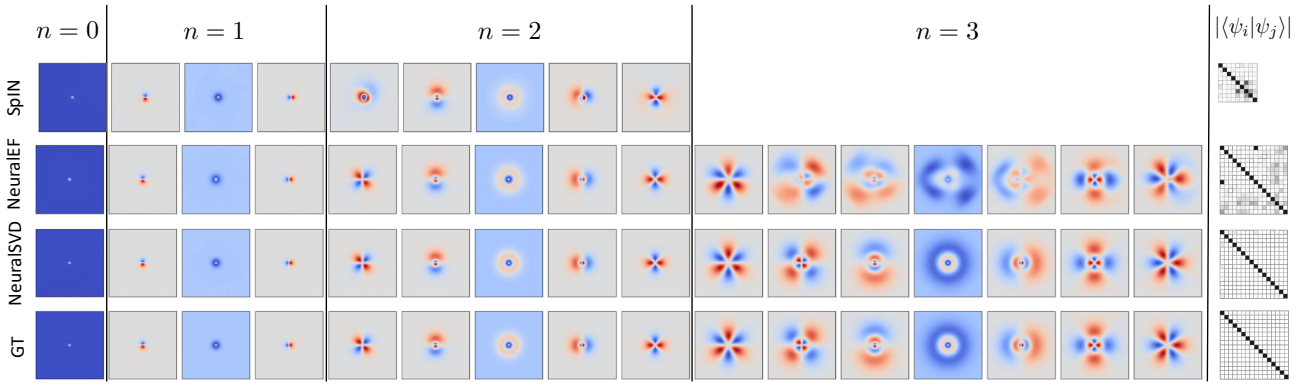


Figure 3: Visualization of the first 16 eigenfunctions of the 2D hydrogen atom. The first three rows present the learned eigenfunctions by SpIN (128), NeuralEF (512), and NeuralSVD$_{\text{seq}}$ (512), respectively. Due to the memory complexity of SpIN, we ran SpIN with only 9 eigenstates. The learned functions are aligned by an orthogonal transformation via the orthogonal Procrustes method within each degenerate subspace to compare with the ground truth (GT) in the fourth row. The rightmost column visualizes the learned orthogonality.

$\lambda_{n,l} = -2(n+1)$ and multiplicity of $n+1$. In contrast to the 2D hydrogen case, it is clear that the negative Hamiltonian is neither PD nor compact. To retrieve eigenfunctions even in this case, we can consider a *shifted* operator $\mathcal{T} + c\mathcal{I}$, where $\mathcal{I}$ is an identity operator and $c \geq 0$ is a constant, so that the spectrum becomes $\lambda_{n,l} = c - 2(n+1)$. Note that shifting only affects the quadratic form $\langle f|\mathcal{T} + c\mathcal{I}|f\rangle = \langle f|\mathcal{T}f\rangle + c\|f\|^2$.

As an example, we chose $c = 16$, so that $\lambda_{n,l} > 0$ for $0 \leq n \leq 6$. We claim that NeuralSVD recovers the eigenfunctions with positive eigenvalues, the first $28 (= 1 + \ldots + 7)$ states for this case, and the nonpositive part will converge to the constant zero function; see Theorem C.5 in Sec. C.6. We note that the LoRA objective (3) is still well-defined even when $\mathcal{T}$ is not compact. While other methods are also applicable and can recover the positive part in principle, the learned functions will be arbitrary for the nonpositive part, unlike NeuralSVD learning zero functions. This implies that one can correctly infer the nonpositive part by computing the norms of the NeuralSVD eigenfunctions.

We report the quantitative measures in Fig. 4(b), where only the positive part, i.e., the first 28 eigenstates, was taken into account for the evaluation. Note that SeqNested-LoRA significantly outperforms NeuralEF in this example as well. Moreover, as explained above, the norms of the learned eigenfunctions with NeuralSVD well approximate the ground truth eigenvalues for the positive part, and almost zero for the non-positive part (data not shown); see Sec. D.5 for the spectrum estimation with NeuralSVD based on function norms. In contrast, one cannot distinguish whether learned eigenfunctions are meaningful or not only based on the learned eigenvalues from NeuralEF.

4.2. Data-Dependent Operators

Beyond analytical operators, we can also consider a special type of *data-dependent* kernels. Given a joint distribution $p(x, y)$, we define

$$k(x, y) := \frac{p(x, y)}{p(x)p(y)}$$

which we call the *canonical dependence kernel* (CDK). Although the CDK cannot be explicitly evaluated, it naturally defines the similarity between x and y based on their joint distribution, and thus can better capture the statistical relationship than a fixed, analytical kernel. Note that its induced integral kernel operator is the conditional expectation operator, i.e., $(\mathcal{K}g)(x) = \mathbb{E}[g(Y)|X = x]$ and $(\mathcal{K}^* f)(y) = \mathbb{E}[f(X)|Y = y]$, where $\mathcal{K}^*$ denotes the *adjoint* of $\mathcal{K}$.

CDK appears and plays a central role in several statistics and machine learning applications, and various connections of CDK to the existing literature such as Hirschfeld–Gebelein–Rényi (HGR) maximal correlation (Hirschfeld, 1935; Gebelein, 1941; Rényi, 1959) are discussed in Sec. B.

One special property of CDK is that we can compute the objective function using paired samples, even though we do not know the kernel value $p(x, y)/(p(x)p(y))$ in general. That is, the "operator term" $\langle g_\ell|\mathcal{K}f_\ell\rangle$ can be computed as

$$\langle g_\ell|\mathcal{K}f_\ell\rangle = \mathbb{E}_{p(x,y)}[f_\ell(x)g_\ell(y)], \tag{9}$$

where we change the measure $p(x)p(y)$ with $p(x, y)$ by the definition of $k(x, y)$. Since the first singular functions are always constant functions, we can simply replace $\mathbf{f}(x)^\intercal\mathbf{g}(y)$ with $1 + \mathbf{f}(x)^\intercal\mathbf{g}(y)$ to exclude the trivial mode, so that we can recover the second singular functions and on. We note in passing that it is equivalent to decomposing kernel $\frac{p(x,y)}{p(x)p(y)} - 1$, which is the convention used in a line of literature; see, e.g., (Huang et al., 2024; Xu & Zheng, 2024).

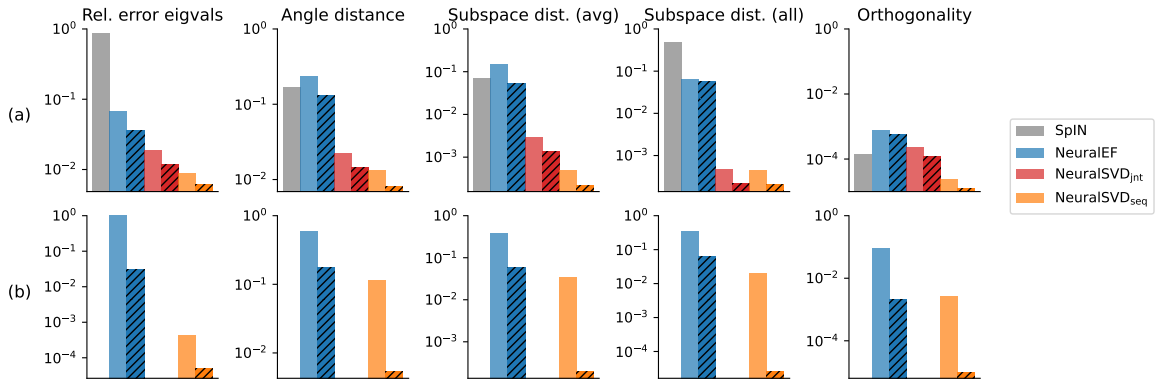


Figure 4: Summary of quantitative evaluations for solving TISEs: (a) 2D hydrogen atom; (b) 2D harmonic oscillator. Non-hatched, light-colored bars represent a batch size of 128, and hatched bars indicate 512. The definitions of reported measures are given in Sec. E.1.3.

Table 1: Evaluation of the ZS-SBIR task with the Sketchy Extended dataset (Sangkloy et al., 2016; Liu et al., 2017). We note that the two baselines require ($*$) a generative model, while NeuralSVD can learn representations directly without such.

Method	Ext. knowledge	Gen. model	Structured	P@100	mAP	Split
LCALE (Lin et al., 2020)	Word embeddings	$*$	✗	0.583	0.476	1
IIAE (Hwang et al., 2020)		$*$	✗	0.659	0.573	1
NeuralSVD$_{jnt}$			✓	$\mathbf{0.670}_{\pm 0.010}$	$\mathbf{0.581}_{\pm 0.008}$	1
				$\mathbf{0.724}_{\pm 0.008}$	$\mathbf{0.641}_{\pm 0.008}$	2

Application: Cross-Domain Retrieval. One natural application of the CDK is in the cross-domain retrieval problem. Specifically, here we consider the *zero-shot sketch-based image retrieval* (ZS-SBIR) task proposed by Yelamarthi et al. (2018). The goal is to construct a good model that retrieves relevant photos y_i's from a given query sketch x, only using a training set with no overlapping classes in the test set (hence called *zero-shot*).

To obtain coembeddings of sketches and photos from the CDK framework, we define a natural joint distribution $p(x, y)$ for sketch x and photo y, by picking a random pair of (x, y) from the same class. Formally, the joint distribution is defined as $p(x, y) = \mathbb{E}_{p(c)}[p(x|c)p(y|c)]$, where $p(c)$ denotes the class distribution, and $p(x|c)$ and $p(y|c)$ the class-conditional sketch and photo distributions, respectively. We emphasize that the resulting joint distribution is *asymmetric*, since x and y are two different modalities, and thus the existing frameworks, such as SpIN or NeuralEF cannot be directly applied. We also note that the matrix approach, which computes the empirical CDK matrix and then performs SVD, is infeasible, as density ratio estimation for constructing the kernel matrix is nontrivial in the high-dimensional space. In sharp contrast, we can learn to decompose the CDK $k(x, y) = \frac{p(x,y)}{p(x)p(y)} \approx 1 + \mathbf{f}(x)^\intercal \mathbf{g}(y)$ directly with NeuralSVD.

After learning the functions $\mathbf{f}$ and $\mathbf{g}$, for a given query x, we can retrieve based on the highest inner-product $\mathbf{f}(x)^\intercal \mathbf{g}(y)$ from $y \in \{y_1, \dots, y_N\}$. This approach has a natural probabilistic interpretation: "retrieve y, if y is more likely to appear together than independently, i.e., $p(x, y) \gg p(x)p(y)$". In addition to the interpretable retrieval scheme, the retrieval system can benefit from the learned spectral structure. That is, when successfully learned, the NeuralSVD representations are ideally stacks of *ordered* top-L singular functions of the CDK. The representations can thus be called *structured* in the sense that the coordinates of representations are ordered by the associated singular values, and also each coordinate encodes exclusive information since different coordinates are constructed so as to be effectively orthogonal. We can thus potentially reduce the dimensionality of the embedding, by keeping only informative coordinates.

Experiment. We aimed to learn $L = 512$ singular functions, parameterizing them by a single network. We followed the standard training setup in the literature (Hwang et al., 2020). We report the Precision@100 (P@100) and mean average precision (mAP) scores on the two test splits in the literature; we defer the definition of these metrics to Sec. E.2. We empirically found that NeuralSVD$_{seq}$ performed much worse than NeuralSVD$_{jnt}$ as discussed in Remark 3.4, only achieving Precision@100 around 0.2. Hence, we only report the result from NeuralSVD$_{jnt}$. Since SpIN and NeuralEF are not directly applicable to asymmetric kernels, we do not include them in the comparison.

Table 1 summarizes the evaluation. It shows that the CDK-based retrieval learned by NeuralSVD, albeit simple, can

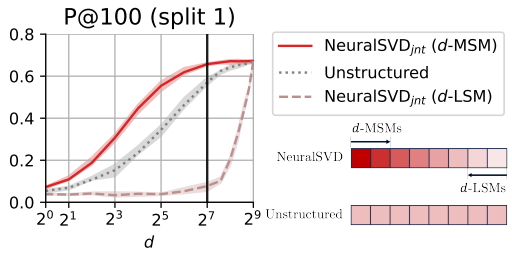


Figure 5: P@100 performance of NeuralSVD on ZS-SBIR task, with varying dimensions. NeuralSVD can achieve full performance only with one-quarter ($2^7 = 128$) of the full dimensions, sweeping from the most significant modes.

outperform the state-of-the-art representation learning methods based on generative models, including a method that incorporates additional knowledge.

As alluded to above, moreover, we can demonstrate that NeuralSVD learns *structured* representations, while the baselines only learn "unstructured" ones. To illustrate, we verify that most information relevant for retrieval is concentrated around the top modes. To illustrate this, we repeated the following evaluation with 10 random initializations and report the summary in Fig. 5. First, from the NeuralSVD representations of $2^9 = 512$ dimensions, we used the d most significant modes (d-MSMs) for $d \in \{1, 2, 2^2, \ldots, 2^9\}$, and evaluated the retrieval performance based on the similarity measure $\mathbf{f}_{[d]}(x)^\intercal \mathbf{g}_{[d]}(y)$. The performance rapidly grows as the dimensionality d gets large, and the best is almost achieved at about $2^7 = 128$, which is only a quarter of the full dimension; see the red lines (NeuralSVD) and the vertical black lines. Also, we can empirically validate that the learned representations are almost perfectly orthogonal.

As a further investigation, we consider two additional baselines. First, from the NeuralSVD representations, we evaluated the retrieval performance with the d least significant modes (d-LSMs) for $d \in \{1, 2, 2^2, \ldots, 2^9\}$; see pink, dashed lines labeled with "NeuralSVD (d-LSM)". The retrieval performance is very poor even when using the bottom 128 dimensions, which indicates that the LSMs do not encode much information. Second, we trained an *unstructured* embedding by training the same network with the LoRA objective without nesting, so that the network only learns the top-512 subspace of CDK; see gray, dotted lines labeled with "Unstructured". As expected, its retrieval performance lies in the middle of NeuralSVD (MSM) and NeuralSVD (LSM). Hence, we can conclude that the learned representations with NeuralSVD are well-structured and effectively encode the information in a compact manner.

Remark 4.1 (Impact of imperfect orthogonality). Since our NeuralSVD framework only implicitly promotes the orthogonality without hard constraints, the learned singular-functions may only exhibit orthogonality to each other in an approximate manner, and such deviations from orthogonality may impact the performance in a downstream task. In the PDE example, if the goal is to find accurate eigenvalues (i.e., eigenenergies of the system), then slightly imperfect orthogonality across non-degenerate modes may result in slightly less accurate eigenvalue estimates. In the representation learning example, if the retrieval performance is the only criterion for the quality of representations, slightly imperfect orthogonality would result in "slightly less structured" representations in that different coordinates might share some redundant information, which will impact the "compressibility" of the representations. All in all, imperfect orthogonality could affect different tasks differently, but we provide an empirical showcase that almost perfect orthogonality can be guaranteed in the present examples.

5. Concluding Remarks

In this paper, we proposed NeuralSVD, a new optimization framework for learning parametric singular- or eigenfunctions of a linear operator via NestedLoRA. Given the efficient unconstrained optimization framework, practitioners can focus on selecting the most suitable parametric functions (or good architectures) and optimization algorithms to meet the practical requirements of their specific problems. We could potentially extend the applicability of the existing classical algorithms based on SVD/EVD in various fields, e.g., quantum chemistry (Hermann et al., 2020; Pfau et al., 2020) or spectral embedding methods (Shi & Malik, 2000; Belkin & Niyogi, 2003) for large-scale, high-dimensional data, combined with the use of powerful neural networks.

Limitations and Future Directions. We conclude with two important limitations and future directions to further advance the applicability of the parametric approach.

- First, the parametric approach is less explored than the nonparametric approach. The challenge is to understand when a large network can approximate a given operator and to determine an optimization algorithm that converges to the desired global optimizer, such as NestedLoRA. Addressing this gap and providing performance guarantees is a valuable research direction.

- Second, users of the parametric approach must choose an appropriate function and optimization hyperparameters. Our investigation has shown the effectiveness of simple MLP architectures and specific hyperparameters in our examples. However, for larger applications, scalability challenges may require more sophisticated architectures and fine-tuning. We advocate for future research to design effective network architectures tailored to specific operators and tasks.

Acknowledgements

The authors appreciate insightful feedback and helpful suggestions from anonymous reviewers to improve earlier versions of the manuscript. The authors also acknowledge the MIT SuperCloud and Lincoln Laboratory Supercomputing Center for providing HPC resources that have contributed to the research results reported within this paper. JJR and GWW were supported, in part, by the MIT-IBM Watson AI Lab under Agreement No. W1771646, and by MIT Lincoln Laboratory. HSME, XX, and LZ were supported by the Office of Naval Research (ONR) under grant N00014-19-1-2621.

