

686 Appendix

687	A Deferred Proofs	17
688	A.1 Equivalence of $\mathcal{L}_{\text{orbital}}^{(p)}(V)$ and $\tilde{\mathcal{L}}_{\text{orbital}}^{(p)}(V)$	17
689	A.2 Proof of Theorem 1	18
690	A.3 Proof of Theorem 2	18
691	B On Implementation	18
692	B.1 Implementation of Sanger’s Variant	18
693	B.2 Pseudocode for Nested Orbital Minimization Loss	19
694	C Details on Experimental Setups and Additional Results	20
695	C.1 Laplacian Representation Learning for Reinforcement Learning	20
696	C.2 Solving Schrödinger Equation	20
697	C.3 Self-Supervised Contrastive Representation Learning	21
698	C.3.1 Representation Learning for Images	21
699	C.3.2 Representation Learning with Graph Data	22
700	D On the Benefit of Higher-Order OMM	23

701 List of Typos and Corrections

702 We have found the following typos that need to be corrected from the main text after the submission.
703 We will update the manuscript accordingly.

- 704 • In line 223, “. . . in the numerical linear algebra literature, . . .”.
- 705 • In line 338, “ $p \in \{1, 2\}$ ”.
- 706 • Below line 203, $\mathcal{L}_{\text{orbital}}^{\text{seq}}(\mathbf{f}, \mathbf{g})$ should be replaced with $\mathcal{L}_{\text{orbital}}^{\text{seq}}(\mathbf{f})$.
- 707 • Below line 214, $\mathcal{L}_{\text{orbital}}^{(p)}(\mathbf{f}; \mathbf{a})$ should be replaced with $\mathcal{L}_{\text{orbital}}^{(p)}(\mathbf{f}; \boldsymbol{\alpha})$.

708 A Deferred Proofs

709 A.1 Equivalence of $\mathcal{L}_{\text{orbital}}^{(p)}(V)$ and $\tilde{\mathcal{L}}_{\text{orbital}}^{(p)}(V)$

710 To establish the equivalence between the original proposal $\mathcal{L}_{\text{orbital}}^{(p)}(V)$ in Eq. (4) and our derivation
711 $\tilde{\mathcal{L}}_{\text{orbital}}^{(p)}(V)$ in Eq. (5), we need to show that

$$-\text{tr}(\mathbf{Q}_p \mathbf{V}^\top \mathbf{A} \mathbf{V}) = \text{tr}((\mathbf{I}_d - \mathbf{V} \mathbf{V}^\top)^{2p} \mathbf{A}) - \text{tr}(\mathbf{A}),$$

712 where we recall

$$\mathbf{Q}_p := \sum_{i=0}^{2p-1} (\mathbf{I}_k - \mathbf{V}^\top \mathbf{V})^i.$$

713 To show this, we first note that we can write

$$\mathbf{Q}_p = \sum_{i=0}^{2p-1} (\mathbf{I}_k - \mathbf{S})^i = \mathbf{S}^{-1} (\mathbf{I}_k - (\mathbf{I}_k - \mathbf{S})^{2p}),$$

714 by letting $\mathbf{S} := \mathbf{V}^\top \mathbf{V}$. Hence, we have

$$\begin{aligned} -\text{tr}(\mathbf{Q}_p \mathbf{V}^\top \mathbf{A} \mathbf{V}) &= -\text{tr}\left(\mathbf{S}^{-1} (\mathbf{I}_k - (\mathbf{I}_k - \mathbf{S})^{2p}) \mathbf{V}^\top \mathbf{A} \mathbf{V}\right) \\ &= -\text{tr}(\mathbf{S}^{-1} \mathbf{V}^\top \mathbf{A} \mathbf{V}) + \text{tr}\left(\mathbf{S}^{-1} (\mathbf{I}_k - \mathbf{S})^{2p} \mathbf{V}^\top \mathbf{A} \mathbf{V}\right) \\ &= -\text{tr}(\mathbf{S}^{-1} \mathbf{V}^\top \mathbf{A} \mathbf{V}) + \text{tr}(\mathbf{S}^{-1} \mathbf{V}^\top \mathbf{A} \mathbf{V}) + \text{tr}\left((\mathbf{I}_d - \mathbf{V} \mathbf{V}^\top)^{2p} \mathbf{A}\right) - \text{tr}(\mathbf{A}) \\ &= \text{tr}\left((\mathbf{I}_d - \mathbf{V} \mathbf{V}^\top)^{2p} \mathbf{A}\right) - \text{tr}(\mathbf{A}). \end{aligned}$$

715 This concludes the proof. □

716 A.2 Proof of Theorem 1

717 *Proof of Theorem 1.* Since VV^\top is of rank at most k and PSD, the optimization can be reparam-
 718 eterized based on the reduced SVD of $VV^\top = P\Sigma P^\top$, where $P := [\mathbf{p}_1, \dots, \mathbf{p}_k] \in \mathbb{R}^{d \times k}$ and
 719 $\Sigma := \text{diag}(\mu_1, \dots, \mu_k) \in \mathbb{R}^{k \times k}$, such that $P^\top P = I_k$ and $\mu_1 \geq \dots \geq \mu_k \geq 0$. For a given orthog-
 720 onal matrix $P \in \mathbb{R}^{d \times k}$, let $P_\perp \in \mathbb{R}^{d \times (d-k)}$ denote a matrix whose columns form an orthonormal
 721 basis of the orthogonal complement of the column subspace of P . Note $PP^\top + P_\perp P_\perp^\top = I_d$ and
 722 $P_\perp^\top P = O_{(d-k) \times k}$ by construction, so that we can write

$$\begin{aligned} (I_d - VV^\top)^{2p} &= (P(I_k - \Sigma)P^\top + P_\perp P_\perp^\top)^{2p} \\ &= P(I_k - \Sigma)^{2p}P^\top + P_\perp P_\perp^\top. \end{aligned}$$