Impact Statement

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

References

Andrew, G., Arora, R., Bilmes, J., and Livescu, K. Deep canonical correlation analysis. In *Proc. Int. Conf. Mach. Learn.*, pp. 1247–1255. PMLR, 2013. 18

Andrews, H. and Patterson, C. Singular value decompositions and digital image processing. *IEEE Trans. Acoust. Speech Signal Process.*, 24(1):26–53, 1976. 1

Bar, L. and Sochen, N. Unsupervised Deep Learning Algorithm for PDE-based Forward and Inverse Problems. *arXiv*, April 2019. 19

Belkin, M. and Niyogi, P. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Comput.*, 15(6):1373–1396, 2003. 1, 9

Ben-Shaul, I., Bar, L., Fishelov, D., and Sochen, N. Deep Learning Solution of the Eigenvalue Problem for Differential Operators. *Neural Comput.*, 35(6):1100–1134, May 2023. ISSN 0899-7667, 1530-888X. doi: 10.1162/neco_a_01583. 19

Bengio, Y., Vincent, P., Paiement, J.-F., Delalleau, O., Ouimet, M., and Le Roux, N. *Spectral clustering and kernel PCA are learning eigenfunctions*, volume 1239. CIRANO, 2003. 1

Bengio, Y., Delalleau, O., Le Roux, N., Paiement, J.-F., Vincent, P., and Ouimet, M. Learning eigenfunctions links spectral embedding and kernel pca. *Neural Comput.*, 16(10):2197–2219, 2004. 2, 16

Blum, A., Hopcroft, J., and Kannan, R. *Foundations of data science*. Cambridge University Press, 2020. 1

Bolla, M. *Spectral clustering and biclustering: Learning large graphs and contingency tables*. John Wiley & Sons, 2013. 15

Carleo, G. and Troyer, M. Solving the quantum many-body problem with artificial neural networks. *Science*, 355(6325):602–606, February 2017. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.aag2302. 19

Choo, K., Mezzacapo, A., and Carleo, G. Fermionic neural-network states for ab-initio electronic structure. *Nat. Commun.*, 11(1):2368, May 2020. ISSN 2041-1723. doi: 10.1038/s41467-020-15724-9. 19

Colbrook, M., Horning, A., and Townsend, A. Computing Spectral Measures of Self-Adjoint Operators. *SIAM Rev.*, 63(3):489–524, January 2021. ISSN 0036-1445. doi: 10.1137/20M1330944. 19

Cox, M. A. and Cox, T. F. Multidimensional scaling. In *Handbook of data visualization*, pp. 315–347. Springer, 2008. 1

Cuzzocrea, A., Scemama, A., Briels, W. J., Moroni, S., and Filippi, C. Variational Principles in Quantum Monte Carlo: The Troubled Story of Variance Minimization. *J. Chem. Theory Comput.*, 16(7):4203–4212, July 2020. ISSN 1549-9618, 1549-9626. doi: 10.1021/acs.jctc.0c00147. 19

Davies, E. B. and Plum, M. Spectral pollution. *IMA J. Numer. Anal.*, 24(3):417–438, 2004. 19

Deng, Z., Shi, J., Zhang, H., Cui, P., Lu, C., and Zhu, J. Neural Eigenfunctions Are Structured Representation Learners. *arXiv*, October 2022a. 21, 22

Deng, Z., Shi, J., and Zhu, J. NeuralEF: Deconstructing kernels by deep neural networks. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proc. Int. Conf. Mach. Learn.*, volume 162 of *Proceedings of Machine Learning Research*, pp. 4976–4992. PMLR, 17–23 Jul 2022b. URL https://proceedings.mlr.press/v162/deng22b.html. 2, 3, 18, 20

Dutta, A. and Akata, Z. Semantically tied paired cycle consistency for zero-shot sketch-based image retrieval. In *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 5089–5098, 2019. 35

E, W. and Yu, B. The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems. *Commun Math Stat.*, 6(1):1–12, 2018. 19

Eckart, C. and Young, G. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, September 1936. ISSN 0033-3123, 1860-0980. doi: 10.1007/BF02288367. 4, 18

Entwistle, M. T., Schätzle, Z., Erdman, P. A., Hermann, J., and Noé, F. Electronic excited states in deep variational Monte Carlo. *Nat. Commun.*, 14(1):274, January 2023. ISSN 2041-1723. doi: 10.1038/s41467-022-35534-5. 19

Gebelein, H. Das statistische Problem der Korrelation als Variations- und Eigenwertproblem und sein Zusammenhang mit der Ausgleichsrechnung. *ZAMM Z. Angew. Math. Mech.*, 21(6):364–379, 1941. ISSN 0044-2267, 1521-4001. doi: 10.1002/zamm.19410210604. 7, 18

Gemp, I., McWilliams, B., Vernade, C., and Graepel, T. EigenGame: PCA as a Nash equilibrium. In *Int. Conf. Learn. Repr.*, 2021. 3, 18, 21, 22, 23

Gemp, I., McWilliams, B., Vernade, C., and Graepel, T. EigenGame unloaded: When playing games is better than optimizing. In *Int. Conf. Learn. Repr.*, 2022. URL https://openreview.net/forum?id=So6YAqnqgMj. 23

Gerard, L., Scherbela, M., Marquetand, P., and Grohs, P. Gold-standard solutions to the Schrödinger equation using deep learning: How much physics do we need? In *Adv. Neural Inf. Proc. Syst.*, December 2022. 19

Goldberg, Y. and Levy, O. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*, 2014. 1

Golub, G. H. and Van Loan, C. F. *Matrix computations*. JHU press, 2013. 1

Greenacre, M. J. *Theory and applications of correspondence analysis*. London (UK) Academic Press, 1984. 18

Griffiths, D. J. and Schroeter, D. F. *Introduction to quantum mechanics*. Cambridge university press, 2018. 6

Guo, Y. and Ming, P. A Deep Learning Method for Computing Eigenvalues of the Fractional Schrödinger Operator. *arXiv*, August 2023. 19

Han, J., Lu, J., and Zhou, M. Solving high-dimensional eigenvalue problems using deep neural networks: A diffusion Monte Carlo like approach. *J. Comput. Phys.*, 423(109792):109792, December 2020. ISSN 0021-9991, 1090-2716. doi: 10.1016/j.jcp.2020.109792. 19

HaoChen, J. Z., Wei, C., Gaidon, A., and Ma, T. Provable guarantees for self-supervised deep learning with spectral contrastive loss. In *Adv. Neural Inf. Proc. Syst.*, 2021. 18

Hermann, J., Schätzle, Z., and Noé, F. Deep-neural-network solution of the electronic Schrödinger equation. *Nat. Chem.*, 12(10):891–897, October 2020. ISSN 1755-4330, 1755-4349. doi: 10.1038/s41557-020-0544-y. 1, 2, 9, 19

Hermann, J., Spencer, J., Choo, K., Mezzacapo, A., Foulkes, W. M. C., Pfau, D., Carleo, G., and Noé, F. Ab-initio quantum chemistry with neural-network wavefunctions. *arXiv*, August 2022. 19

Hinton, G., Srivastava, N., and Swersky, K. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. , 2012. 33

Hirschfeld, H. O. A Connection between Correlation and Contingency. *Math. Proc. Cambridge Philos. Soc.*, 31(4): 520–524, October 1935. ISSN 1469-8064, 0305-0041. doi: 10.1017/S0305004100013517. 7, 18

Holliday, E. G., Lindner, J. F., and Ditto, W. L. Solving two-dimensional quantum eigenvalue problems using physics-informed machine learning. *arXiv*, February 2023. 19

Hsu, H., Salamatian, S., and Calmon, F. P. Correspondence Analysis Using Neural Networks. In *Int. Conf. Artif. Int. Statist.*, February 2019. 18

Hsu, H., Salamatian, S., and Calmon, F. P. Generalizing Correspondence Analysis for Applications in Machine Learning. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44 (12):9347–9362, December 2022. ISSN 0162-8828. doi: 10.1109/TPAMI.2021.3127870. 18

Hu, B. and Principe, J. C. The cross density kernel function: A novel framework to quantify statistical dependence for random processes. *arXiv preprint arXiv:2212.04631*, 2022. 18

Huang, S.-L., Makur, A., Wornell, G. W., and Zheng, L. Universal features for high-dimensional learning and inference. *Found. Trends Commun. Inf. Theory*, 21(1-2):1–299, 2024. ISSN 1567-2190. doi: 10.1561/0100000107. URL http://dx.doi.org/10.1561/0100000107. 7, 30

Hwang, H., Kim, G.-H., Hong, S., and Kim, K.-E. Variational interaction information maximization for cross-domain disentanglement. In *Adv. Neural Inf. Proc. Syst.*, volume 33, 2020. 8, 35

Jin, H., Mattheakis, M., and Protopapas, P. Unsupervised Neural Networks for Quantum Eigenvalue Problems. *arXiv*, October 2020. 19

Jin, H., Mattheakis, M., and Protopapas, P. Physics-Informed Neural Networks for Quantum Eigenvalue Problems. In *Proc. Int. Jt. Conf. Neural Netw.*, pp. 1–8, July 2022. doi: 10.1109/IJCNN55064.2022.9891944. 19, 21

Knyazev, A. Recent implementations, applications, and extensions of the Locally Optimal Block Preconditioned Conjugate Gradient method (LOBPCG). *arXiv preprint arXiv:1708.08354*, 2017. 17

Knyazev, A. V. Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method. *SIAM J. Sci. Comput.*, 23(2):517–541, 2001. 1, 17

Kusupati, A., Bhatt, G., Rege, A., Wallingford, M., Sinha, A., Ramanujan, V., Howard-Snyder, W., Chen, K., Kakade, S., Jain, P., and Farhadi, A. Matryoshka representation learning. In *Adv. Neural Inf. Proc. Syst.*, volume 35, pp. 30233–30249, 2022. 18

Lagaris, I. E., Likas, A., and Fotiadis, D. I. Artificial neural network methods in quantum mechanics. *Comput. Phys. Commun.*, 104(1-3):1–14, August 1997. ISSN 0010-4655, 1879-2944. doi: 10.1016/s0010-4655(97)00054-4. 19

Landauer, T. K., Foltz, P. W., and Laham, D. An introduction to latent semantic analysis. *Discourse Process.*, 25(2-3): 259–284, 1998. 1

Li, H. and Ying, L. A semigroup method for high dimensional elliptic PDEs and eigenvalue problems based on neural networks. *arXiv*, May 2021. 19

Li, H., Zhai, Q., and Chen, J. Z. Y. Neural-network-based multistate solver for a static Schrödinger equation. *Phys. Rev. A*, 103(3):032405, March 2021a. ISSN 1050-2947. doi: 10.1103/PhysRevA.103.032405. 19

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Fourier neural operator for parametric partial differential equations. In *Int. Conf. Learn. Repr.*, 2021b. 19

Lin, K., Xu, X., Gao, L., Wang, Z., and Shen, H. T. Learning cross-aligned latent embeddings for zero-shot cross-modal retrieval. In *Proc. AAAI Conf. Artif. Int.*, volume 34, pp. 11515–11522, 2020. 8

Liu, J., Dou, X., He, X., Yang, C., and Jiang, G. Calculating many excited states of the multidimensional time-independent Schrödinger equation using a neural network. *Phys. Rev. A*, 108(3):032803, September 2023. ISSN 1050-2947. doi: 10.1103/PhysRevA.108.032803. 19

Liu, L., Shen, F., Shen, Y., Liu, X., and Shao, L. Deep sketch hashing: Fast free-hand sketch-based image retrieval. In *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 2862–2871, 2017. 8, 35

Loshchilov, I. and Hutter, F. SGDR: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 33, 36

Markovsky, I. *Low rank approximation: algorithms, implementation, applications*, volume 906. Springer, 2012. 1

Mattheakis, M., Schleder, G. R., Larson, D. T., and Kaxiras, E. First principles physics-informed neural network for quantum wavefunctions and eigenvalue surfaces. In *Machine Learning and the Physical Sciences workshop*, December 2022. 19

Michaeli, T., Wang, W., and Livescu, K. Nonparametric canonical correlation analysis. In *Proc. Int. Conf. Mach. Learn.*, pp. 1967–1976, 2016. 1, 19

Mirsky, L. Symmetric gauge functions and unitarily invariant norms. *Q. J. Math.*, 11(1):50–59, January 1960. ISSN 0033-5606. doi: 10.1093/qmath/11.1.50. 18

Ng, A., Jordan, M., and Weiss, Y. On spectral clustering: Analysis and an algorithm. In *Adv. Neural Inf. Proc. Syst.*, volume 14, pp. 849–856, 2001. 1

Pfau, D., Petersen, S., Agarwal, A., Barrett, D. G., and Stachenfeld, K. L. Spectral inference networks: Unifying deep and spectral learning. In *Int. Conf. Learn. Repr.*, 2019. 2, 3, 20, 33

Pfau, D., Spencer, J. S., Matthews, A. G. D. G., and Foulkes, W. M. C. Ab initio solution of the many-electron Schrödinger equation with deep neural networks. *Phys. Rev. Res.*, 2(3):033429, September 2020. doi: 10.1103/PhysRevResearch.2.033429. 1, 2, 9, 19

Pfau, D., Axelrod, S., Sutterud, H., von Glehn, I., and Spencer, J. S. Natural Quantum Monte Carlo Computation of Excited States. *arXiv*, August 2023. 19

Rényi, A. On measures of dependence. *Acta Math. Hungarica*, 10(3):441–451, September 1959. ISSN 1588-2632. doi: 10.1007/BF02024507. 7, 18

Roweis, S. T. and Saul, L. K. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290 (5500):2323–2326, 2000. 1

Saad, Y. Krylov subspace methods for solving large unsymmetric linear systems. *Math. Comput.*, 37(155):105–126, 1981. 1

Salton, G. and McGill, M. J. Introduction to modern information retrieval, 1986. 36

Sanger, T. D. Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural Netw.*, 2(6):459–473, January 1989. ISSN 0893-6080. doi: 10.1016/0893-6080(89)90044-0. 18

Sangkloy, P., Burnell, N., Ham, C., and Hays, J. The sketchy database: Learning to retrieve badly drawn bunnies. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 2016. 8, 35

Scetbon, M., Elad, M., and Milanfar, P. Deep k-svd denoising. *IEEE Trans. Image Proc.*, 30:5944–5955, 2021. 1

Schmidt, E. Zur Theorie der linearen und nichtlinearen Integralgleichungen. *Math. Ann.*, 63(4):433–476, December 1907. ISSN 0025-5831, 1432-1807. doi: 10.1007/BF01449770. 2, 4, 17, 24

Schölkopf, B., Smola, A., and Müller, K.-R. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Comput.*, 10(5):1299–1319, 1998. 1

Schütt, K., Kindermans, P.-J., Sauceda Felix, H. E., Chmiela, S., Tkatchenko, A., and Müller, K.-R. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Adv. Neural Inf. Proc. Syst.*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/303ed4c69846ab36c2904d3ba8573050-Paper.pdf. 19

Shi, J. and Malik, J. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):888–905, 2000. 1, 9

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. In *Int. Conf. Learn. Repr.*, 2015. 35

Sirignano, J. and Spiliopoulos, K. DGM: A deep learning algorithm for solving partial differential equations. *J. Comput. Phys.*, 375:1339–1364, 2018. 19

Sprekeler, H. On the relation of slow feature analysis and Laplacian eigenmaps. *Neural Comput.*, 23(12):3287–3302, 2011. 1

Stewart, G. W. On the early history of the singular value decomposition. *SIAM Rev. Soc. Ind. Appl. Math.*, 35(4):551–566, December 1993. ISSN 0036-1445, 1095-7200. doi: 10.1137/1035134. 18

Stewart, G. W. Fredholm, Hilbert, Schmidt: three fundamental papers on integral equations. , 2011. URL http://users.umiacs.umd.edu/~stewart/FHS.pdf. 17

Tancik, M., Srinivasan, P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J., and Ng, R. Fourier features let networks learn high frequency functions in low dimensional domains. In *Adv. Neural Inf. Proc. Syst.*, volume 33, pp. 7537–7547, 2020. 33

Tenenbaum, J. B., De Silva, V., and Langford, J. C. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000. 1

Trefethen, L. N. and Bau, D. *Numerical linear algebra*, volume 181. SIAM, 2022. 16

Tsai, Y.-H. H., Wu, Y., Salakhutdinov, R., and Morency, L.-P. Self-supervised learning from a multi-view perspective. In *Int. Conf. Learn. Repr.*, 2020. 18

Turk, M. and Pentland, A. Eigenfaces for recognition. *J. Cogn. Neurosci.*, 3(1):71–86, 1991. 1

Wang, L., Wu, J., Huang, S.-L., Zheng, L., Xu, X., Zhang, L., and Huang, J. An Efficient Approach to Informative Feature Extraction from Multimodal Data. In *Proc. AAAI Conf. Artif. Int.*, volume 33, pp. 5281–5288, July 2019. doi: 10.1609/aaai.v33i01.33015281. 1, 18

Wang, Y. and Xie, H. Computing Multi-Eigenpairs of High-Dimensional Eigenvalue Problems Using Tensor Neural Networks. *arXiv*, May 2023. 19

Weidmann, J. *Linear operators in Hilbert spaces*, volume 68. Springer Science & Business Media, 2012. 2

Williams, C. and Seeger, M. Using the Nyström method to speed up kernel machines. In *Adv. Neural Inf. Proc. Syst.*, volume 13, 2000. 2, 16

Wiskott, L. and Sejnowski, T. J. Slow feature analysis: Unsupervised learning of invariances. *Neural Comput.*, 14(4):715–770, 2002. 1

Wu, J., Wang, S. F., and Perdikaris, P. A dive into spectral inference networks: improved algorithms for self-supervised learning of continuous spectral representations. *Appl. Math. Mech.*, 44(7):1199–1224, July 2023. ISSN 0253-4827, 1573-2754. doi: 10.1007/s10483-023-2998-7. 6, 21, 33

Xu, X. and Zheng, L. Neural feature learning in function space. *J. Mach. Learn. Res.*, 25(142):1–76, 2024. URL http://jmlr.org/papers/v25/23-1202.html. 7, 18

Xu, X., Huang, S.-L., Zheng, L., and Wornell, G. W. An information theoretic interpretation to deep neural networks. *Entropy*, 24(1):135, 2022. 18

Yang, Q., Deng, Y., Yang, Y., He, Q., and Zhang, S. Neural networks based on power method and inverse power method for solving linear eigenvalue problems. *Comput. Math. Appl.*, 147:14–24, October 2023. ISSN 0898-1221. doi: 10.1016/j.camwa.2023.07.013. 19

Yang, X. L., Guo, S. H., Chan, F. T., Wong, K. W., and Ching, W. Y. Analytic solution of a two-dimensional hydrogen atom. I. Nonrelativistic theory. *Phys. Rev. A*,

43(3):1186–1196, February 1991. ISSN 1050-2947. doi: 10.1103/physreva.43.1186. 32

Yelamarthi, S. K., Reddy, S. K., Mishra, A., and Mittal, A. A zero-shot framework for sketch based image retrieval. In *Proc. Eur. Conf. Comput. Vis.*, pp. 300–317, 2018. 8

Zhang, W., Li, T., and Schütte, C. Solving eigenvalue PDEs of metastable diffusion processes using artificial neural networks. *J. Comput. Phys.*, 465:111377, September 2022. ISSN 0021-9991. doi: 10.1016/j.jcp.2022.111377. 19

Appendix

A. On Standard Linear Algebra Techniques

A.1. Empirical SVD and Nyström Method

A standard variational characterization of SVD is based on the following sequence of optimization problems:

$$\begin{aligned} &\underset{\tilde{\phi}_\ell \in \mathcal{F}, \tilde{\psi}_\ell \in \mathcal{G}}{\text{maximize}} \ \langle \tilde{\psi}_\ell | \mathcal{T} \tilde{\phi}_\ell \rangle, \\ &\text{subject to } \ \langle \tilde{\phi}_i | \tilde{\phi}_\ell \rangle = \langle \tilde{\psi}_i | \tilde{\psi}_\ell \rangle = \delta_{i\ell} \quad \forall i \in [\ell] := \{1, \dots, \ell\}. \end{aligned} \tag{10}$$

If $\mathcal{T}$ is compact and the previous $(\ell - 1)$ pairs of functions $\{(\tilde{\phi}_i, \tilde{\psi}_i)\}_{i \in [\ell-1]}$ are the top-$(\ell-1)$ singular functions, then the maximum value of the ℓ-th problem, is attained by the ℓ-th singular functions (ϕ_ℓ, ψ_ℓ) (Bolla, 2013, Proposition A.2.8).

While the notion of SVD and its variational characterization are mathematically well defined, we cannot solve the infinite-dimensional problem (10) directly in general, except a very few cases with known closed-form solutions. Hence, in practice,

a common approach is to perform the SVD of an *empirical kernel matrix* induced by finite points (samples, in learning scenarios). That is, given $x_1, \dots, x_M \sim p(x)$ and $y_1, \dots, y_N \sim p(y)$, we define the empirical kernel matrix $\hat{\mathsf{K}} \in \mathbb{R}^{M \times N}$ as $(\hat{\mathsf{K}})_{ij} := k(x_i, y_j)$. Suppose we perform the (matrix) SVD of $\hat{\mathsf{K}}/\sqrt{MN}$ and obtain the top-L left- and right-singular vectors $\hat{\mathsf{U}} = [\hat{\mathbf{u}}_1, \dots, \hat{\mathbf{u}}_L] \in \mathbb{R}^{M \times L}$ and $\hat{\mathsf{V}} = [\hat{\mathbf{v}}_1, \dots, \hat{\mathbf{v}}_L] \in \mathbb{R}^{N \times L}$ (normalized as $\hat{\mathsf{U}}^\intercal \hat{\mathsf{U}} = M\mathsf{I}$ and $\hat{\mathsf{V}}^\intercal \hat{\mathsf{V}} = N\mathsf{I}$, where I denotes the identity matrix) with the top-L singular values $\hat{\sigma}_1 \geq \ldots \geq \hat{\sigma}_L \geq 0$. Then, for each ℓ, $\hat{\mathbf{u}}_\ell$ and $\hat{\mathbf{v}}_\ell$ approximate the evaluation of ϕ_ℓ and ψ_ℓ at training data, i.e.,

$$\hat{\mathbf{u}}_\ell \approx [\phi_\ell(x_1), \dots, \phi_\ell(x_M)]^\intercal, \quad \hat{\mathbf{v}}_\ell \approx [\psi_\ell(y_1), \dots, \psi_\ell(y_N)]^\intercal, \quad \text{and} \quad \hat{\sigma}_\ell \approx \sigma_\ell.$$

Hence, for $\ell \geq 1$ with $\hat{\sigma}_\ell > 0$, the ℓ-th left-singular function at x can be estimated as

$$\hat{\phi}_\ell(x) := \frac{\hat{\sigma}_\ell^{-1}}{N} \sum_{j=1}^{N} k(x, y_j)(\hat{\mathbf{v}}_\ell)_j, \tag{11}$$

which is a finite-sample approximation of the relation $\phi_\ell(x) = \sigma_\ell^{-1}(K\psi_\ell)(x)$. This is often referred to as the *Nyström method*; see, e.g., (Williams & Seeger, 2000; Bengio et al., 2004).

Performing SVD of the kernel matrix K can be viewed as solving (10) with finite samples in the nonparametric limit. The sample SVD approach is limited, however, due to its memory and computational complexity. The time complexity of full SVD is $O(\min\{MN^2, M^2N\})$ not scalable, but there exist iterative subspace methods that can perform top-L SVD in an efficient way. Note, however, that the data matrix should be stored in memory to run standard SVD algorithms, which may not be feasible for large-scale data. Moreover, while the query complexity $O(N)$ or $O(M)$ of the Nyström method could be reduced by choosing a subset of training data, the challenge posed by the curse of dimensionality can potentially undermine the reliability of the Monte Carlo approximation (11) as an estimator.

A.2. On the Effectiveness of Nesting vs. the Rayleigh–Ritz Method

One may question the advantages of learning the ordered eigenfunctions via nesting compared to first learning the subspace and then determining the order within the subspace using the *Rayleigh–Ritz method* in numerical linear algebra, which is a numerical algorithm to approximate eigenvalues (Trefethen & Bau, 2022). The idea of Rayleigh–Ritz is to use an orthonormal basis of some smaller-dimensional subspace and solve the surrogate eigenvalue problem of smaller dimension projected on the subspace. The quality of the Rayleigh–Ritz approximation depends on the user-defined orthonormal basis $\{|\phi_j\rangle\}_{j=1}^d$. That is, as the basis better captures the desired eigenmodes of the target operator, the approximation becomes more accurate. Otherwise, for example, if an eigenmode is orthogonal to the subspace of the eigenbasis, it cannot be found by this procedure. For completeness, we describe the procedure at the end of this section.

Given this standard tool, one may ask whether it is necessary to learn the ordered singular functions as done by NeuralSVD, SpIN, and NeuralEF. Instead, since by minimizing LoRA objective we can approximately learn the top-L eigensubspace (Theorem 3.1), one can consider applying Rayleigh–Ritz with the learned functions trained by LoRA. Though the idea is valid and the full EVD of $L \times L$ matrix in Rayleigh–Ritz would be virtually at no additional cost, we remark that the two-stage procedure has several drawbacks compared to the direct approach with NeuralSVD. First, note that the learned functions with the LoRA objective are necessarily orthogonal and Gram–Schmidt process should be applied for obtaining the orthonormal basis before Rayleigh–Ritz. Note, however, that Gram–Schmidt becomes nontrivial in function spaces, as we need to compute the inner products and norms of functions at each step. Moreover, computing the inner products $\langle\phi_i|\mathcal{T}\phi_j\rangle$ to compute the reduced operator as described below may introduce an additional estimation error.

More crucially, we empirically verified that NeuralSVD$_{\text{seq}}$ or NeuralSVD$_{\text{jnt}}$ can attain lower subspace distance than that learned by LoRA (without nesting), while being able to correctly ordered orthogonal eigenbasis simultaneously. Since the quality of Rayleigh–Ritz is limited by the quality of the given subspace, if the learned subspace has lower quality, the outcome must be worse. For example, in the 2D hydrogen atom experiment, we observed that the subspace distance over the 16 eigenmodes was $3.56 \times 10^{-4}{}_{\pm 6.60 \times 10^{-5}}$ with LoRA, while $2.12 \times 10^{-4}{}_{\pm 2.09 \times 10^{-5}}$ and $2.06 \times 10^{-4}{}_{\pm 1.91 \times 10^{-5}}$ were attained by NeuralSVD$_{\text{jnt}}$ and NeuralSVD$_{\text{seq}}$, respectively; see Fig. 4(a). Since nesting does not increase complexity compared to the non-nested case via its efficient gradient implementation with masking, we argue that NeuralSVD can be more efficient than the two-stage approach.

Rayleigh–Ritz for Operator EVD. For the ease of exposition, here we describe the procedure for an operator eigenvalue problem. Given a self-adjoint operator $\mathcal{T}$, suppose that we wish to solve an eigenvalue problem

$$\mathcal{T}|\psi\rangle = \lambda|\psi\rangle.$$

Since the problem may be hard to solve directly, the Rayleigh–Ritz method assumes that a set of orthonormal functions $\{|\phi_1\rangle, \ldots, |\phi_d\rangle\}$ for some $d \geq 1$, preferably $d \ll N$, and define $\mathsf{B} \in \mathbb{R}^{d\times d}$ such that $\mathsf{B}_{ij} := \langle\phi_i|\mathcal{T}\phi_j\rangle$. Then, we solve the eigenvalue problem

$$\mathsf{B}\mathbf{y} = \mu\mathbf{y}.$$

Given an eigenpair $(\mu_i, \mathbf{y}_i)$, we compute the Ritz function $|\tilde{\psi}_i\rangle := \sum_{j=1}^{d} y_j|\phi_j\rangle$, and set the Ritz value $\tilde{\lambda}_i := \mu_i$. The output of the Rayleigh–Ritz method are the Ritz pairs $\{(\tilde{\lambda}_i, \tilde{\psi}_i)\}_{i=1}^{d}$.

A.3. Comparison to a Nonparametric Approach

A reader familiar with numerical linear algebra literature may wonder how the parametric approach is compared to the standard techniques. To this end, as a quick comparison, we performed the following baseline experiment, with one of the standard matrix-free techniques called "Locally Optimal Block Preconditioned Conjugate Gradient" (LOBPCG) for finding top-L eigenvalues of large matrices (Knyazev, 2001; 2017). To learn the first $L = 16$ eigenstates of the 2D hydrogen atom, we consider a truncated domain $[-50, 50]$ and discretize each axis by N grid points. We then perform the top-L EVD of the discretized Hamiltonian matrix of size $N \times N$ using LOBPCG (using the PyTorch functionality `torch.lobpcg`). The result is summarized in Fig. 6. In the first panel, we present the relative errors in the estimated eigenvalues in parallel to Fig. 4(a). In the second panel, the blue line summarizes the average absolute relative error for each N. The accuracy improves as N becomes larger as expected in general, but we observe that the quality of estimates of latter eigenvalues become worse with $N = 1600$ than with $N = 800$. Compared to the best result obtained by NeuralSVD$_{\text{seq}}$(512) (indicated by the red dashed horizontal line), this naive approach may take substantially more time to achieve comparable accuracy as it may not scale well as N increases as shown in the third panel. This briefly showcases a possible advantage of NeuralSVD (or the parametric approach at large) over the matrix-based approach.

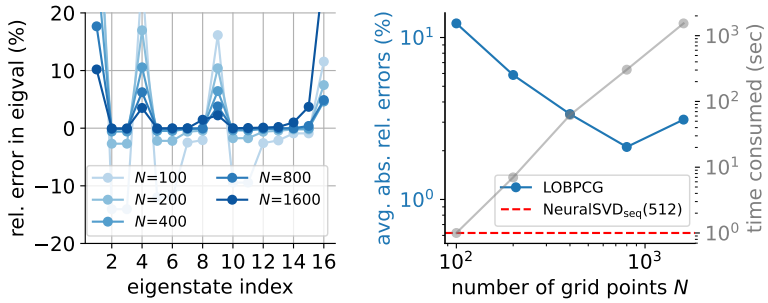


Figure 6: Performance of LOBPCG 2D hydrogen experiment.

We remark, however, some caveats in this comparison. First, the LOBPCG implementation we used here might not be fully optimized and there could exist a version that exhibits better scalability. The runtime could be also drastically reduced by using GPU or parallel machines as such numerical linear algebra algorithms are known to be very well optimized for such resources, while the current experiment was run on a CPU machine. Second, the discretization is rather naive and a more sophisticated discretization with some choice of orthonormal basis could lead to a better solution. Third, the relatively large error in the first eigenvalue estimate is seemingly due to the non-differentiable cusp of the first eigenfunction, and thus the discretization approach could behave better for other examples.

B. Related Work

B.1. General Literature Review

B.1.1. Low-Rank Approximation

The theory of low-rank approximation was initially developed to solve partial differential equations (PDEs), including Schmidt (1907)'s work; see, e.g., (Stewart, 2011). A special case for finite-dimensional matrices was independently

discovered later by Eckart & Young (1936) and Mirsky (1960), which are perhaps better known in the literature. We refer an interested reader to (Stewart, 1993) for detailed historical remarks. For matrices, Mirsky (1960) extended the low-rank approximation theory to any unitarily invariant norms. While it would be interesting to extend the proposed framework in the current paper for other norms, they do not seem to easily admit an optimizable objective function.

B.1.2. Canonical Dependence Kernels

Interestingly, there exists a rich literature on decomposing canonical dependence kernels (CDK); see Sec. 4.2. The CDK has a close relationship to the Hirschfeld–Gebelein–Rényi (HGR) maximal correlation (Hirschfeld, 1935; Gebelein, 1941; Rényi, 1959). Note that the first singular functions are trivially constant functions, and the corresponding singular value σ_1 is always 1. When $L = 2$, the second singular value is known as the HGR maximal correlation. In general, the optimization problem can be understood as the high-dimensional extension of the maximal correlation; for given $L \geq 2$, the optimal functions $\boldsymbol{\phi}^\star$ and $\boldsymbol{\psi}^\star$ are the optimal L-dimensional projections of $x \sim p(x)$ and $y \sim p(y)$ that are maximally correlated. The CDK plays an important role in learning applications and has been frequently redeveloped, bearing different names, e.g., correspondence analysis (Greenacre, 1984) and *principal inertia components* (Hsu et al., 2019; 2022) for finite alphabets, the contrastive kernel (HaoChen et al., 2021; Deng et al., 2022b) and the pointwise dependence (Tsai et al., 2020) in the self-supervised representation learning setup.

The nonnested objective $\mathcal{L}_{\mathsf{LoRA}}(\mathbf{f}, \mathbf{g})$ for CDK was proposed and studied in Wang et al. (2019) and related works, e.g., (Xu et al., 2022). The H-score was first introduced by Wang et al. (2019) who coined the term H-score (or Soft-HGR), for learning HGR maximal correlation functions with neural networks. It also appeared as a local approximation to log-loss of classification deep neural networks (Xu et al., 2022). We mention in passing that the nonnested objective has been recently proposed independently under the name of the *spectral contrastive loss* (HaoChen et al., 2021), specifically when the CDK is induced by the random augmentation from the standard self-supervised representation learning setup. A recent work (Hu & Principe, 2022) proposes to learn features of two modalities based on the EVD of the so-called cross density ratio, which can be equivalently understood as the CDK of a symmetrized joint distribution. This paper, however, also only aims to characterize the top-L subspace without the structure. Their optimization problem is based on minimizing the log-determinant of a normalized autocorrelation function; compared to our LoRA objective, the resulting optimization inherently suffers from biased gradients, which may lead to issues in practice.