723 Hence, the objective function can be lower bounded as

$$\begin{aligned} \text{tr}((I_d - VV^\top)^{2p}A) &= \text{tr}(P(I_k - \Sigma)^{2p}P^\top A) + \text{tr}(P_\perp^\top A P_\perp) \\ &\stackrel{(a)}{\geq} \text{tr}(P_\perp^\top A P_\perp) \\ &\stackrel{(b)}{\geq} \sum_{\ell=k+1}^d \lambda_\ell. \end{aligned}$$

724 Here, (a) follows since $P(I_k - \Sigma)^{2p}P^\top$ and A are both PSD and the inner product of two PSD matrices
 725 is always nonnegative. Further, (b) follows since $\text{tr}(P_\perp^\top A P_\perp)$ is minimized as $\sum_{\ell=k+1}^d \lambda_\ell$ when
 726 optimized over an orthogonal matrix $P_\perp \in \mathbb{R}^{d \times (d-k)}$. We now consider the equality condition. First,
 727 (b) holds with equality if and only if P_\perp consists of a bottom- $(d-k)$ eigenbasis of A , or equivalently
 728 P can be written as $P = VQ^\top$ by an orthogonal matrix $Q \in \mathbb{R}^{k \times k}$. Given that, we can write

$$\begin{aligned} \text{tr}(P(I_k - \Sigma)^{2p}P^\top A) &= \text{tr}((I_k - \Sigma)^{2p}QV^\top AVQ^\top) \\ &= \text{tr}((I_k - \Sigma)^{2p}Q\Lambda_{1:k}Q^\top) \\ &= \sum_{\ell=1}^k (1 - \mu_\ell)^{2p} \lambda_\ell \mathbf{q}_\ell \mathbf{q}_\ell^\top. \end{aligned}$$

729 This implies that (a) holds with equality if and only if $\mu_\ell = 1$ for all $\ell \in [k \wedge r]$. \square

730 A.3 Proof of Theorem 2

731 *Proof of Theorem 2.* The global minima of the jointly nested objective $\mathcal{L}_{\text{orbital}}^{(p)}(V; \alpha) :=$
 732 $\sum_{i=1}^k \alpha_i \mathcal{L}_{\text{orbital}}^{(p)}(V_{1:i})$ is achieved if and only if $\mathcal{L}_{\text{orbital}}^{(p)}(V_{1:i})$ is minimized for each $i \in [k]$. Hence, if
 733 the $V^* \in \mathbb{R}^{d \times k}$ achieves the global minima, then by the optimality condition from $\mathcal{L}_{\text{orbital}}^{(p)}(V_{1:i}^*)$, it
 734 must satisfy

$$\sum_{j=1}^i \mathbf{v}_j^* (\mathbf{v}_j^*)^\top \sum_{j=1}^i \mathbf{w}_j \mathbf{w}_j^\top$$

735 for each $i \in [k]$. By telescoping, this leads to $\mathbf{v}_i = \mathbf{w}_i$ for each i . \square

736 B On Implementation

737 B.1 Implementation of Sanger's Variant

738 Similar to the implementation of the sequential nesting by the following partially stop-gradient-ed
 739 objective

$$\mathcal{L}_{\text{orbital}}^{\text{seq}}(\mathbf{f}) := -2\text{tr}(M_\rho^{\text{seq}}[\mathbf{f}, \mathcal{T}\mathbf{f}]) + \text{tr}(M_\rho^{\text{seq}}[\mathbf{f}]M_\rho^{\text{seq}}[\mathbf{f}, \mathcal{T}\mathbf{f}]),$$

740 we can implement the Sanger variant by auto-differentiating the following objective

$$\mathcal{L}_{\text{orbital}}^{\text{sanger}}(\mathbf{f}) := -2\text{tr}(M_\rho[\mathbf{f}, \mathcal{T}\mathbf{f}]) + \text{tr}(M_\rho^{\text{seq}}[\mathbf{f}]\text{sg}[M_\rho[\mathbf{f}, \mathcal{T}\mathbf{f}]]).$$

741 With the appropriate stop-gradient operation, its gradient implements the Sanger variant.

742 B.2 Pseudocode for Nested Orbital Minimization Loss

743 Here, we include a unified PyTorch [37] implementation of various versions of OMM-1, i.e., OMM-1
 744 without any nesting (when nesting is None), OMM-1 with the joint nesting (when nesting is jnt),
 745 OMM-1 with the sequential nesting (when nesting is seq), and OMM-1 with the Sanger variant
 746 (when nesting is sanger).

```

747 1 class NestedOrbitalLoss:
748 2     def __init__(
749 3         self,
750 4         nesting=None,
751 5         n_modes=None,
752 6     ):
753 7         assert nesting in [None, 'jnt', 'seq', 'sanger']
754 8         self.nesting = nesting
755 9         if self.nesting == 'jnt':
756 10             assert n_modes is not None
757 11             self.vec_mask, self.mat_mask = get_joint_nesting_masks(weights=np.ones(
758 n_modes) / n_modes)
759 12         else:
760 13             self.vec_mask, self.mat_mask = None, None
761 14
762 15     def __call__(self, f: torch.Tensor, Tf: torch.Tensor):
763 16         # f: [b, k]
764 17         # Tf: [b, k]
765 18         if self.nesting == 'jnt':
766 19             M_f = compute_second_moment(f)
767 20             M_f_Tf = compute_second_moment(f, Tf)
768 21             operator_term = -2 * (torch.diag(self.vec_mask.to(f.device)) * M_f_Tf).mean
769 22             (0).sum()
770 23             metric_term = (self.mat_mask.to(f.device) * M_f_Tf * M_f).sum()
771 24         else: # if self.nesting in [None, 'seq', 'sanger']:
772 25             M_f = compute_second_moment(f, seq_nesting=self.nesting in ['seq', 'sanger']
773 26         ])
774 27             M_f_Tf = compute_second_moment(f, Tf, seq_nesting=self.nesting == 'seq')
775 28             if self.nesting == 'sanger':
776 29                 M_f_Tf = M_f_Tf.detach()
777 30                 operator_term = -2 * torch.trace(M_f_Tf)
778 31                 metric_term = (M_f_Tf * M_f).sum()
779 32
780 33         return operator_term + metric_term
781 34
782 35
783 36     def compute_second_moment(
784 37         f: torch.Tensor,
785 38         g: torch.Tensor = None,
786 39         seq_nesting: bool = False
787 40     ) -> torch.Tensor:
788 41         """
789 42         compute (optionally sequentially nested) second-moment matrix
790 43         M_ij = <f_i, g_j>
791 44         with partial stop-gradient handling when seq_nesting is True.
792 45
793 46         args
794 47         ----
795 48         f : (n, k) tensor
796 49         g : (n, k) tensor or None
797 50         seq_nesting : bool
798 51         """
799 52         if g is None:
800 53             g = f
801 54             n = f.shape[0]
802 55         if not seq_nesting:
803 56             return (f.T @ g) / n
804 57         else:

```

```

805 56     # apply partial stop-gradient
806 57     # lower-triangular: <f_i, sg[g_j]> for i > j
807 58     lower = torch.tril(f.T @ g.detach(), diagonal=-1)
808 59     # upper-triangular: <sg[f_i], g_j> for i < j
809 60     upper = torch.triu(f.detach().T @ g, diagonal=+1)
810 61     # diagonal: <f_i, g_i> (no stop-grad)
811 62     diag = torch.diag((f * g).sum(dim=0))
812 63     return (lower + diag + upper) / n
813 64
814 65
815 66 def get_joint_nesting_masks(weights: np.ndarray):
816 67     vector_mask = list(np.cumsum(list(weights)[::-1]))[::-1])
817 68     vector_mask = torch.tensor(np.array(vector_mask)).float()
818 69     matrix_mask = torch.minimum(
819 70         vector_mask.unsqueeze(1), vector_mask.unsqueeze(1).T
820 71     ).float()
821 72     return vector_mask, matrix_mask

```

822 C Details on Experimental Setups and Additional Results

823 In this section, we describe the detail experiment setups. All codebases to reproduce the experiments
824 will be made public upon acceptance of the paper. All experiments were conducted on a single GPU,
825 either a NVIDIA GeForce RTX 3090 (24GB) or a NVIDIA RTX A6000 (48GB).

826 C.1 Laplacian Representation Learning for Reinforcement Learning

827 We followed the experiment setup in [13] closely. Our implementation was built upon the codebase
828 of Gomez et al. [13].³ We visualize the RL environments in Figure 3.

- 829 • **Data generation:** For each experiment, we generated $N = 10^6$ transition samples from
830 a uniform random policy and uniform initial state distribution. The (x, y) coordinates are
831 given to a neural network as inputs.
- 832 • **Architecture:** We used a fully connected neural network with 3 hidden layers of 256
833 units with ReLU activations, with an output layer of $k = 11$ to learn the top- k Laplacian
834 eigenfunction representation.
- 835 • **Optimization:** We trained for 80,000 epochs using the Adam optimizer with a learning rate
836 of 10^{-3} . Additionally, we reproduce the ALLO training with identical number of transition
837 samples, epochs, and optimizer, with the suggested initial barrier coefficient $b = 2.0$ and
838 $\alpha_{\text{barrier}} = 0.01$ reported in [13]. We trained with NestedOMM for four random runs, and
839 ALLO with 10 runs. Since the smallest eigenvalue of each Laplacian matrix can be negative
840 in this case, but being bounded below by -1 (by the Gershgorin circle theorem), we add the
841 Laplacian matrix by the identity matrix to shift the spectrum.
- 842 • **Performance metric:** The performance is measured by the average of cosine similarities of
843 each mode, where degenerate spaces are handled by an oracle knowing the degeneracy.