B.1.3. Nesting

The idea of *joint* nesting was first introduced by Xu & Zheng (2024) as a general construction to decompose multivariate dependence, which is equivalent to CDK in our terminology, for learning structured features; see the paper for more detailed discussion. The joint nesting proposed in this paper can be understood as an extension of the idea to general operators beyond CDK. While the idea of sequential nesting with LoRA is new, we observe that it conceptually resembles the idea of a class of streaming PCA algorithms such as (Sanger, 1989; Gemp et al., 2021), in which the $(\ell + 1)$-th eigenvector is updated under the assumption that the estimates for the first ℓ eigenvectors are accurate.

We note that a recent work (Kusupati et al., 2022) proposed learning a structured representation using a concept similar to the joint nesting technique introduced in our current paper. The method is referred to as *Matryoshka Representation Learning* (MRL). The key difference in MRL is that it assumes a labeled image dataset and uses the multi-class softmax cross-entropy loss function as its constituent loss function for "nesting". Compared with MRL, the features learned by our CDK-based representation learning framework are interpretable as singular functions of the dependence kernel. This fundamental relation provides the learned features with theoretical guarantees, such as the uncorrelatedness of features. It is also worth noting that our features defined by the global minimizer of $\text{NestedLoRA}_{\mathsf{jnt}}$ is invariant to the choice of weights, while a different choice of such weights in MRL would characterize different features. Our framework is also not restricted to the supervised case, as demonstrated in the cross-domain retrieval example (Sec. 4.2).

B.1.4. Other Correlation Analysis Methods

There exists another line of related literature in correlation analysis. Deep canonical correlation analysis (DCCA) (Andrew et al., 2013) can be understood as solving a restricted HGR maximal correlation problem, searching over a class of neural networks instead of all measurable functions. The DCCA objective function, however, requires a nontrivial optimization technique and cannot be easily extended to find higher modes. The correspondence-analysis neural network (CA-NN) (Hsu et al., 2019) also aims to decompose the CDK based on a different optimization framework, but they involve the L-th

Ky-Fan norm and the inversion of $L \times L$ matrix, which complicate the optimization procedure. Instead of deploying neural networks, Michaeli et al. (2016) proposed to decompose an empirical CDK matrix constructed by the Gaussian kernel density estimators and coined the method as nonparametric canonical correspondence analysis (NCCA).

B.1.5. Neural-Network-Based Methods for Eigenvalue Problems

As alluded to earlier, there exists a rather separate line of work on solving eigenvalue problems using neural networks for solving linear PDEs that are in the form of an eigenvalue problem (EVP) in the physics or scientific computing community. Given the vastness of the literature and the rapid evolution of the field, providing a comprehensive overview is challenging. Nonetheless, we will emphasize key concepts and ideas.

Computational Physics Literature. The idea of using neural networks for solving PDEs which can be reduced to eigenvalue problems dates back to (Lagaris et al., 1997), where an explicit Gram–Schmidt process was proposed to attain multiple eigenstates. Unlike the methods in the machine learning literature, many recent works rely on minimizing the sum of residual losses, mostly in the form of $\|(\mathcal{T} - \lambda_\ell I)\phi_\ell\|$ where λ_ℓ needs be also optimized or estimated from $|\phi_\ell\rangle$, with regularization terms that penalize the normalization of and the orthogonality between the parametric functions; see (Bar & Sochen, 2019; Ben-Shaul et al., 2023; Li et al., 2021a; Zhang et al., 2022; Liu et al., 2023; Wang & Xie, 2023; Guo & Ming, 2023; Mattheakis et al., 2022; Liu et al., 2023; Jin et al., 2020; 2022; Holliday et al., 2023). We note that NeuralSVD is distinct from these regularization-based approaches: while regularization-based approaches are often susceptible to the choice of regularization parameters, NeuralSVD, which utilizes nesting techniques, characterizes the ordered eigenbasis as its global optimizer without tuning any hyperparameter in the objective functions.

Other approaches include: Han et al. (2020) proposes a stochastic differential equation framework that can learn the first mode of an eigenvalue problem; Yang et al. (2023) propose a way to use neural networks for power and inverse power methods; Li & Ying (2021) proposed a semigroup method for high dimensional elliptic PDEs and eigenvalue problems using neural networks. More broadly, there exist other deep-learning-based solvers for general PDEs beyond EVP PDEs such as deep Ritz method (E & Yu, 2018), deep Galerkin method (Sirignano & Spiliopoulos, 2018), and Fourier neural operator (Li et al., 2021b).

Quantum Chemistry Literature. Quantum chemistry has witnessed rapid recent advancements in this particular direction. While the main problem in quantum chemistry is to solve the TISE of a given electronic system, the problem size grows rapidly: the domain has $3N$ dimension with N electrons, and thus the complexity of solving TISE exponentially blows up even with N of a moderate size. Therefore, the development in this domain has been focused on developing a new neural network architecture (called neural network *ansatzes*) that better embed physical inductive bias for more expressivity. Representative works include (Carleo & Troyer, 2017), SchNet (Schütt et al., 2017), Fermionic neural networks (Choo et al., 2020), FermiNet (Pfau et al., 2020), PauliNet (Hermann et al., 2020), and DeepErwin (Gerard et al., 2022); see a comprehensive, recent review paper (Hermann et al., 2022) for the overview of the field.

In most, if not all, of the works, the quantum Monte Carlo (QMC), also known as variational Monte Carlo (VMC) (Cuzzocrea et al., 2020), has been used as the de facto. QMC is essentially a special way to minimize the Rayleigh quotient to obtain the ground state energy. Until recently, most of the works focused on the ground state (i.e., the first bottom mode); a few recent exceptions are (Entwistle et al., 2023; Pfau et al., 2023), which proposed variations of QMC for excited states. We believe that applying the proposed NestedLoRA framework to quantum chemistry problem can be an exciting research direction.

B.1.6. Spectral Pollution

We remark in passing that in the numerical linear algebra literature, there exists a phenomenon called *spectral pollution*, which refers to spurious eigenvalues introduced by discretizing an infinite-dimensional operator with respect to a fixed orthonormal basis (Davies & Plum, 2004), and it is an active research area to address the issue; see, e.g., (Colbrook et al., 2021) for a recent attempt. In principle, as our framework directly optimizes parametric eigenfunctions to fit the underlying eigenfunctions, we would not encounter such an issue due to discretization, provided that the parametric functions are sufficiently expressive. In experiments, we also did not observe any spurious eigenvalues with any of the methods for parametric eigenfunctions for the hydrogen atom.

Table 2: Comparison with SpIN (Pfau et al., 2019) and NeuralEF (Deng et al., 2022b).

	SpIN	**NeuralEF**	**NeuralSVD**
Goal	EVD	EVD	SVD/EVD
(a) To handle orthonormality constraints	Cholesky decomposition	function normalization	-
(b) To remove bias in gradient estimation	bi-level optimization; need to store Jacobian	large batch size	-

B.2. In-Depth Review of SpIN and NeuralEF

B.2.1. SPIN

Suppose that we wish to learn the top-L eigenpairs of a linear operator $\mathcal{T}$. For trial eigenfunctions $\hat{\phi}_1, \dots, \hat{\phi}_L$, define two $L \times L$ matrices Σ and Π, which we call the gram matrix and the quadratic form matrix, respectively, as follows:

$$\Sigma_{\ell\ell'} := \langle \hat{\phi}_\ell | \hat{\phi}_{\ell'} \rangle \quad \text{and} \quad \Pi_{\ell\ell'} := \langle \hat{\phi}_\ell | \mathcal{T} \hat{\phi}_{\ell'} \rangle.$$

Based on the trace maximization framework, we can solve the following optimization problem:

$$\underset{\hat{\phi}_1,\dots,\hat{\phi}_L \in \mathcal{F}}{\text{maximize}} \operatorname{tr}(\Sigma^{-1}\Pi).$$

Let $\Sigma = \mathsf{L}\mathsf{L}^\intercal$ be the Cholesky decomposition of Σ, where L is a lower-triangular matrix. Define $\Lambda := \mathsf{L}^{-1}\Pi\mathsf{L}^{-\intercal} \in \mathbb{R}^{L\times L}$. By the property of trace, we can write

$$\operatorname{tr}(\Sigma^{-1}\Pi) = \operatorname{tr}((\mathsf{L}\mathsf{L}^\intercal)^{-1}\Pi) = \operatorname{tr}(\mathsf{L}^{-1}\Pi\mathsf{L}^{-\intercal}) = \operatorname{tr}(\Lambda) = \sum_{\ell=1}^{L} \Lambda_{\ell\ell}.$$

Optimization with Masked Gradient. The key idea behind the SpIN optimization framework is in the following lemma.

Lemma B.1. *For each $\ell = 1, \dots, L$, $\Lambda_{\ell\ell}$ is only a function of $\hat{\phi}_1, \dots, \hat{\phi}_\ell$.*

Proof. It immediately follows from the upper triangular property of L^{-1}. □

Assuming that $\hat{\phi}_1, \dots, \hat{\phi}_{\ell-1}$ learn the top-$(\ell-1)$ eigen-subspace, SpIN updates $\hat{\phi}_\ell$ to only maximize $\Lambda_{\ell\ell}$, i.e., based on the gradient $\partial_{\hat{\phi}_\ell}\Lambda_{\ell\ell}$ for each ℓ. Once optimized, the learned functions can be orthogonalized by $\mathsf{L}^{-1}\hat{\boldsymbol{\phi}}$. Let L_Σ denote the Cholesky factor for a matrix Σ and define

$$\mathsf{A}_{\Sigma,\Pi} := \mathsf{L}_\Sigma^{-\intercal}\mathsf{triu}(\mathsf{L}_\Sigma^{-1}\Pi\mathsf{L}_\Sigma^{-\intercal}\operatorname{diag}(\mathsf{L}_\Sigma)^{-1}).$$

Then, the *masked* gradient can be collectively written as

$$-\tilde{\partial}_\theta \operatorname{tr}(\Lambda) = -\mathbb{E}_{p(x)}\Big[(\mathcal{T}\hat{\boldsymbol{\phi}})(X)^\intercal \mathsf{L}_\Sigma^{-1} \operatorname{diag}(\mathsf{L}_\Sigma)^{-1} \frac{\partial \hat{\boldsymbol{\phi}}(X)}{\partial\theta}\Big] + \mathbb{E}_{p(x)}\Big[\hat{\boldsymbol{\phi}}(X)^\intercal \mathsf{A}_{\Sigma,\Pi} \frac{\partial \hat{\boldsymbol{\phi}}(X)}{\partial\theta}\Big]. \tag{12}$$

See eq. (25) of (Pfau et al., 2019) for the original expression with derivation. A naive estimator of this gradient with minibatch samples would be to plug in the empirical (unbiased) estimates of Σ and Π, which are

$$\hat{\Sigma} := \hat{\mathbb{E}}_{p(x)}[\hat{\boldsymbol{\phi}}(X)\hat{\boldsymbol{\phi}}(X)^\intercal] \quad \text{and} \quad \hat{\Pi} := \hat{\mathbb{E}}_{p(x)}[\hat{\boldsymbol{\phi}}(X)(\mathcal{T}\hat{\boldsymbol{\phi}})(X)^\intercal].$$

Note, however, the resulting gradient estimate is biased, since L_Σ, L_Σ^{-1}, and Σ^{-1} are not linear in Σ.

Bi-level Stochastic Optimization for Unbiased Gradient Estimates. To detour the issue with the biased gradient estimate, Pfau et al. (2019) proposed to plug-in exponentially weighted moving average (EWMA) of two statistics into the expression, which can be understood as an instance of a bi-level stochastic optimization procedure with unbiased gradient estimates. To motivate the approach, we rewrite the second term of (12) as

$$\mathbb{E}_{p(x)}\Big[\hat{\boldsymbol{\phi}}(X)^\intercal \mathsf{A}_{\Sigma,\Pi} \frac{\partial \hat{\boldsymbol{\phi}}(X)}{\partial\theta}\Big] = \operatorname{tr}\Big(\mathsf{A}_{\Sigma,\Pi}\mathbb{E}_{p(x)}\Big[\frac{\partial \hat{\boldsymbol{\phi}}(X)}{\partial\theta}\hat{\boldsymbol{\phi}}(X)^\intercal\Big]\Big).$$

Based on the expression, we maintain the EWMAs of Σ and $\mathbb{E}_{p(x)}\left[\frac{\partial \hat{\boldsymbol{\phi}}(X)}{\partial \theta}\hat{\boldsymbol{\phi}}(X)^\intercal\right]$, which are denoted as $\bar{\Sigma}$ and $\bar{\mathbf{J}}$, and updated via minibatch samples as follows:

$$\bar{\Sigma} \leftarrow \beta\bar{\Sigma} + (1-\beta)\hat{\Sigma}, \tag{13}$$

$$\bar{\mathbf{J}} \leftarrow \beta\bar{\mathbf{J}} + (1-\beta)\hat{\mathbb{E}}_{p(x)}\left[\frac{\partial \hat{\boldsymbol{\phi}}(X)}{\partial \theta}\hat{\boldsymbol{\phi}}(X)^\intercal\right]. \tag{14}$$

Here $\beta \in [0,1]$ is the decay parameter for EWMA. Now, given these statistics, we update the parameter θ by the following gradient estimate with minibatch samples:

$$-\widehat{\partial}_\theta \operatorname{tr}(\Lambda) = -\hat{\mathbb{E}}_{p(x)}\left[(\mathcal{T}\hat{\boldsymbol{\phi}})(X)^\intercal \mathsf{L}_{\bar{\Sigma}}^{-1} \operatorname{diag}(\mathsf{L}_{\bar{\Sigma}})^{-1}\frac{\partial \hat{\boldsymbol{\phi}}(X)}{\partial \theta}\right] + \operatorname{tr}(\mathsf{A}_{\bar{\Sigma},\hat{\Pi}}\bar{\mathbf{J}}).$$

Note that the randomness in the second term is in $\hat{\Pi}$ and the second term is linear in $\hat{\Pi}$. After all, the estimate is unbiased given $\bar{\Sigma}$ and $\bar{\mathbf{J}}$.

Discussion. SpIN is a pioneering work, being the first parametric framework to perform the top-L EVD of a self-adjoint operator with parametric eigenfunctions. However, the derivation of the masked gradient is rather involved, and the resulting algorithm's complexity is not favorably scaling in L. In terms of the computational complexity, the Cholesky decomposition step that takes $O(L^3)$ for each iteration is not scalable in L. Also, due to the bi-level stochastic optimization for unbiased gradient estimates, SpIN needs to maintain a separate copy of the Jacobian (14), which may consume significant memory with large networks. The decay parameter in the bi-level stochastic optimization is another sensitive hyperparameter to be tuned in the framework. Finally, we remark that the idea of masked gradient is similar to the sequential nesting, and thus when it is applied to a shared parameterization, it cannot guarantee a desired optimization behavior.

SpIN-X. There exists a follow-up work of SpIN that proposed an alternative optimization method with several practical optimization techniques (Wu et al., 2023). As the paper does not coin a term for the proposed method, we call it SpIN-X here. The proposed method is based on the following modified objective function

$$\mathcal{L} = \frac{1}{L}\Big\{-w_0\sum_{\ell=1}^{L} a_\ell \Lambda_{\ell\ell} + \sum_{\ell=1}^{L} w_\ell \|(\mathcal{T} - \Lambda_{\ell\ell} I)\hat{\phi}_\ell\|^2\Big\}.$$

Here, the weights $w_0, \ldots, w_L$ are defined as $w_\ell := (K_0 + \ldots + K_L)/K_\ell$, where $K_\ell := \mathsf{sg}(\|\,\partial_\theta \mathcal{L}_\ell\|_2)$, and $\Lambda_{\ell\ell}$ are eigenvalues still obtained from the Cholesky decomposition steps. Though the experimental results in (Jin et al., 2022) show improved results over SpIN, some optimization techniques such as balanced gradients are nontrivial to apply, and thus we do not include a comparison with this approach.

B.2.2. NeuralEF

In essence, NeuralEF (Deng et al., 2022a) starts from the following characterization of eigenfunctions, which can be understood as a sequential version of (2).

Proposition B.2. *Let $\mathcal{T}\colon \mathcal{F} \to \mathcal{F}$ be a linear, self-adjoint operator, where $\mathcal{F}$ is a Hilbert space. Given functions $\tilde{\phi}_1, \ldots, \tilde{\phi}_{\ell-1} \in \mathcal{F}$, consider the optimization problem*

$$(P_\ell) \quad \begin{aligned} &\underset{\tilde{\phi}_\ell \in \mathcal{F}}{\text{maximize}} \;\; \langle \tilde{\phi}_\ell | \mathcal{T}\tilde{\phi}_\ell\rangle \\ &\text{subject to} \;\; \langle \tilde{\phi}_\ell | \tilde{\phi}_i\rangle = \delta_{\ell i} \quad \forall 1 \le i \le \ell. \end{aligned}$$

If $\tilde{\phi}_1, \ldots, \tilde{\phi}_{\ell-1}$ are the top $\ell-1$ eigenfunctions of the operator $\mathcal{T}$, then the solution $\tilde{\phi}_\ell$ of the optimization problem (P_ℓ) is the ℓ-th eigenfunction.