844 C.2 Solving Schrödinger Equation

845 We followed the setup in [41] closely, implementing our code based on top of their codebase.⁴ Exactly
846 same configurations were used for both the Sanger variant and LoRA with sequential nesting. Since
847 the configuration is almost same as [41, Appendix E.1.1], we note the setup succinctly, and refer an
848 interested reader to therein for the detailed setup.

- 849 • **Data generation:** We opted to use a Gaussian distribution $\mathcal{N}(0, 16I_2)$ as a sampling distri-
850 bution, and generated new minibatch sample of size 512.

³Github repository: https://github.com/tarod13/laplacian_dual_dynamics

⁴Github repository: <https://github.com/jongharyu/neural-svd>

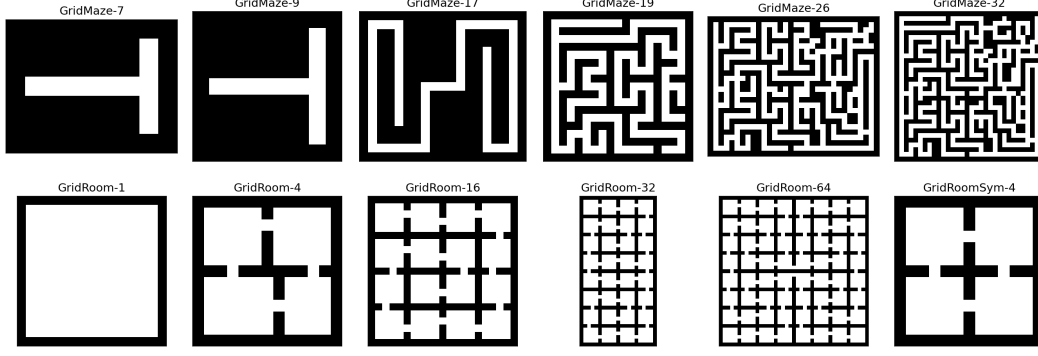


Figure 3: Grid environments for Laplacian representation learning.

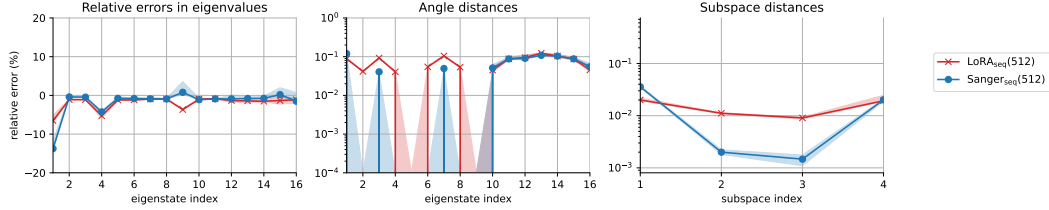


Figure 4: Summary of the 2D hydrogen experiment. The shaded region indicates \pm standard deviations.

- **Architecture:** We used 16 separate fully connected neural networks with 3 hidden layers of 128 units with the softplus activation. We also used the multi-scale Fourier features.
- **Optimization:** We trained the neural networks for 10^5 iterations. To enable a direct comparison between OMM and LoRA under consistent computational settings, we used five times fewer iterations than in [41]. While longer training may improve performance, our choice facilitates a controlled evaluation of relative efficacy. We note that we used Adam optimizer [21] with learning rate 10^{-4} with the cosine learning rate scheduler, instead of RMSprop, which we found to perform worse than Adam. We multiplied the operator by 100, but did not shift it by a multiple of identity.
- **Performance metric:** Following [41], we report the relative errors in eigenvalue estimates, angle distances for each mode (also similar to the Laplacian representation learning for RL), and subspace distances within each degenerate subspace. For each metric, we followed the same procedure defined in [41]. We ran 10 different training runs for each method and report average values with standard deviations.

The numerical comparisons are summarized in Figure 4. First, we note that the OMM_{seq} diverged quickly without any additional identity shift, as we alluded to earlier. The Sanger variant is comparable to or sometimes even better than LoRA_{seq} (i.e., $\text{NeuralSVD}_{\text{seq}}$) in terms of different metrics. Since LoRA_{seq} was the strongest method from [41], this result suggests that OMM can be also alternatively used in place of LoRA when we wish to decompose a PSD operator. As alluded to earlier, however, OMM cannot deal with unbounded differential operators (such as harmonic oscillators) inherently, while LoRA is capable of that.

C.3 Self-Supervised Contrastive Representation Learning

In this section, we describe the experiment setup for the image experiment in the main text, as well as an additional graph experiment.

C.3.1 Representation Learning for Images

For this experiment, we used the solo-learn codebase of da Costa et al. [4].⁵

⁵Github repository: <https://github.com/vturrisi/solo-learn>

- **Data generation:** We used the default augmentation for CIFAR-100 in the codebase. The exact configurations to reproduce the results will be shared upon acceptance.
- **Architecture:** We used ResNet-18 [17] as our backbone model and adopted two different feature encoding strategies: (1) we used a nonlinear projector of shape `Linear(feature_dim, 2048)-BatchNorm1D-ReLU-Linear(2048, 2048)-BatchNorm1D-ReLU-Linear(2048, 256)`; (2) similar to DirectCLR [20], we removed the projector and simply train the top $k = 64$ dimensions of the ResNet-18 feature as the top- k eigenfunctions using the OMM objective. In both cases, each feature vector is normalized by its ℓ_2 -norm following the standard convention.
- **Optimization:** We used the default optimization configuration of the codebase.⁶ we used LARS optimizer [51] with weight decay set to 0, initial learning rate of 0.3 governed by a cosine decay schedule, batch size 256, and 1000 epochs.
- **Evaluation:** We evaluate the representation based on the linear probe accuracy on the test split, trained by SGD with learning rate 0.1, batch size 256, and 100 epochs.

C.3.2 Representation Learning with Graph Data

Table 2: Summary of OGBN-products experiment (%). The model was trained once for each method, but the linear probe were trained for 10 different times for each case. The \pm ’s indicate the standard deviations. “Representation” refers to the linear probe accuracy based on the output of the MLP backbone, and “Embedding” refers to that based on the output of the projector, which is trained to fit the underlying eigenfunctions. The LoRA objective failed to yield convergent training dynamics.

	Finetuning		Correct & Smooth [19]	
	representation	embedding	representation	embedding
LoRA [41]	N/A	N/A	N/A	N/A
Neural Eigenmaps [6]	74.05 \pm 1.71	50.76 \pm 0.72	82.40 \pm 0.91	80.90 \pm 0.20
OMM (ours)	73.66 \pm 1.88	74.17\pm0.19	82.06 \pm 1.03	84.11\pm0.12

We can apply the orbital minimization principle to a graph data. Suppose that we are given an adjacency matrix $A \in \mathbb{R}^{N \times N}$, where A_{ij} encodes the connectivity between nodes i and j . In the spectral graph theory [43], it is well known that the lowest eigenvectors of (symmetrically normalized) graph Laplacian $L_{\text{sym}} := I - D^{-1/2}AD^{-1/2}$ encodes important properties of the underlying graph.

When the graph is large and when each node $i \in [N]$ is associated with a feature vector \mathbf{x}_i , computing the eigenvectors numerically might be cumbersome and extrapolation to new points is nontrivial. Hence, in this case, it is natural to learn eigenfunctions $\mathbf{f}(\mathbf{x})$ of the graph Laplacian as a function of the feature vector \mathbf{x} . In this section, we show the applicability of OMM in this scenario, and assess the quality of the learned eigenfunctions of the graph Laplacian in the node classification task.

We closely followed the experiment setup in the *Neural Eigenmaps* paper [6], implementing our code based on the codebase of Deng et al. [6].⁷ Neural Eigenmaps was proposed as a method to find eigenfunctions of a PSD operator similar to OMM, but it is a regularization-based approach and thus does not have the sharp global optimality that OMM enjoys. A practitioner also needs to tune the regularization parameter α in the framework, while OMM is hyperparameter-free.

- **Data:** We used the ogbn-products dataset [18], where the feature vector has 100 dimensions and the classification task has 47 classes. It is a large-scale node property prediction benchmark, and the accompanied graph consists of 2,449,029 nodes and 61,859,140 edges. The density of this graph is 2.06×10^{-5} , suggesting that the underlying graph is extremely sparse.
- **Architecture:** We parameterize the eigenfunctions by an 11-layer MLP encoder with a width of 2048 and residual connections [17], followed by the projector of same architecture

⁶We refer the reader to the configuration for SimCLR pretraining: <https://github.com/vturrisi/solo-learn/blob/main/scripts/pretrain/cifar/simclr.yaml>.

⁷Github repository: <https://github.com/thudzj/NEigenmaps>

for the image experiment. We used 8192-4096 hidden units for OMM and LoRA, and 8192-8192 for Neural Eigenmaps. We note that, with OMM, we did not require the feature normalization by ℓ_2 -norm. In contrast, Neural Eigenmaps quickly diverged without the ℓ_2 -norm normalization and thus used the normalization so that the feature has ℓ_2 -norm 10. Even worse, we also trained a model with LoRA [41] (i.e., NeuralSVD without nesting), but the training dynamics was unstable and diverged as well regardless of ℓ_2 -normalization.

- **Optimization:** Training was conducted over 20 epochs on the full set of nodes using the LARS optimizer [51] with a batch size of 16384, weight decay set to 0, and an initial learning rate of 0.3 with a cosine learning rate scheduler. We used the default hyperparameter $\alpha = 0.3$ for Neural Eigenmaps as suggested in the paper [6], which was selected based on linear probe accuracy on the validation set.
- **Evaluation:** Similar to the image experiment, we evaluate the representation based on the linear probe accuracy on the test split, trained by SGD with learning rate 0.01 and weight decay 10^{-3} , batch size 256, and 100 epochs. We consider two evaluation strategies. The first is to train a linear classifier directly on the original training labels. The second follows the *Correct & Smooth* (C&S) method of Huang et al. [19], which enhances node classification by first correcting the training labels using a graph-based error estimation and then smoothing the corrected labels via feature propagation. This procedure produces a refined supervision signal for training, often leading to improved downstream performance. We used the default configuration in the Neural Eigenmaps codebase for C&S.

Results. We summarize our result in Table 2.