To avoid the explicit orthogonality constraint, NeuralEF proposes to solve the following optimization problem, generalizing the formulation of EigenGame (Gemp et al., 2021) for operators:

$$(P'_\ell) \quad \begin{aligned} &\underset{\tilde{\phi}_\ell \in \mathcal{F}}{\text{minimize}} \;\; \mathcal{L}_\ell(\boldsymbol{\phi}_{1:\ell}) := -\langle \tilde{\phi}_\ell | \mathcal{T}\tilde{\phi}_\ell\rangle + \sum_{i=1}^{\ell-1} \frac{\langle \tilde{\phi}_\ell | \mathcal{T}\tilde{\phi}_i\rangle}{\langle \tilde{\phi}_i | \mathcal{T}\tilde{\phi}_i\rangle} \\ &\text{subject to} \;\; \langle \tilde{\phi}_\ell | \tilde{\phi}_\ell\rangle = 1. \end{aligned} \tag{15}$$

Replacing the constraints $\langle\tilde{\phi}_\ell|\tilde{\phi}_i\rangle = 0$ as $\langle\tilde{\phi}_i|\mathcal{T}\tilde{\phi}_i\rangle^2 = 0$ for each $1 \le i \le \ell - 1$, we can view (P'_ℓ) as a relaxed optimization problem of (P_ℓ). Here, $\langle\tilde{\phi}_i|\mathcal{T}\tilde{\phi}_i\rangle^{-1}$ plays the role of a Lagrangian multiplier for the i-th constraint. With this specific choice of weights, this partially unconstrained optimization problem has the same guarantee (Proposition B.2) for (P_ℓ) as follows:

Theorem B.3. *If $\tilde{\phi}_1, \dots, \tilde{\phi}_{\ell-1}$ are the top $\ell - 1$ eigenfunctions $\phi_1, \dots, \phi_{\ell-1}$ of $\mathcal{T}$, then the solution $\tilde{\phi}_\ell$ of the optimization problem (P'_ℓ) is the ℓ-the eigenfunction.*

Informal proof. For the sake of simplicity, we assume that there are only m finite eigenvalues $\lambda_1, \dots, \lambda_m$ and $\ell + 1 \le m$. Let $\phi_1, \phi_2, \dots$ be the eigenfunctions of K which form an othornormal basis of $\mathcal{L}^2_{p(x)}(\mathcal{X})$. We first write a function $\tilde{\phi}_\ell$ as a linear combination of the eigenfunctions

$$\tilde{\phi}_\ell(x) = \sum_{i=1}^{\infty} \langle\tilde{\phi}_\ell|\phi_i\rangle.$$

Then, we can readily observe that

$$\begin{aligned}
\langle\tilde{\phi}_\ell|\mathcal{T}\tilde{\phi}_\ell\rangle &= \sum_{i=1}^{m} \lambda_i \langle\tilde{\phi}_\ell|\phi_i\rangle^2, \\
\langle\tilde{\phi}_\ell|\mathcal{T}\phi_i\rangle &= \lambda_i \langle\tilde{\phi}_\ell|\phi_i\rangle, \\
\langle\phi_i|\mathcal{T}\phi_i\rangle &= \lambda_i.
\end{aligned}$$

Therefore, the objective becomes

$$\begin{aligned}
-\langle\tilde{\phi}_\ell|\mathcal{T}\tilde{\phi}_\ell\rangle + \sum_{i=1}^{\ell-1} \frac{\langle\tilde{\phi}_\ell|\mathcal{T}\phi_i\rangle^2}{\langle\phi_i|\mathcal{T}\phi_i\rangle} &= -\sum_{i=1}^{m} \lambda_i \langle\tilde{\phi}_\ell, \phi_i\rangle^2 + \sum_{i=1}^{\ell-1} \lambda_i \langle\tilde{\phi}_\ell, \phi_i\rangle^2 \\
&= -\sum_{i=\ell}^{m} \lambda_i \langle\tilde{\phi}_\ell, \phi_i\rangle^2,
\end{aligned}$$

which implies that the objective is uniquely minimized when $\langle\tilde{\phi}_\ell, \phi_i\rangle = \delta_{i\ell}$ for $i \ge \ell$, i.e., when $\tilde{\phi}_\ell$ is the ℓ-th eigenfunction ϕ_ℓ. □

Hence, solving the sequence of optimization problems (P'_ℓ) leads to finding the eigenfunctions in order. To emulate to solve the sequential optimization, Deng et al. (2022a) proposed to solve

$$\begin{aligned}
&\underset{\tilde{\phi}_1, \dots, \tilde{\phi}_L \in \mathcal{F}}{\text{minimize}} \sum_{\ell=1}^{L} \Big\{ -\langle\tilde{\phi}_\ell|\mathcal{T}\tilde{\phi}_\ell\rangle + \sum_{i=1}^{\ell-1} \frac{\langle\tilde{\phi}_\ell|\mathcal{T}\mathsf{sg}(\tilde{\phi}_i)\rangle^2}{\langle\mathsf{sg}(\tilde{\phi}_i)|\mathcal{T}\mathsf{sg}(\tilde{\phi}_i)\rangle} \Big\} \\
&\text{subject to } \langle\tilde{\phi}_\ell|\tilde{\phi}_\ell\rangle = 1 \text{ for } 1 \le \ell \le L.
\end{aligned}$$

Here, sg denotes the stop-gradient operation, and thus this is not a properly defined optimization problem, rather defining an optimization procedure. It is worth emphasizing that the minimization procedure no longer guarantees a structured solution if the stop gradient operations are removed. To satisfy the normalization constraints, NeuralEF uses the L_2-batch normalization during training.

$$\partial_{\phi_\ell} \mathcal{L}_\ell(\boldsymbol{\phi}_{1:\ell}) = 4\Big\{ -|\mathcal{T}\phi_\ell\rangle + \sum_{i=1}^{\ell-1} \frac{\langle\phi_i|\mathcal{T}\phi_\ell\rangle}{\langle\phi_i|\mathcal{T}\phi_i\rangle} |\mathcal{T}\phi_i\rangle \Big\}. \tag{16}$$

Discussion. NeuralEF improves SpIN in general, providing a simpler optimization procedure, i.e., without the costly Cholesky decomposition steps and the Jacobian updates. The game-theoretic formulation that stemmed from EigenGame (Gemp et al., 2021) is similar to the idea of sequential nesting, and it might be problematic when applied to a shared parameterization as the sequential nesting is. The crucial difference of NeuralEF is that the ℓ-th objective of NeuralEF has a guarantee only if the previous $(\ell - 1)$ eigenfunctions are well learned, whereas the LoRA objective can characterize the eigensubspace and thus we can apply the joint nesting for a shared parameterization. Moreover, NestedLoRA can naturally handle SVD.

An Unbiased-Gradient Variation. We note that in the streaming PCA literature, the authors of EigenGame (Gemp et al., 2021) proposed an unbiased variant of the original EigenGame in their subsequent work (Gemp et al., 2022). Following the same idea, one can easily think of an unbiased variant of NeuralEF, which corresponds to the following gradient:

$$\partial_{\phi_\ell}\mathcal{L}_\ell(\boldsymbol{\phi}_{1:\ell}) = 4\Big\{-|\mathcal{T}\phi_\ell\rangle + \sum_{i=1}^{\ell-1}\langle\phi_i|\phi_\ell\rangle|\mathcal{T}\phi_i\rangle\Big\},$$
$$\partial_{\phi_\ell}\mathcal{L}_\ell(\boldsymbol{\phi}_{1:\ell}) = 4\Big\{-|\mathcal{T}\phi_\ell\rangle + \sum_{i=1}^{\ell-1}\langle\phi_i|\mathcal{T}\phi_\ell\rangle|\phi_i\rangle\Big\}.$$

In our experiment, we used this variant instead of the original (16), as we found that the original NeuralEF performs much worse than its variant. In the current manuscript, we show that NeuralSVD can even outperform the improved version of NeuralEF.

C. Technical Details and Deferred Proofs

C.1. Derivation of the Low-Rank Approximation Objective

Recall that we define the LoRA objective as

$$\mathcal{L}_{\mathsf{LoRA}}(\mathbf{f}_{1:L}, \mathbf{g}_{1:L}) := -2\sum_{\ell=1}^{L}\langle g_\ell|\mathcal{T}f_\ell\rangle + \sum_{\ell=1}^{L}\sum_{\ell'=1}^{L}\langle f_\ell|f_{\ell'}\rangle\langle g_\ell|g_{\ell'}\rangle.$$

When $\mathcal{T}$ is a compact operator, the LoRA objective can be derived as the approximation error of $\mathcal{T}$ via a low-rank expansion $\sum_{\ell=1}^{L}|f_\ell\rangle\langle g_\ell|$ measured in the squared Hilbert–Schmidt norm. For a linear operator $\mathcal{T}\colon \mathcal{F}\to\mathcal{G}$ for Hilbert spaces $\mathcal{F}$ and $\mathcal{G}$, the *Hilbert–Schmidt norm* $\|\mathcal{T}\|_{\mathsf{HS}}$ of an operator $\mathcal{T}$ is defined as

$$\|\mathcal{T}\|_{\mathsf{HS}}^2 := \sum_{i\in I}\|\mathcal{T}\phi_i\|^2$$

for an orthonormal basis $\{\phi_i\colon i\in I\}$ of the Hilbert space $\mathcal{F}$. Note that the Hilbert–Schmidt norm is well-defined in that it is independent of the choice of the orthonormal basis. When $\mathcal{F}$ and $\mathcal{G}$ are finite-dimensional, i.e., when $\mathcal{T}$ is a matrix, it boils down to the Frobenius norm. When $\|\mathcal{T}\|_{\mathsf{HS}}<\infty$, $\mathcal{T}$ is said to be *compact*.

Lemma C.1. *If $\mathcal{T}$ is compact, then*

$$\mathcal{L}_{\mathsf{LoRA}}(\mathbf{f}_{1:L}, \mathbf{g}_{1:L}) = \Big\|\mathcal{T} - \sum_{\ell=1}^{L}|g_\ell\rangle\langle f_\ell|\Big\|_{\mathsf{HS}}^2 - \|\mathcal{T}\|_{\mathsf{HS}}^2. \tag{17}$$

Proof. Pick an orthonormal basis $\{\phi_i\colon i\in I\}$ of $\mathcal{T}$. Note that $\sum_{i\in I}|\phi_i\rangle\langle\phi_i| = I$, where I denotes the identity operator. Hence, we have

$$\begin{aligned}
\Big\|\mathcal{T} - \sum_{\ell=1}^{L}|g_\ell\rangle\langle f_\ell|\Big\|_{\mathsf{HS}}^2 - \|\mathcal{T}\|_{\mathsf{HS}}^2 &= \sum_{i\in I}\Big\|\mathcal{T}|\phi_i\rangle - \sum_{\ell=1}^{L}|g_\ell\rangle\langle f_\ell|\phi_i\rangle\Big\|^2 - \sum_{i\in I}\|\mathcal{T}|\phi_i\rangle\|^2 \\
&= \sum_{i\in I}\Big(-2\sum_{\ell=1}^{L}\langle f_\ell|\phi_i\rangle\langle g_\ell|\mathcal{T}|\phi_i\rangle + \sum_{\ell=1}^{L}\sum_{\ell'=1}^{L}\langle f_\ell|\phi_i\rangle\langle f_{\ell'}|\phi_i\rangle\langle g_\ell|g_{\ell'}\rangle\Big) \\
&= -2\sum_{\ell=1}^{L}\Big\langle g_\ell\Big|\mathcal{T}\Big(\sum_{i\in I}|\phi_i\rangle\langle\phi_i|\Big)\Big|f_\ell\Big\rangle + \sum_{\ell=1}^{L}\sum_{\ell'=1}^{L}\Big\langle f_\ell\Big|\Big(\sum_{i\in I}|\phi_i\rangle\langle\phi_i|\Big)\Big|f_{\ell'}\Big\rangle\langle g_\ell|g_{\ell'}\rangle \\
&= -2\sum_{\ell=1}^{L}\langle g_\ell|\mathcal{T}f_\ell\rangle + \sum_{\ell=1}^{L}\sum_{\ell'=1}^{L}\langle f_\ell|f_{\ell'}\rangle\langle g_\ell|g_{\ell'}\rangle \\
&= \mathcal{L}_{\mathsf{LoRA}}(\mathbf{f}_{1:L}, \mathbf{g}_{1:L}).
\end{aligned}$$

□

Though this relationship (17) holds only for a compact operator, we remark that the LoRA objective (3) is well-defined for any operator $\mathcal{T}$.

C.2. Proof of Theorem 3.1

From Lemma C.1, Theorem 3.1 follows as a corollary of Schmidt's LoRA theorem, stated below. □

Theorem C.2 (Schmidt (1907)). *Suppose that $\mathcal{T}\colon \mathcal{F} \to \mathcal{G}$ is a compact operator with $\{(\sigma_\ell, f_\ell, g_\ell)\}_{\ell=1}^{\infty}$ as its singular triplets. Define*

$$(\mathbf{f}^\star, \mathbf{g}^\star) := \operatorname*{arg\,min}_{f_\ell \in \mathcal{F}, g_\ell \in \mathcal{G},\ \ell \in [L]} \left\| \mathcal{T} - \sum_{\ell=1}^{L} |g_\ell\rangle\langle f_\ell| \right\|_{\mathsf{HS}}^2.$$

If $\sigma_L > \sigma_{L+1}$, we have

$$\sum_{\ell=1}^{L} |g_\ell^\star\rangle\langle f_\ell^\star| = \sum_{\ell=1}^{L} \sigma_\ell |\psi_\ell\rangle\langle\phi_\ell|.$$

C.3. Proof of Theorem 3.2 (Sequential Nesting)

Recall that we assume

$$\sum_{i=1}^{\ell-1} |g_i\rangle\langle f_i| = \sum_{i=1}^{\ell-1} \sigma_i |\psi_i\rangle\langle\phi_i|.$$

By Lemma C.1, the LoRA objective can be written as

$$\begin{aligned}
\mathcal{L}_{\mathsf{LoRA}}(\mathbf{f}_{1:\ell}, \mathbf{g}_{1:\ell}) &= \left\| \mathcal{T} - \sum_{i=1}^{\ell} |g_i\rangle\langle f_i| \right\|_{\mathsf{HS}}^2 - \|\mathcal{T}\|_{\mathsf{HS}}^2 \\
&= \left\| \sum_{i\geq 1} \sigma_i |\psi_i\rangle\langle\phi_i| - \sum_{i=1}^{\ell-1} \sigma_i |\psi_i\rangle\langle\phi_i| - |g_\ell\rangle\langle f_\ell| \right\|_{\mathsf{HS}}^2 - \|\mathcal{T}\|_{\mathsf{HS}}^2 \\
&= \left\| \sum_{i\geq \ell} \sigma_i |\psi_i\rangle\langle\phi_i| - |g_\ell\rangle\langle f_\ell| \right\|_{\mathsf{HS}}^2 - \|\mathcal{T}\|_{\mathsf{HS}}^2.
\end{aligned}$$

Hence, minimizing $\mathcal{L}_{\mathsf{LoRA}}(\mathbf{f}_{1:\ell}, \mathbf{g}_{1:\ell})$ with respect to (f_ℓ, g_ℓ) is equivalent to minimizing the LoRA objective $\mathcal{L}_{\mathsf{LoRA}}(f_\ell, g_\ell; \mathcal{T}_{\geq \ell})$ defined with respect to the truncated operator $\mathcal{T}_{\geq \ell} := \sum_{i\geq \ell} \sigma_i |\psi_i\rangle\langle\phi_i|$. Since $\sigma_\ell > \sigma_{\ell+1}$, by Schmidt's theorem (Theorem C.2), the global optimizer must satisfy $|g_\ell\rangle\langle f_\ell| = \sigma_\ell |\psi_\ell\rangle\langle\phi_\ell|$. □

C.4. Proof of Theorem 3.3 (Joint Nesting)

We first prove the following lemma; Theorem 3.3 readily follows as a corollary.