- First, unlike the standard image contrastive representation learning setting and Neural Eigenmaps, we found that OMM was capable of training the final embedding trained to fit the eigenfunctions to become highly performant on the classification task. That is, remarkably, the linear probe performance from the embedding is better than the intermediate feature, which is the output of the MLP. We note that the drastic performance drop in Neural Eigenmaps from $\sim 74\%$ (representation) to $\sim 50\%$ (embedding) is a typical behavior. This implies, on the one hand, that while Neural Eigenmaps might provide a signal for the intermediate feature to capture relevant information about each node, the *embedding* might not be truly trained to fit the eigenfunctions and thus worse discriminative power. On the other hand, the good embedding performance of embedding (even better than representation) of OMM suggests that the OMM objective may behave better than the competitors.
- Second, we observe that the C&S postprocessing boosts the classification accuracy for all cases. Nonetheless, even after the application of C&S, we find that the performance of the OMM embedding is clearly the best.

D On the Benefit of Higher-Order OMM

In the self-supervised representation learning experiment, we observe sharp increase in the downstream task performance by using the OMM-2 objective $\mathcal{L}_{\text{orbital}}^{(2)}(\mathbf{V})$, and even better by using the mixed objective $\mathcal{L}_{\text{orbital}}^{(1)}(\mathbf{V}) + \mathcal{L}_{\text{orbital}}^{(2)}(\mathbf{V})$. In this section, we provide a theoretical argument on the practical benefit of the higher-order OMM based on a gradient analysis.

We analyze the gradient for the finite-dimensional case for simplicity, but the same argument is readily extended to the function case. Let $\mathbf{R} := \mathbf{I}_d - \mathbf{V}\mathbf{V}^\top$. Then, we can show, by chain rule, that the gradient of the OMM- p objective is

$$\begin{aligned}
\nabla_{\mathbf{V}} \mathcal{L}_{\text{orbital}}^{(p)}(\mathbf{V}) &= \nabla_{\mathbf{V}} \text{tr}((\mathbf{I}_d - \mathbf{V}\mathbf{V}^\top)^{2p} \mathbf{A}) \\
&= \nabla_{\mathbf{R}} \text{tr}(\mathbf{R}^{2p} \mathbf{A}) \nabla_{\mathbf{V}} \mathbf{R} \\
&= -2 \left(\sum_{i=0}^{2p-1} \mathbf{R}^i \mathbf{A} \mathbf{R}^{2p-1-i} \right) \mathbf{V} \\
&= -2(\mathbf{R}^{2p-1} \mathbf{A} + \mathbf{R}^{2p-2} \mathbf{A} \mathbf{R} + \dots + \mathbf{R} \mathbf{A} \mathbf{R}^{2p-2} + \mathbf{A} \mathbf{R}^{2p-1}) \mathbf{V}.
\end{aligned}$$

956 For the purpose of our analysis, we restrict our attention to the case where $R = VV^\top$ is idempotent,
 957 i.e., $R^2 = R$. Then the gradient expression simplifies to

$$\begin{aligned}\nabla_V \mathcal{L}_{\text{orbital}}^{(p)}(V) &= -2(RA + AR + (2p - 2)RAR)V \\ &= \nabla_V \mathcal{L}_{\text{orbital}}^{(1)}(V) - 4(p - 1)RARV.\end{aligned}$$

958 Hence, if we consider the Frobenius norm of the gradient,

$$\|\nabla_V \mathcal{L}_{\text{orbital}}^{(p)}(V)\|_F^2 = \|\nabla_V \mathcal{L}_{\text{orbital}}^{(1)}(V)\|_F^2 + \Delta,$$

959 where we let

$$\Delta := 16(p - 1)^2 \|RARV\|_F^2 - 8(p - 1) \text{tr} \left(\nabla_V \mathcal{L}_{\text{orbital}}^{(1)}(V)^\top RARV \right).$$

960 This shows that when $p > 1$ is sufficiently large, the gradient norm can be made strictly larger than the
 961 norm of $\nabla_V \mathcal{L}_{\text{orbital}}^{(1)}(V)$. This can improve convergence speed at a near convergence, especially when
 962 $\|I_k - V^\top V\|_F$ is close to 0, since the gradient norm of the original OMM gradient $\|\nabla_V \mathcal{L}_{\text{orbital}}^{(1)}(V)\|_F$
 963 becomes small proportional to $\|I_k - V^\top V\|_F$. In practical optimization, the flat minima may cause a
 964 too immature convergence, and the additional gradient signal from $p > 1$ can help escape the flat
 965 minima.