Lemma C.3. *Suppose that all the nonzero singular values of the target kernel are distinct. If $\sigma_\ell > \sigma_{\ell+1}$, the objective function $\tilde{\mathcal{L}}(\mathbf{f}, \mathbf{g}) := \mathcal{L}_{\mathsf{LoRA}}(\mathbf{f}_{1:\ell}, \mathbf{g}_{1:\ell}) + w\mathcal{L}_{\mathsf{LoRA}}(\mathbf{f}_{[L]}, \mathbf{g}_{[L]})$ with $w > 0$ is minimized if and only if*

$$\sum_{i=1}^{\ell} |g_i^*\rangle\langle f_i^*| = \sum_{i=1}^{\ell} \sigma_i |\psi_i\rangle\langle\phi_i| \quad \text{and} \quad \sum_{i=\ell+1}^{L} |g_i^*\rangle\langle f_i^*| = \sum_{i=\ell+1}^{L} \sigma_i |\psi_i\rangle\langle\phi_i|.$$

Proof. First, note that by the Schmidt theorem (Theorem C.2),

$$\mathcal{L}_{\mathsf{LoRA}}(\mathbf{f}_{1:\ell}, \mathbf{g}_{1:\ell}) \geq \mathcal{L}_{\mathsf{LoRA}}(\mathbf{f}_{1:\ell}^*, \mathbf{g}_{1:\ell}^*) = \sum_{i=1}^{\ell} \sigma_\ell^2,$$

where the equality holds if and only if

$$\sum_{i=1}^{\ell} |g_i^*\rangle\langle f_i^*| = \sum_{i=1}^{\ell} \sigma_i |\psi_i\rangle\langle\phi_i|.$$

Using this property, we immediately have a lower bound

$$\mathcal{L}_{\mathsf{LoRA}}(\mathbf{f}_{1:\ell}, \mathbf{g}_{1:\ell}) + w\mathcal{L}_{\mathsf{LoRA}}(\mathbf{f}_{1:L}, \mathbf{g}_{1:L}) \geq \sum_{i=1}^{\ell} \sigma_i^2 + w \sum_{j=1}^{L} \sigma_j^2,$$

where the equality holds if and only if

$$\sum_{i=1}^{\ell} |g_i^*\rangle\langle f_i^*| = \sum_{i=1}^{\ell} \sigma_i |\psi_i\rangle\langle\phi_i| \quad \text{and} \quad \sum_{i=1}^{L} |g_i^*\rangle\langle f_i^*| = \sum_{i=1}^{L} \sigma_i |\psi_i\rangle\langle\phi_i|,$$

which is equivalent to

$$\sum_{i=1}^{\ell} |g_i^*\rangle\langle f_i^*| = \sum_{i=1}^{\ell} \sigma_i |\psi_i\rangle\langle\phi_i| \quad \text{and} \quad \sum_{i=\ell+1}^{L} |g_i^*\rangle\langle f_i^*| = \sum_{i=\ell+1}^{L} \sigma_i |\psi_i\rangle\langle\phi_i|. \qquad \square$$

We are now ready to prove Theorem 3.3.

Proof of Theorem 3.3. By inductively applying Lemma C.3 to the grouping $\mathcal{L}_{\mathsf{jnt}}(\mathbf{f},\mathbf{g}) = \sum_{i=1}^{\ell} w_i \mathcal{L}_{\mathsf{LoRA}}(\mathbf{f}_{[i]},\mathbf{g}_{[i]}) + \sum_{i=\ell+1}^{L} w_i \mathcal{L}_{\mathsf{LoRA}}(\mathbf{f}_{[i]},\mathbf{g}_{[i]})$ for $\ell = 1,\dots,L-1$, a minimizer must satisfy

$$\sum_{i=1}^{\ell} |g_i^*\rangle\langle f_i^*| = \sum_{i=1}^{\ell} \sigma_i |\psi_i\rangle\langle\phi_i|,$$

for each $\ell = 1,\dots,L$. This implies that the equivalence $|g_i^*\rangle\langle f_i^*| = \sigma_i|\psi_i\rangle\langle\phi_i|$ should hold term by term. $\square$

C.5. One-Shot Computation of Jointly Nested Objective

The gradient of the joint nesting objective (8) can be computed based on the following observation:

Proposition C.4 (One-shot computation)**.** *Given a positive weight vector* $\mathbf{w}$*, define* $\mathbf{m} \in \mathbb{R}^L$ *and* $\mathsf{M} \in \mathbb{R}^{L\times L}$ *as* $\mathsf{m}_i := \sum_{\ell=i}^{L} w_\ell$ *and* $\mathsf{M}_{ij} := \mathsf{m}_{\max\{i,j\}}$*. Then, the nested objective is written as*

$$\mathcal{L}_{\mathsf{jnt}}(\mathbf{f},\mathbf{g};\mathbf{w}) := -2\sum_{\ell=1}^{L} \mathsf{m}_\ell \langle g_\ell|\mathcal{T} f_\ell\rangle + \sum_{\ell=1}^{L}\sum_{\ell'=1}^{L} \mathsf{M}_{\ell\ell'}\langle f_\ell|f_{\ell'}\rangle\langle g_\ell|g_{\ell'}\rangle.$$

Proof. Recall that

$$\begin{aligned}\mathcal{L}_{\mathsf{jnt}}(\mathbf{f},\mathbf{g};\mathbf{w}) &= \sum_{\ell=1}^{L} w_\ell \mathcal{L}_{\mathsf{LoRA}}(\mathbf{f}_{1:\ell},\mathbf{g}_{1:\ell}) \\ &= \sum_{\ell=1}^{L} w_\ell \Big\{ -2\sum_{i=1}^{\ell}\langle g_i|\mathcal{T} f_i\rangle + \sum_{i=1}^{\ell}\sum_{j=1}^{\ell}\langle f_i|f_j\rangle\langle g_i|g_j\rangle \Big\}\end{aligned}$$

For the first term, we can write

$$\sum_{\ell=1}^{L} w_\ell \sum_{i=1}^{\ell}\langle g_i|\mathcal{T} f_i\rangle = \sum_{\ell=1}^{L} w_\ell \sum_{i=1}^{\ell} f_i(x) g_i(y) = \sum_{\ell=1}^{L} \mathsf{m}_\ell \langle g_\ell|\mathcal{T} f_\ell\rangle,$$

where $\mathsf{m}_\ell := \sum_{i=\ell}^{L} w_i$. For the second term, we can write

$$\sum_{\ell=1}^{L} w_\ell \sum_{1\le i,j\le \ell} \langle f_i|f_j\rangle\langle g_i|g_j\rangle = \sum_{1\le i,j\le L} \mathsf{M}_{ij}\langle f_i|f_j\rangle\langle g_i|g_j\rangle,$$

where $\mathsf{M}_{ij} := \mathsf{m}_{\max\{i,j\}} = \sum_{\ell=\max\{i,j\}}^{L} w_\ell$. This concludes the proof. $\square$

C.6. EVD with Non-Compact Operators

For a self-adjoint operator $\mathcal{T}$, we can apply our framework by considering the induced LoRA objective

$$\mathcal{L}_{\mathsf{LoRA}}(\mathbf{f}_{1:L}) := -2\sum_{\ell=1}^{L}\langle f_\ell|\mathcal{T}f_\ell\rangle + \sum_{\ell=1}^{L}\sum_{\ell'=1}^{L}\langle f_\ell|f_{\ell'}\rangle^2.$$

Though the original LoRA theorem of Schmidt (Theorem C.2) holds for a compact operator, it can be extended to a certain class of non-compact operators, which have discrete eigenvalues.

Theorem C.5. *For a self-adjoint operator $\mathcal{T}$, define*

$$\mathbf{f}^\star := \operatorname*{arg\,min}_{f_\ell\in\mathcal{F},\ \ell\in[L]} \mathcal{L}_{\mathsf{LoRA}}(\mathbf{f}).$$

Suppose that the operator $\mathcal{T}$ has r positive eigenvalues $\lambda_1 \geq \ldots \geq \lambda_r > 0 \geq \lambda_{r+1} \geq \ldots$ with corresponding orthonormal eigenfunctions $\{\phi_\ell\}_{\ell\geq 1}$, for some $r \in \mathbb{N}\cup\{\infty\}$. If $r < \infty$, the span of $|\mathbf{f}^\star\rangle$ is equal to the span of the top-$\min\{L,r\}$ eigenfunctions of the operator $\mathcal{T}$, or more precisely

$$\sum_{\ell=1}^{L}|f_\ell^\star\rangle\langle f_\ell^\star| = \sum_{\ell=1}^{\min\{L,r\}}\lambda_\ell|\phi_\ell\rangle\langle\phi_\ell|.$$

If $r=\infty$, i.e., when there are countably infinitely many positive eigenvalues, the same holds if $\lambda_L > \lambda_{L+1}$.

As a consequence of this theorem, when we optimize the $|\mathbf{f}_{1:L}\rangle$ with nesting for $L > r$, one can easily show that the optimal $|f_{r+1}^\star\rangle, \ldots, |f_L^\star\rangle$ are zero functions; we omit the proof.

Proof. We first consider when r is finite. Define the positive part of the operator as

$$\mathcal{T}_+ := \sum_{\ell=1}^{r}\lambda_\ell|\phi_\ell\rangle\langle\phi_\ell|,$$

which is compact by definition. Then, $\mathcal{T}_+ - \mathcal{T}$ is PSD with eigenvalues $0 \leq -\lambda_{r+1} \leq -\lambda_{r+2} \leq \ldots$ and eigenfunctions $\{\phi_\ell\}_{\ell\geq r+1}$. Then, we can rewrite and lower bound the LoRA objective as

$$\begin{aligned}\mathcal{L}_{\mathsf{LoRA}}(\mathbf{f}_{1:L}) &= \Big\|\mathcal{T}_+ - \sum_{\ell=1}^{L}|f_\ell\rangle\langle f_\ell|\Big\|_{\mathsf{HS}}^2 - \|\mathcal{T}_+\|_{\mathsf{HS}}^2 + 2\sum_{\ell=1}^{L}\langle f_\ell|(\mathcal{T}_+ - \mathcal{T})f_\ell\rangle \\ &\overset{(a)}{\geq} \Big\|\mathcal{T}_+ - \sum_{\ell=1}^{L}|f_\ell\rangle\langle f_\ell|\Big\|_{\mathsf{HS}}^2 - \|\mathcal{T}_+\|_{\mathsf{HS}}^2,\end{aligned}$$

where the inequality (a) follows since $\mathcal{T}_+ - \mathcal{T}$ is PSD. We note that the lower bound is minimized if and only if the span of $|\mathbf{f}^\star\rangle$ is equal to the span of the top-$\min\{L,r\}$ eigenfunctions of the operator $\mathcal{T}_+$ by applying Schmidt's theorem (Theorem C.2). We further note that (a) holds with equality, as $|f_1^\star\rangle, \ldots, |f_L^\star\rangle$ belong to the null space of $\mathcal{T}_+ - \mathcal{T}$. Hence, this concludes that the LoRA objective is minimized if and only if the span of $|\mathbf{f}^\star\rangle$ is equal to the span of the top-$\min\{L,r\}$ eigenfunctions of the operator $\mathcal{T}$.

When $r=\infty$, given that $\lambda_L > \lambda_{L+1}$, the rank-$L$ approximation of $\mathcal{T}$

$$\mathcal{T}_L := \sum_{\ell=1}^{L}\lambda_\ell|\phi_\ell\rangle\langle\phi_\ell|$$

is well-defined. The same proof for $r<\infty$ is valid if we replace $\mathcal{T}_+$ with $\mathcal{T}_L$. □

D. Implementation Details with Code Snippets

In this section, we explain how to implement the proposed NestedLoRA updates, providing readily deployable code snippets written in PyTorch. These code snippets are simplified from the actual implementation which can be found online[6] for the ease of exposition.

D.1. Helper Functions: Computing Nesting Masks and Metric Loss

As noted in Sec. 3.2.2, both versions of NestedLoRA can be implemented in a unified way via the nesting masks $(\mathsf{m}_\ell)_{\ell\in[L]}$ and $(\mathsf{M}_{i\ell})_{i\in[L],\ell\in[L]}$. Recall that for joint nesting, given positive weights $(w_1,\dots,w_L)$, we define $\mathsf{m}_\ell := \sum_{i=\ell}^{L} w_i$ and $\mathsf{M}_{i\ell} := \mathsf{m}_{\max\{i,\ell\}}$.

```
def get_joint_nesting_masks(weights: np.ndarray, set_first_mode_const: bool = False):
    vector_mask = list(np.cumsum(list(weights)[::-1])[::-1])
    if set_first_mode_const:
        vector_mask = [vector_mask[0]] + vector_mask
    vector_mask = torch.tensor(np.array(vector_mask)).float()
    matrix_mask = torch.minimum(vector_mask.unsqueeze(1), vector_mask.unsqueeze(1).T).float()
    return vector_mask, matrix_mask
```

Here, when the argument `set_first_mode_const` is set to be `True`, it outputs masks for CDK, for which we explicitly add the constant first mode; see Sec. D.4.

The sequential nesting (4) can be implemented by defining $\mathsf{m}_\ell := 1$ and $\mathsf{M}_{i\ell} := \mathbb{1}\{i \le \ell\}$.

```
def get_sequential_nesting_masks(L, set_first_mode_const: bool = False):
    if set_first_mode_const:
        L += 1
    vector_mask = torch.ones(L)
    matrix_mask = torch.triu(torch.ones(L, L))
    return vector_mask, matrix_mask
```

In the LoRA objective (3), the second term (with nesting), which we call the *metric loss*,

$$\sum_{\ell=1}^{L}\sum_{\ell'=1}^{L} \mathsf{M}_{\ell\ell'}\langle f_\ell|f_{\ell'}\rangle\langle g_\ell|g_{\ell'}\rangle = \sum_{\ell=1}^{L}\sum_{\ell'=1}^{L} (\mathsf{M}\odot\Lambda_{\mathbf{f}}\odot\Lambda_{\mathbf{g}})_{\ell\ell'} \tag{18}$$

is independent of the operator, where we define $(\Lambda_{\mathbf{f}})_{\ell\ell'} := \langle f_\ell|f_{\ell'}\rangle$ for $\ell,\ell'\in[L]$ and $\odot$ denotes the elementwise matrix product. Given samples $x_1,\dots,x_B$, we can estimate each entry of the matrix $\Lambda_{\mathbf{f}}\in\mathbb{R}^{L\times L}$ by

$$(\hat{\Lambda}_{\mathbf{f}})_{\ell\ell'} := \frac{1}{B}\sum_{b=1}^{B} f_\ell(x)f_{\ell'}(x),$$

which can be computed with PyTorch as:

```
def compute_lambda(f):
    return torch.einsum('bl,bm->lm', f, f) / f.shape[0]  # (L, L)
```

Then, the metric loss can be computed as follows:

```
def compute_loss_metric(f, g, matrix_mask):
    lam_f = compute_lambda(f)
    lam_g = compute_lambda(g)
    # compute loss_metric = E_{p(x)p(y)}[(f^T(x) g(y))^2]
    # f: (B1, L)
    # g: (B2, L)
    # lam_f, lam_g: (L, L)
    return (matrix_mask * lam_f * lam_g).sum(), lam_f, lam_g  # O(L ** 2)
```

Note that this metric loss needs not be computed when computing gradients.

[6] https://github.com/jongharyu/neural-svd

D.2. NestedLoRA Gradient Computation for Analytical Operators

In this section, we explain how to implement the NestedLoRA gradient updates for analytical operators. For the sake of simplicity, we explain for the implementation for EVD; the implementation for SVD can be found in our official PyTorch implementation.