References

- [1] David R Bowler and Tsuyoshi Miyazaki. Methods in electronic structure calculations. *Rep. Prog. Phys.*, 75(3):036503, 2012. 1
- [2] James Chapman, Lennie Wells, and Ana Lawry Aguila. Unconstrained stochastic CCA: Unifying multiview and self-supervised learning. In *Int. Conf. Learn. Repr.*, 2024. URL <https://openreview.net/forum?id=PHLVmV88Zy>. 5
- [3] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *Proc. Int. Conf. Mach. Learn.*, pages 1597–1607. PMLR, 2020. 8, 9
- [4] Victor Guilherme Turrise da Costa, Enrico Fini, Moin Nabi, Nicu Sebe, and Elisa Ricci. solo-learn: A library of self-supervised methods for visual representation learning. *J. Mach. Learn. Res.*, 23(56):1–6, 2022. URL <http://jmlr.org/papers/v23/21-1155.html>. 21
- [5] Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural Comput.*, 5(4):613–624, 1993. 7
- [6] Zhijie Deng, Jiaxin Shi, Hao Zhang, Peng Cui, Cewu Lu, and Jun Zhu. Neural Eigenfunctions Are Structured Representation Learners. *arXiv*, October 2022. 22, 23
- [7] Zhijie Deng, Jiaxin Shi, and Jun Zhu. NeuralEF: Deconstructing kernels by deep neural networks. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proc. Int. Conf. Mach. Learn.*, volume 162 of *Proceedings of Machine Learning Research*, pages 4976–4992. PMLR, 17–23 Jul 2022. URL <https://proceedings.mlr.press/v162/deng22b.html>. 8
- [8] M T Entwistle, Z Schätzle, P A Erdman, J Hermann, and F Noé. 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. 1
- [9] Weiguo Gao, Yingzhou Li, and Bichen Lu. Global convergence of triangularized orthogonalization-free method. *Commun. Math. Sci.*, 21(1):195–218, October 2021. ISSN 1539-6746,1945-0796. doi: 10.4310/cms.2023.v21.n1.a9. 3, 5
- [10] Weiguo Gao, Yingzhou Li, and Bichen Lu. Triangularized Orthogonalization-Free Method for Solving Extreme Eigenvalue Problems. *J. Sci. Comput.*, 93(63):1–28, October 2022. doi: 10.1007/s10915-022-02025-0. 3, 5, 6
- [11] Ian Gemp, Brian McWilliams, Claire Vernade, and Thore Graepel. EigenGame: PCA as a Nash equilibrium. In *Int. Conf. Learn. Repr.*, 2021. 5
- [12] Ian Gemp, Brian McWilliams, Claire Vernade, and Thore Graepel. EigenGame unloaded: When playing games is better than optimizing. In *Int. Conf. Learn. Repr.*, 2022. URL <https://openreview.net/forum?id=So6YAqnqgMj>. 5
- [13] Diego Gomez, Michael Bowling, and Marlos C. Machado. Proper Laplacian representation learning. In *Int. Conf. Learn. Repr.*, 2024. URL <https://openreview.net/forum?id=7gLfQT52Nn>. 2, 7, 20
- [14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Adv. Neural Inf. Proc. Syst.*, volume 27, 2014. 7
- [15] David J Griffiths and Darrell F Schroeter. *Introduction to quantum mechanics*. Cambridge university press, 2018. 8
- [16] Jeff Z HaoChen, Colin Wei, Adrien Gaidon, and Tengyu Ma. Provable guarantees for self-supervised deep learning with spectral contrastive loss. In *Adv. Neural Inf. Proc. Syst.*, 2021. 1, 6, 9

- [17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pages 770–778, 2016. 22
- [18] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. In *Adv. Neural Inf. Proc. Syst.*, volume 33, pages 22118–22133, 2020. 22
- [19] Qian Huang, Horace He, Abhay Singh, Ser-Nam Lim, and Austin R Benson. Combining label propagation and simple models out-performs graph neural networks. In *Int. Conf. Learn. Repr.*, 2021. 22, 23
- [20] Li Jing, Pascal Vincent, Yann LeCun, and Yuandong Tian. Understanding dimensional collapse in contrastive self-supervised learning. In *Int. Conf. Learn. Repr.*, 2022. URL <https://openreview.net/forum?id=YevsQ05DEN7>. 9, 22
- [21] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Int. Conf. Learn. Repr.*, 2015. 5, 21
- [22] Vladimir R Kostic, Pietro Novelli, Riccardo Grazzi, Karim Lounici, and Massimiliano Pontil. Learning invariant representations of time-homogeneous stochastic dynamical systems. In *Int. Conf. Learn. Repr.*, 2024. URL <https://openreview.net/forum?id=twSnZwi0Im>. 1
- [23] Vladimir R Kostic, Gregoire Pacreau, Giacomo Turri, Pietro Novelli, Karim Lounici, and Massimiliano Pontil. Neural conditional probability for uncertainty quantification. In *Adv. Neural Inf. Proc. Syst.*, 2024. URL <https://openreview.net/forum?id=zXfhHJnMB2>. 6
- [24] T P Krasulina. The method of stochastic approximation for the determination of the least eigenvalue of a symmetrical matrix. *USSR Computational Mathematics and Mathematical Physics*, 9(6):189–195, January 1969. ISSN 0041-5553. doi: 10.1016/0041-5553(69)90135-9. 5
- [25] I E Lagaris, A Likas, and D I Fotiadis. 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. 1
- [26] Omer Levy and Yoav Goldberg. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 302–308, 2014. 9
- [27] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Adv. Neural Inf. Proc. Syst.*, volume 27, pages 2177–2185, 2014. 9
- [28] Xin Liu, Zaiwen Wen, and Yin Zhang. An efficient Gauss–Newton algorithm for symmetric low-rank product matrix approximations. *SIAM J. Optim.*, 25(3):1571–1608, 2015. 6
- [29] Jianfeng Lu and Kyle Thicke. Orbital minimization method with ℓ^1 regularization. *J. Comput. Phys.*, 336:87–103, May 2017. ISSN 0021-9991. doi: 10.1016/j.jcp.2017.02.005. 3
- [30] Marlos C. Machado, Clemens Rosenbaum, Xiaoxiao Guo, Miao Liu, Gerald Tesauero, and Murray Campbell. Eigenoption discovery through the deep successor representation. In *Int. Conf. Learn. Repr.*, 2018. URL <https://openreview.net/forum?id=Bk8ZcAxR->. 5, 7
- [31] Andreas Mardt, Luca Pasquali, Hao Wu, and Frank Noé. VAMPnets for deep learning of molecular kinetics. *Nature Comm.*, 9(1):5, January 2018. ISSN 2041-1723. doi: 10.1038/s41467-017-02388-1. URL <https://doi.org/10.1038/s41467-017-02388-1>. 1
- [32] Francesco Mauri and Giulia Galli. Electronic-structure calculations and molecular-dynamics simulations with linear system-size scaling. *Phys. Rev. B Condens. Matter*, 50(7):4316, 1994. 1, 3
- [33] Francesco Mauri, Giulia Galli, and Roberto Car. Orbital formulation for electronic-structure calculations with linear system-size scaling. *Phys. Rev. B Condens. Matter*, 47(15):9973, 1993. 1, 3

- [34] Erkki Oja. A simplified neuron model as a principal component analyzer. *J. Math. Biol.*, 15(3): 267–273, 1982. ISSN 0303-6812. doi: 10.1007/BF00275687. 5
- [35] Pablo Ordejón, David A Drabold, Richard M Martin, and Matthew P Grumbach. Linear system-size scaling methods for electronic-structure calculations. *Phys. Rev. B Condens. Matter*, 51(3): 1456, 1995. 1, 3
- [36] Pablo Ordejón, David A Drabold, Matthew P Grumbach, and Richard M Martin. Unconstrained minimization approach for electronic computations that scales linearly with system size. *Phys. Rev. B Condens. Matter*, 48(19):14646–14649, November 1993. ISSN 0163-1829. doi: 10.1103/PhysRevB.48.14646. 1, 3
- [37] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Adv. Neural Inf. Proc. Syst.*, Red Hook, NY, USA, 2019. Curran Associates Inc. 19
- [38] David Pfau, Stig Petersen, Ashish Agarwal, David GT Barrett, and Kimberly L Stachenfeld. Spectral inference networks: Unifying deep and spectral learning. In *Int. Conf. Learn. Repr.*, 2019. 8
- [39] David Pfau, James S Spencer, Alexander G D G Matthews, and W M C Foulkes. 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
- [40] David Pfau, Simon Axelrod, Halvard Sutterud, Ingrid von Glehn, and James S Spencer. Accurate computation of quantum excited states with neural networks. *Science*, 385(6711):eadn0137, 2024. 1
- [41] J. Jon Ryu, Xiangxiang Xu, Hasan Sabri Melihcan Erol, Yuheng Bu, Lizhong Zheng, and Gregory W. Wornell. Operator SVD with neural networks via nested low-rank approximation. In *Proc. Int. Conf. Mach. Learn.*, 2024. URL <https://openreview.net/forum?id=qESG5HaoJ>. 1, 2, 4, 6, 8, 9, 20, 21, 22, 23
- [42] Terence D Sanger. 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. 2, 5
- [43] Daniel Spielman. Spectral and algebraic graph theory. *Lecture notes (Yale University)*, 4:47, 2019. 22
- [44] Cheng Tang. Exponentially convergent stochastic k-PCA without variance reduction. In e, editor, *Adv. Neural Inf. Proc. Syst.*, volume 32. Curran Associates, Inc., April 2019. 5
- [45] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. 9
- [46] Kaixin Wang, Kuangqi Zhou, Qixin Zhang, Jie Shao, Bryan Hooi, and Jiashi Feng. Towards better Laplacian representation in reinforcement learning with generalized graph drawing. In *Proc. Int. Conf. Mach. Learn.*, pages 11003–11012. PMLR, 2021. 5
- [47] Lichen Wang, Jiaxiang Wu, Shao-Lun Huang, Lizhong Zheng, Xiangxiang Xu, Lin Zhang, and Junzhou Huang. An Efficient Approach to Informative Feature Extraction from Multimodal Data. In *Proc. AAAI Conf. Artif. Int.*, volume 33, pages 5281–5288, July 2019. doi: 10.1609/aaai.v33i01.33015281. 6
- [48] Zaiwen Wen, Chao Yang, Xin Liu, and Yin Zhang. Trace-penalty minimization for large-scale eigenspace computation. *J. Sci. Comput.*, 66:1175–1203, 2016. 6
- [49] Yifan Wu, George Tucker, and Ofir Nachum. The Laplacian in RL: Learning representations with efficient approximations. In *Int. Conf. Learn. Repr.*, 2019. URL <https://openreview.net/forum?id=HJLNpoA5YQ>. 7

- 1109 [50] Xiangxiang Xu and Lizhong Zheng. Neural feature learning in function space. *J. Mach. Learn.*
1110 *Res.*, 25(142):1–76, 2024. 6
- 1111 [51] Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks.
1112 *arXiv preprint arXiv:1708.03888*, 2017. 22, 23