For EVD of a self-adjoint operator $\mathcal{T}$, identifying $\mathbf{g}$ with $\mathbf{f}$, we need to compute the gradient

$$(\partial_{f_\ell}\mathcal{L})(x_b) = 2\Big\{-\mathsf{m}_\ell(\mathcal{T}f_\ell)(x_b) + \sum_{i=1}^{L}\mathsf{M}_{i\ell}f_i(x_b)\langle f_i\,|\,f_\ell\rangle\Big\} \tag{19}$$

for each $\ell \in [L]$; see (8) for the general expression. We can compute the gradient in an unbiased manner by plugging in the unbiased estimate of $\Lambda_{\mathbf{f}}$ based on $\{x'_1,\dots,x'_{B'}\}$. We remark that the minibatch samples for estimating $\Lambda_{\mathbf{f}}$ needs to be independent to $\{x_1,\dots,x_B\}$ so that the overall gradient estimate for $\langle\partial_\theta f_\ell|\,\partial_{f_\ell}\mathcal{L}\rangle$ becomes unbiased. This can be efficiently implemented in a vectorized manner by writing a custom backward function with the automatic differentiation package of PyTorch as follows. In what follows, we assume that $\{\mathbf{f}(x_b)\}_{b=1}^B$ and $\{(\mathcal{T}\mathbf{f})(x_b)\}_{b=1}^B$ are already computed for a given $\mathbf{f}$ and provided as `f` and `Tf`, resepectively. Further, `f1` and `f2` must be independent to each other.

```python
class NestedLoRALossFunctionEVD(torch.autograd.Function):
    @staticmethod
    @torch.cuda.amp.custom_fwd
    def forward(
            ctx: torch.autograd.function.FunctionCtx,
            f,
            Tf,
            f1,
            f2,
            vector_mask,
            matrix_mask,
    ):
        """
        the reduction assumed here is `mean` (i.e., we take average over batch)
            f: (B, L) or (B, L, O)
            Tf: (B, L) or (B, L, O)
            f1: (B1, L) or (B1, L, O)
            f2: (B2, L) or (B2, L, O)
        warning: f1 and f2 must be independent
        """
        ctx.vector_mask = vector_mask = vector_mask.to(f.device)
        ctx.matrix_mask = matrix_mask = matrix_mask.to(f.device)
        loss_metric, lam_f1, lam_f2 = compute_loss_metric(f1, f2, matrix_mask)
        ctx.save_for_backward(f, Tf, f1, f2, lam_f1, lam_f2)
        # compute loss_operator = -2 * E_{p(x)}[\sum_{l=1}^L f_l^T(x) (Tf_l)(x)]
        loss_operator = - 2 * torch.einsum('l,bl,bl->b', vector_mask, f, Tf).mean()  # O(B1 * L * O)
        loss = loss_operator + loss_metric
        return loss

    @staticmethod
    @torch.cuda.amp.custom_bwd
    def backward(
            ctx: torch.autograd.function.FunctionCtx,
            grad_output: torch.Tensor
    ) -> Tuple[torch.Tensor, ...]:
        """
        Args:
            ctx: The context object to retrieve saved tensors
            grad_output: The gradient of the loss with respect to the output
        """
        f, Tf, f1, f2, lam_f1, lam_f2 = ctx.saved_tensors
        operator_f = - (4 / f.shape[0]) * torch.einsum('l,bl->bl', ctx.vector_mask, Tf)
        metric_f1 = (2 / f1.shape[0]) * torch.einsum('lm,lm,bl->bm', ctx.matrix_mask, lam_f2, f1)
        metric_f2 = (2 / f2.shape[0]) * torch.einsum('lm,lm,bl->bm', ctx.matrix_mask, lam_f1, f2)
        return grad_output * operator_f, None, grad_output * metric_f1, grad_output * metric_f2, \
            None, None, None
```

In practice, when given a minibatch $\{x_1, \ldots, x_B\}$, we can use the entire batch to compute `f` and `Tf`, and split `f` into two equal parts and plug in them to `f1` and `f2` to ensure the independence. In what follows, the operator is given as an abstract function `operator`, whose interface is explained in the next section (Sec. D.3).

```
def compute_loss_operator(
        model,
        operator,
        x,
        importance=None,
):
    Tf, f = operator(model, x, importance=importance)
    f1, f2 = torch.chunk(f, 2)
    loss = NestedLoRALossFunctionEVD.apply(
        f, Tf, f1, f2,
        vector_mask,
        matrix_mask,
    )
    return loss, dict(f=f, Tf=Tf, eigvals=None)
```

After this function returns `loss`, calling `loss.backward()` will backpropagate the gradients based on the custom backward function, and populate the gradient for each model parameter.

D.3. Importance Sampling

Unlike machine learning applications where the sampling distribution is given by data, the underlying measure $\mu(x)$ is the Lebesgue measure over a given domain when solving PDEs. Note that, when the domain is not bounded, we cannot sample from the measure, and thus it is necessary to introduce a sampling distribution to apply our framework. Given a distribution $p_{\mathsf{tr}}(x)$ that is supported over the support of $\mu(x)$, the inner product between $|f\rangle$ and $|\mathcal{T}f\rangle$ can be written as

$$\begin{aligned}\langle f|\mathcal{T}f\rangle &= \int f(x)\mathcal{T}f(x)\mu(x)\,\mathrm{d}x \\ &= \int f(x)\sqrt{\frac{\mu(x)}{p_{\mathsf{tr}}(x)}}\mathcal{T}f(x)\sqrt{\frac{\mu(x)}{p_{\mathsf{tr}}(x)}}p_{\mathsf{tr}}(x)\,\mathrm{d}x \\ &= \int \frac{f(x)}{\sqrt{w_{\mathsf{tr}}(x)}}\frac{\mathcal{T}f(x)}{\sqrt{w_{\mathsf{tr}}(x)}}p_{\mathsf{tr}}(x)\,\mathrm{d}x.\end{aligned}$$

Here, we define the (training) importance function $w_{\mathsf{tr}}(x) := \frac{p_{\mathsf{tr}}(x)}{\mu(x)}$. For the case of the Lebesgue measure, one can simply regard $\mu(x)$ as 1. It is sometimes crucial to choose a good training sampling distribution, especially for high-dimensional problems.

Suppose now that we directly parameterize $\frac{f(x)}{\sqrt{w_{\mathsf{tr}}(x)}}$ by a neural network $\tilde{f}(x)$. Then, the inner product can be computed as

$$\langle f|\mathcal{T}f\rangle = \int \tilde{f}(x)\frac{\mathcal{T}f(x)}{\sqrt{w_{\mathsf{tr}}(x)}}p_{\mathsf{tr}}(x)\,\mathrm{d}x.$$

Here, $\mathcal{T}f(x)$ can be computed by applying the operator $\mathcal{T}$ to the function $x \mapsto \sqrt{w_{\mathsf{tr}}(x)}\tilde{f}(x)$.

During the test phase, we may use another test distribution $p_{\mathsf{te}}(x)$ to evaluate the inner product. Given another valid sampling distribution $p_{\mathsf{te}}(x)$,

$$\langle f|\mathcal{T}f\rangle = \int \tilde{f}(x)\frac{\mathcal{T}f(x)}{\sqrt{w_{\mathsf{tr}}(x)}}\frac{p_{\mathsf{tr}}(x)}{p_{\mathsf{te}}(x)}p_{\mathsf{te}}(x)\,\mathrm{d}x = \int \tilde{f}(x)\frac{\mathcal{T}f(x)}{\sqrt{w_{\mathsf{tr}}(x)}}\frac{w_{\mathsf{tr}}(x)}{w_{\mathsf{te}}(x)}p_{\mathsf{te}}(x)\,\mathrm{d}x,$$

where we define the (test) importance function $w_{\mathsf{te}}(x) := \frac{p_{\mathsf{te}}(x)}{\mu(x)}$. Again, for high-dimensional problems, it is crucial to choose a good sampling distribution for reliable evaluation.

In our implementation, the operator is defined with the following interface: given a neural network model $\tilde{f}(x)$ and a training importance function $p_{\mathsf{te}}(x)$, `operator(model, x, importance)` outputs $\frac{\mathcal{T}f(x)}{\sqrt{w_{\mathsf{tr}}(x)}}$ and $\tilde{f}(x)$, so that they can be taken to the inner product directly under $p_{\mathsf{tr}}(x)$. Given this, the original function value $f(x)$ can be recovered as $f(x) = \sqrt{w_{\mathsf{tr}}(x)}\tilde{f}(x)$.

For example, we implement the negative Hamiltonian as follows:

```
class NegativeHamiltonian:
    def __init__(self,
                 local_potential_ftn,
                 scale_kinetic=1.,
                 laplacian_eps=1e-5,
                 n_particles=1):
        self.laplacian_eps = laplacian_eps
        self.laplacian = VectorizedLaplacian(eps=laplacian_eps)
        self.local_potential_ftn = local_potential_ftn
        self.scale_kinetic = scale_kinetic
        self.n_particles = n_particles

    def __call__(self, f, xs, importance=None, threshold=1e5):
        # threshold is to detect an anomaly in the hamiltonian
        lap, grad, fs = self.laplacian(f, xs, importance)
        kinetic = - self.scale_kinetic * lap
        potential = self.local_potential_ftn(xs.reshape((xs.shape[0], self.n_particles, -1))).view(-1, 1) *
     fs
        hamiltonian = kinetic + potential
        return - hamiltonian, fs
```

Here, `VectorizedLaplacian` refers to a function for vectorized Laplacian computation, whose implementation can be found in our code.

D.4. NestedLoRA Gradient Computation for CDK

Recall that the CDK is defined as $k(x,y) = \overline{k}(x,y) - 1$ with $\overline{k}(x,y) := \frac{p(x,y)}{p(x)p(y)}$. We note that it is known that $\overline{k}(x,y)$ has the constant functions as the singular functions with singular value 1, i.e., $(1, x \mapsto 1, y \mapsto 1)$ is the first singular triplet of $\overline{k}(x,y)$; see, e.g., (Huang et al., 2024). Hence, the term "-1" in the definition of CDK is to remove the first *trivial* mode of $\overline{k}(x,y)$.

In our implementation, we handle the decomposition of CDK by considering $\overline{k}(x,y)$ with explicitly augmenting the constant functions as the fictitious first singular functions, so that we effectively learn from the second singular functions of $\overline{k}(x,y)$ and on. For $\overline{k}(x,y) = \frac{p(x,y)}{p(x)p(y)}$, the "operator term" $\langle g_\ell | \overline{\mathcal{K}} f_\ell \rangle$ can be computed as, again by change of measure,

$$\langle g_\ell | \overline{\mathcal{K}} f_\ell \rangle = \mathbb{E}_{p(x,y)}[f_\ell(X) g_\ell(Y)].$$

Hence, compared to (19), the gradient becomes, for each $\ell = 1, \dots, L$,

$$(\partial_{f_\ell} \mathcal{L})(x_b) = 2\Big\{ -\mathsf{m}_\ell g_\ell(x_b) + \sum_{i=0}^{L} \mathsf{M}_{i\ell} f_i(x_b) \langle g_i | g_\ell \rangle \Big\} \tag{20}$$

where we set $f_0(x) \equiv 1$ and $g_0(y) \equiv 1$; $(\partial_{g_\ell} \mathcal{L})(x_b)$ is similarly computed. The following snippet implements this gradient using a custom gradient as before. Note that the constant 1's are explicitly appended as the first mode in line 16-18.

```
class NestedLoRALossFunctionForCDK(torch.autograd.Function):
    @staticmethod
    @torch.cuda.amp.custom_fwd
    def forward(
            ctx: torch.autograd.function.FunctionCtx,
            f,
            g,
            vector_mask,
            matrix_mask,
    ):
        """
        the reduction assumed here is `mean` (i.e., we take average over batch)
            f: (B, L)
            g: (B, L)
        """
```

```
        pad = nn.ConstantPad1d((1, 0), 1)
        f = pad(f)
        g = pad(g)
        ctx.vector_mask = vector_mask = vector_mask.to(f.device)
        ctx.matrix_mask = matrix_mask = matrix_mask.to(f.device)
        loss_metric, lam_f, lam_g = compute_loss_metric(f, g, matrix_mask)
        ctx.save_for_backward(f, g, lam_f, lam_g)
        # compute loss_operator = -2 * E_{p(x,y)}[f^T(x) g(y)]
        loss_operator = - 2 * torch.einsum('l,bl,bl->b', vector_mask, f, g).mean()  # O(B1 * L)
        loss = loss_operator + loss_metric
        gram_matrix = f @ g.T  # (B, B); each entry is (f^T(x_i) g(y_j))
        rs_joint = gram_matrix.diag()
        rs_indep = off_diagonal(gram_matrix)
        return loss, loss_operator, loss_metric, rs_joint, rs_indep

    @staticmethod
    @torch.cuda.amp.custom_bwd
    def backward(
            ctx: torch.autograd.function.FunctionCtx,
            grad_output: torch.Tensor,
            *args
    ) -> Tuple[torch.Tensor, ...]:
        """
        Args:
            ctx: The context object to retrieve saved tensors
            grad_output: The gradient of the loss with respect to the output
        """
        f, g, lam_f, lam_g = ctx.saved_tensors
        # for grad(f)
        operator_f = - (2 / f.shape[0]) * torch.einsum('l,bl->bl', ctx.vector_mask, g)
        metric_f = (2 / f.shape[0]) * torch.einsum('il,il,bi->bl', ctx.matrix_mask, lam_g, f)
        grad_f = operator_f + metric_f
        # for grad(g)
        operator_g = - (2 / g.shape[0]) * torch.einsum('l,bl->bl', ctx.vector_mask, f)
        metric_g = (2 / g.shape[0]) * torch.einsum('il,il,bi->bl', ctx.matrix_mask, lam_f, g)
        grad_g = operator_g + metric_g
        grad_f = grad_f[:, 1:]
        grad_g = grad_g[:, 1:]
        return grad_output * grad_f, grad_output * grad_g, None, None, None, None
```

In practice, given minibatch samples $\{(x_b, y_b)\}_{b=1}^B$ drawn from a joint distribution $p(x, y)$, we can compute $\{(\mathbf{f}(x_b), \mathbf{g}(y_b))\}_{b=1}^B$ and plug in to the function above as follows.

```
def compute_loss(
        f,
        g,
) -> torch.Tensor:
    return NestedLoRALossFunctionForCDK.apply(
        f,
        g,
        vector_mask,
        matrix_mask,
    )
```

Here, `vector_mask` and `matrix_mask` should be computed using the mask computing functions in Sec. D.1 with `set_first_mode_const=True`.

D.5. Spectrum Estimation via Norm Estimation

As alluded to in the main text, we can estimate the singular values $(\sigma_1, \ldots, \sigma_L)$ from the learned functions and training data, i.e.,

$$\sigma_\ell = \sqrt{\mathbb{E}_{p(x)}[(f_\ell^\star(X))^2]\mathbb{E}_{p(y)}[(g_\ell^\star(Y))^2]} \quad \text{for } \ell = 1, \ldots, L.$$

Here, by replacing the expectation with the empirical expectation and the optimal $f_\ell^\star, g_\ell^\star$ with the learned ones $\hat{f}_\ell, \hat{g}_\ell$, we obtain the singular value estimator:

$$\hat{\sigma}_\ell := \sqrt{\mathbb{E}_{\hat{p}(x)}[(\hat{f}_\ell(X))^2]\mathbb{E}_{\hat{p}(y)}[(\hat{g}_\ell(Y))^2]} \quad \text{for } \ell = 1, \dots, L.$$

```
def singular_values(f_t, g_t):  # f_t: (M, L); g_t: (N, L)
    return ((f_t ** 2).mean(dim=0) * (g_t ** 2).mean(dim=0)).sqrt()  # (L, )
```

For symmetric, PD kernels and operators, the eigenvalue estimator becomes:

$$\hat{\lambda}_\ell := \mathbb{E}_{\hat{p}(x)}[(\hat{f}_\ell(X))^2] \quad \text{for } \ell = 1, \dots, L.$$

D.6. Sequential Nesting for Shared parameterization

As alluded to earlier in footnote 2, we can still apply sequential nesting even when the functions $\{(f_\ell, g_\ell)\}_{\ell=1}^L$ are parameterized by a shared model with a collective parameter θ. The idea is to consider a *masked* gradient $\tilde{\partial}_\theta(\mathcal{L}_{\mathsf{LoRA}})_\ell$, which is a masked version of the original gradient $\partial_\theta(\mathcal{L}_{\mathsf{LoRA}})_\ell$ computed with the assumption that $|\,\partial_{f_{\ell'}}(\mathcal{L}_{\mathsf{LoRA}})_\ell\rangle = 0$ and $|\,\partial_{g_{\ell'}}(\mathcal{L}_{\mathsf{LoRA}})_\ell\rangle = 0$ for every $1 \le \ell' < \ell$, for each ℓ. The resulting masked gradient can be explicitly written as

$$\tilde{\partial}_\theta(\mathcal{L}_{\mathsf{LoRA}})_\ell = \sum_{\ell=1}^{L}\{\langle \partial_\theta f_\ell|\,\partial_{f_\ell}(\mathcal{L}_{\mathsf{LoRA}})_\ell\rangle + \langle \partial_\theta g_\ell|\,\partial_{g_\ell}(\mathcal{L}_{\mathsf{LoRA}})_\ell\rangle\}.$$

E. Experiment Details

In this section, we provide all the details for our experiments. All experiments were run on a single GPU (NVIDIA GeForce RTX 3090). Codes and scripts to replicate the experiments have been open-sourced online.[7]

E.1. Solving Time-Independent Schrödinger Equations

E.1.1. 2D Hydrogen Atom

Analytical Solution. For the 2D-confined hydrogen-like atom, the Hamiltonian is given as $\mathcal{H} = \mathcal{T} + \mathcal{V} = -\frac{\hbar^2}{2m}\nabla^2 - \frac{Ze^2}{\|\mathbf{x}\|_2}$, where Z is the charge of the nucleus. Yang et al. (1991) provides a closed-form expression of the eigenfunctions for this special case. Here, we present the formula with slight modifications for visualization purposes.

By normalizing constants (i.e., $Ze^2 \leftarrow 1$, $\frac{2m}{\hbar^2} \leftarrow 1$), we can simplify it to the eigenvalue problem $(\nabla^2 + \frac{1}{\|\mathbf{x}\|_2})\psi(\mathbf{x}) = \lambda\psi(\mathbf{x})$ for $\mathbf{x} \in \mathbb{R}^2$. Each eigenstate is parameterized by a pair of integers (n, l) for $n \ge 0$ and $-n \le l \le n$, where the (negative) eigenenergy is $\lambda_{n,l} := (2n+1)^{-2}$. Note that for each $n \ge 0$, there exist $2n+1$ degenerate states that have the same energy. Further, the operator is PD and compact, since $\lambda_{n,l} > 0$ and the Hilbert–Schmidt norm of the negative Hamiltonian is finite, i.e., $\sum_{n\ge0}\sum_{l=-n}^{n}\lambda_{n,l}^2 = \sum_{n\ge0}\frac{1}{(2n+1)^3} < \infty$.

The eigenfunctions can be explicitly expressed in the spherical coordinate system

$$\psi_{n,l}(\mathbf{x}) = \psi_{n,l}(r,\theta) := \psi_{n,l}(r)\psi_l(\theta), \tag{21}$$

where the radial part is

$$\psi_{n,l}(r) = \frac{\beta_n}{(2|l|)!}\left(\frac{(n+|l|)!}{(2n+1)(n-|l|)!}\right)^{\frac{1}{2}}(\beta_n r)^{|l|}e^{-\frac{\beta_n r}{2}}\,{}_1F_1(-n+|l|, 2|l|+1, \beta_n r)$$

with $\beta_n = (n+\frac{1}{2})^{-1}$, and the angular part is

$$\psi_l(\theta) = \begin{cases} \frac{1}{\sqrt{\pi}}\cos(l\theta) & \text{if } l > 0 \\ \frac{1}{\sqrt{2\pi}} & \text{if } l = 0 \\ \frac{1}{\sqrt{\pi}}\sin(l\theta) & \text{if } l < 0. \end{cases}$$

Here, ${}_1F_1(a; b; x)$ denotes the confluent hypergeometric function.

[7] https://github.com/jongharyu/neural-svd

Implementation Details. We adopted the training setup of (Pfau et al., 2019) with some variations.

- **Differential operator.** To reduce the overall complexity of the optimization, we approximated the Laplacian by the standard finite difference approximation: for $\epsilon > 0$ sufficiently small,

$$\nabla^2 f(\mathbf{x}) \approx \frac{1}{\epsilon^2} \sum_{i=1}^{D} (f(\mathbf{x} + \epsilon \mathbf{e}_i) + f(\mathbf{x} - \epsilon \mathbf{e}_i) - 2f(\mathbf{x})).$$

  In this paper, we used $\epsilon = 0.01$ throughout.

- **Sampling distribution.** We chose a sampling distribution $p_{\mathsf{tr}}(x)$ as a Gaussian distribution $\mathcal{N}(0, 16^2 \mathsf{I}_2)$; see Sec. D.3.

- **Architecture.** We used 16 disjoint three-layer MLPs with 128 hidden units to learn the first $L = 16$ eigenfunctions, except $L = 9$ for SpIN that did not fit to a single GPU due to the large memory requirement; see Sec. B.2.1. For the nonlinear activation function, we used the softplus activation $f(x) = \log(1 + e^x)$ following the implementation of (Pfau et al., 2019). We also found that multi-scale Fourier features (Wu et al., 2023) are effective, especially the non-differentiable points at the origin for some eigenstates of the 2D hydrogen atom. The multi-scale Fourier feature is defined as follows. Let $D = 2$ denote the input dimension. For $K \in \mathbb{N}$ and $\kappa > 0$, we initialize and fix a Gaussian random matrix $\mathsf{B} \in \mathbb{R}^{K \times D}$, each of which entry is drawn from $\mathcal{N}(0, 2\pi\kappa)$. An input is projected by B to the K dimensional space, and mapped into Fourier features $(\cos(\mathsf{B}\mathbf{x}), \sin(\mathsf{B}\mathbf{x})) \in \mathbb{R}^{2K}$, following (Tancik et al., 2020). In our experiments, we also appended the raw input $\mathbf{x}$ to the Fourier feature, so that the feature dimension becomes $2K + D$. We used $K = 1024$ for NeuralEF and NeuralSVD, and $K = 512$ for SpIN. Lastly, $\kappa = 0.1$ was used.

- **Optimization.** We trained the networks for 5×10^5 iterations with batch size 128 and 512. For all methods, we used the RMSProp optimizer (Hinton et al., 2012) with learning rate 10^{-4} and the cosine learning rate schedule (Loshchilov & Hutter, 2016).

- **Evaluation.** During the evaluation, we applied the exponential moving average (over the model parameters) with a decay rate of 0.995 for smoother results. We also used a uniform distribution over $[-100, 100]^2$ as a sampling distribution, assuming that the eigenfunctions vanish outside the box, which is approximately true. Sec. D.3 for the detailed procedure for the importance sampling during evaluation.

E.1.2. 2D Harmonic Oscillator

Analytical Solution. Define

$$\psi_n(x) = \frac{1}{\sqrt{2^n\, n!}} \left(\frac{b}{\pi}\right)^{1/4} e^{-\frac{bx^2}{2}} H_n(\sqrt{b}x), \qquad n = 0, 1, 2, \ldots.$$

Here, $H_n(x)$ denotes the physicists' Hermite polynomials

$$H_n(z) = (-1)^n\, e^{z^2} \frac{d^n}{dz^n}(e^{-z^2}),$$

and we simplify the constant $b = \frac{m\omega}{\hbar}$ to 1. Then $\{\psi_n(x)\}_{n \geq 0}$ characterizes the eigenbasis of 1D harmonic oscillator.

Each eigenstate of the 2D harmonic oscillator is characterized by a pair of nonnegative integers (n_x, n_y), and $\lambda_{n_x, n_y} = 2(n_x + n_y + 1)$, where a canonical representation of the eigenfunction is

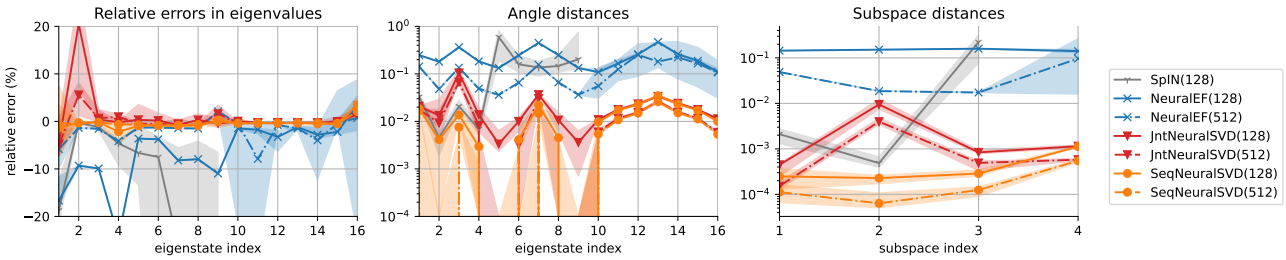
$$\psi_{n_x, n_y}(x, y) := \psi_{n_x}(x)\psi_{n_y}(y).$$

Note that for each $n \geq 0$, there exist $n + 1$ eigenstates that share the same eigenvalue $2(n + 1)$.

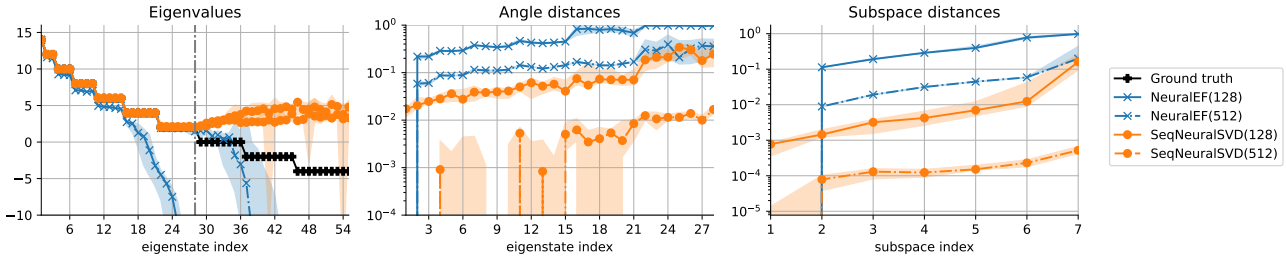
Implementation Details. We used an almost identical setup to the 2D hydrogen atom experiment except the followings.

- **Operator shifting.** We chose to decompose $\mathcal{T} + cI$ for $c = 16$, so that the first 28 eigenstates have positive eigenvalues.

- **Sampling distribution.** We chose a sampling distribution $p_{\mathsf{tr}}(x)$ as a Gaussian distribution $\mathcal{N}(0, 4^2 \mathsf{I}_2)$.



(a) 2D hydrogen atom (16 eigenstates)



(b) 2D harmonic oscillator (only 28 positive eigenstates out of 55 states are to be learned)

Figure 7: Quantitative evaluations of the learned eigenfunctions. The shaded region indicates 20% and 80% quantiles with respect to 10 random seeds. In the first panel of (b), the left of the black vertical line indicates the positive eigenvalues.

- **Architecture.** We used the same disjoint parameterization as before, but with $K = 256$ and $\kappa = 1$.
- **Optimization.** We trained the networks for 10^5 iterations with batch size 128 and 512.
- **Evaluation.** We also used a uniform distribution over $[-5, 5]^2$ as a sampling distribution.

Remark E.1 (On shifting). We note that the shifting technique can be applied to similar non-compact operators in general, but the shifting parameter c needs to be tuned by trial and error as the underlying spectrum is unknown in practice. But how should one choose the parameter? Since any c beyond a certain threshold makes the first L modes with strictly positive eigenvalues for a fixed number of modes L, one may ask whether using larger c is always a safe choice. On one hand, if c is too large, the shifted operator $\mathcal{T} + cI$ is dominated by the identity operator, which admits *any* set of orthonormal functions as its orthonormal eigenbasis. On the other hand, c needs to be sufficiently large to ensure the L-th mode to be recovered. Therefore, in practice, c needs to be tuned as a hyperparameter considering such a trade-off.

E.1.3. Definitions of Reported Measures

We first provide the definitions of the reported measures in Fig. 4 and Fig. 7.

- **Relative errors in eigenvalues**: Given a learned eigenfunction $\tilde{\psi}_\ell(x)$, we estimate the learned eigenvalue by the Rayleigh quotient

$$\tilde{\lambda}_\ell := \frac{\langle \tilde{\psi}_\ell | \mathcal{T} \tilde{\psi}_\ell \rangle}{\langle \tilde{\psi}_\ell | \tilde{\psi}_\ell \rangle},$$

  where each inner product is computed by importance sampling with finite samples from a given sampling distribution; see Sec. D.3. For each ℓ, we then report the absolute relative error

$$\frac{(\tilde{\lambda}_\ell - \lambda_\ell)}{\lambda_\ell} \times 100\ (\%),$$

  for $\lambda_\ell > 0$.

- **Angle distances**: When there is degeneracy, i.e., several eigenstates share same eigenvalue, we need to *align* the learned functions within each subspace before we evaluate the performance eigenstate-wise. For such an alignment, we

use the orthogonal Procrustes (OP) procedure defined as follows. Suppose that $\mathsf{A} \in \mathbb{R}^{N\times K}$ and $\mathsf{B} \in \mathbb{R}^{N\times K}$ are given. We wish to find the find the orthogonal transformation $\mathsf{A}\Omega$ that best approximates the reference B. The OP procedure defines the optimal $\Omega^\star$ by the optimization problem

$$\begin{aligned} \underset{\Omega\in\mathbb{R}^{K\times K}}{\text{minimize}} \quad & \|\mathsf{A}\Omega - \mathsf{B}\|_F \\ \text{subject to} \quad & \Omega^\intercal\Omega = I. \end{aligned}$$

The solution is characterized by the SVD of $\mathsf{A}^\intercal\mathsf{B} \in \mathbb{R}^{K\times K}$. If $\mathsf{A}^\intercal\mathsf{B} = \mathsf{U}\mathsf{S}\mathsf{V}^\intercal$ is the SVD, then $\Omega^\star = \mathsf{U}\mathsf{V}^\intercal$. In our case, A is the vertical stack of the learned eigenfunctions and B is that of the ground truth eigenfunctions that correspond to a degenerate eigensubspace. Here, K is the number of degeneracy and N is the number of points used for the alignment. Given the aligned learned function $\bar{\psi}_\ell(x)$, we report the normalized angle distance

$$\angle(|\bar{\psi}_\ell\rangle, |\psi_\ell\rangle) := \frac{2}{\pi}\arccos|\langle\psi_\ell|\bar{\psi}_\ell\rangle| \in [0,1].$$

Here, we assume that both $|\bar{\psi}_\ell\rangle$ and $|\psi_\ell\rangle$ are normalized.

- **Subspace distances**: Another standard quantitative measure is the *subspace distance* defined as follows. Given $\mathsf{A} \in \mathbb{R}^{N\times K}$ and $\mathsf{B} \in \mathbb{R}^{N\times K}$, the normalized subspace distance between the column subspaces of the two matrices is defined as

$$d(\mathsf{A},\mathsf{B}) := 1 - \frac{1}{K}\operatorname{tr}(\mathsf{P}\mathsf{Q}),$$

where $\mathsf{P} = \mathsf{A}(\mathsf{A}^\intercal\mathsf{A})^{-1}\mathsf{A}^\intercal \in \mathbb{R}^{N\times N}$ and $\mathsf{Q} = \mathsf{B}(\mathsf{B}^\intercal\mathsf{B})^{-1}\mathsf{B}^\intercal \in \mathbb{R}^{N\times N}$ are the projection matrices onto the column subspaces of A and B, respectively. We note that A and B correspond to the learned and ground truth eigenfunctions that correspond to a given subspace as above.

The reported measures in Fig. 4 are *averaged* versions of the quantities defined above, except the orthogonality.

- **Relative errors in eigenvalues**: Report the average of the absolute relative errors over the eigenstates.
- **Angle distance**: Report the average of the angle distances over the eigenstates.
- **Subspace distance**: Report the average of the subspace distances over the degenerate subspaces.
- **Orthogonality**: To measure the orthogonality of the learned eigenfunctions, we report

$$\frac{1}{N^2}\sum_{\ell=1}^{L}\sum_{\ell'=1}^{L}(\langle\tilde{\psi}_\ell|\tilde{\psi}_{\ell'}\rangle - \delta_{\ell\ell'})^2.$$

E.2. Cross-Domain Retrieval with Canonical Dependence Kernel

We used the Sketchy Extended dataset (Sangkloy et al., 2016; Liu et al., 2017) to train and evaluate our framework. There are total 75,479 sketches (x) and 73,002 photos (y) from 125 different classes.

We followed the standard training setup in the literature (Hwang et al., 2020).

- **Sampling distribution.** As described in the main text, we define a sampling distribution as follows. First, note that we are given (empirical) class-conditional distributions $\{p(x|c)\}_{c=1}^K$ and $\{p(y|c)\}_{c=1}^K$ for each class $c \in [K]$. Given the (empirical) class distribution $p(c)$, we define the joint distribution

$$p(x,y) := \mathbb{E}_{p(c)}[p(x|C)p(y|C)].$$

That is, in practice, to draw a sample from $p(x,y)$, we can draw $C \sim p(c)$, and draw $(X,Y) \sim p(x|C)p(y|C)$.

- **Pretrained fetures.** We used a pretrained VGG16 network (Simonyan & Zisserman, 2015) to extract features of the sketches and images. The pretrained VGG network and train-test splits for evaluation are from the codebase[8] of (Dutta & Akata, 2019). Hence, each sketch and photo is represented by a 512-dim. feature from the VGG network.

[8] https://github.com/AnjanDutta/sem-pcyc

- **Architecture.** Treating the 512-dim. pretrained features as input, we used a single one-layer MLP of 8192 hidden units whose output dimension is 512. At the end of the network, we regularized the output so that the norm of the output has ℓ_2-norm less than equal to $\mu = 16$, i.e., $\|\mathbf{f}(x)\|_2 \leq \mu$ for every x.
- **Optimization.** We trained the network for 10 epochs with batch size of 4096. We used the SGD optimizer with learning rate 5×10^{-3} and momentum 0.9, together with the cosine learning rate schedule (Loshchilov & Hutter, 2016).

Evaluation Metrics. Precision@k and mean average precision are widely used metrics for evaluating a retrieval system such as search engines (Salton & McGill, 1986). When k items are retrieved for a query, Precision@k (P@k) is defined as the number of relevant items (i.e., the number of photos of the same class as a query sketch in our scenario) divided by k. Average precision (AP) is also defined for a certain query point. When there are n photos in the candidate pool, the AP is defined as $\sum_{k=1}^{n} P(k) \times (R(k) - R(k-1))$, where $P(k)$ denotes P@k and $R(k)$ denotes Recall@k, which is defined as the number of relevant items in the k retrievals divided by the total number of relevant items (i.e.,, number of "all" photos of the same class as a query sketch). Then, finally, the mean average precision (mAP) is defined as the average of all average precision over all possible queries.