Transformers as Provable Approximators of Sparse Principal Component Analysis

Qinyan Liu[1] **Yan Chen**[2] **Canhong Wen**[3]

[1]School of the Gifted Young, University of Science and Technology of China;
[2]Department of Statistics and Actuarial Science, The University of Hong Kong;
[3]Department of Statistics and Finance, University of Science and Technology of China
[1,3]Hefei, China; [2]Hong Kong SAR, China
susanaliu@mail.ustc.edu.cn, yanc25@hku.hk, wench@ustc.edu.cn

Abstract

Transformer architectures exhibit remarkable expressive power, enabling them to approximate diverse iterative algorithms in supervised and semi-supervised settings. However, their capabilities for fully unsupervised tasks remain largely unexplored, particularly in high-dimensional and noisy regimes where structured sparsity and spectral estimation are critical. Sparse Principal Component Analysis (Sparse PCA, or SPCA) is a fundamental high-dimensional unsupervised problem, where classical solvers such as the Truncated Power Method (TPower) provide strong guarantees but require careful initialization and sparsity tuning. In this work, we show that TPower provides a constructive algorithmic template for multi-layer Transformers to implement SPCA. Under a noisy spiked covariance model, we derive approximation and generalization guarantees decomposing total error into algorithmic approximation and statistical estimation. Beyond the constructive theory, the Transformer formulation suggests practical advantages such as amortized inference and parallel batch processing. Empirical studies on synthetic and real-world high-dimensional datasets show that Transformers trained with unsupervised objectives can recover sparse structures at levels comparable to classical SPCA baselines in the reported settings. Our results provide one of the first rigorous characterizations of Transformer architectures for noisy unsupervised sparse spectral estimation, with empirical validation in high-dimensional settings.

1 Introduction

Since its introduction in [23], the Transformer architecture has become a dominant force in machine learning. Beyond well-studied tasks such as natural language processing, computer vision, and reinforcement learning [7, 11, 12, 21], recent studies show that Transformers can perform statistical tasks and implicitly implement algorithmic procedures in in-context learning (ICL) [1, 15], but theoretical understanding largely remains restricted to supervised or semi-supervised settings [13], leaving their behavior in unsupervised regimes less understood.

Sparse Principal Component Analysis (Sparse PCA, or SPCA) [29] is a canonical unsupervised high-dimensional problem, widely used in genomics [18] and finance [29]. It extends classical PCA by enforcing sparsity on the principal components, which enhances interpretability but introduces a nonconvex optimization challenge. Classical SPCA algorithms such as the Truncated Power Method (TPower) [27] provide statistical guarantees but require careful initialization and sparsity tuning, motivating a data-driven approach.

Our Contributions. In this paper, we establish a constructive approximation framework showing that multi-layer Transformers can emulate TPower-style SPCA updates, and we complement this

STAI-X 2026 (Full Paper Track). Do not distribute.

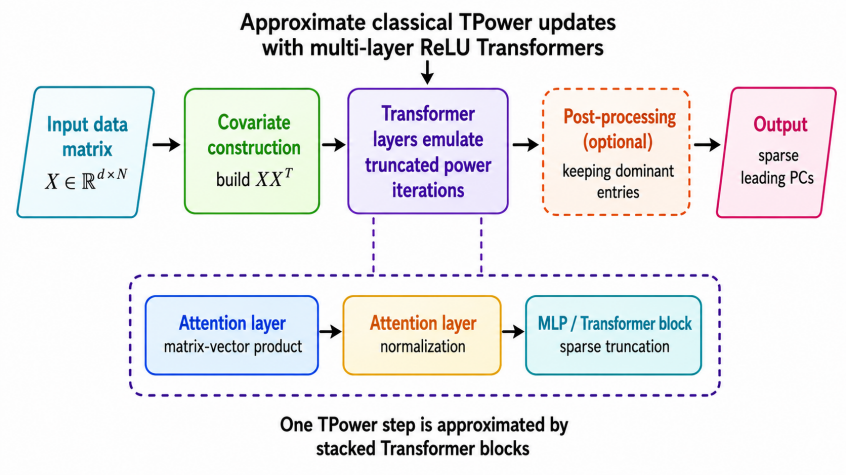

Figure 1: Overview of our Transformer-based SPCA framework.

theory with empirical evidence from unsupervised training. Figure 1 summarizes our framework, highlighting how Transformers emulate iterative SPCA updates. Our main contributions include:

- **Provable Approximation for Unsupervised SPCA under Noise:** Under the spiked covariance model and necessary assumptions, we construct ReLU-based Transformers that approximate SPCA using TPower as a constructive algorithmic template, yielding near-geometric convergence with small noise for the algorithmic error. Unlike supervised imitation approaches that learn from labels generated by a chosen SPCA algorithm, our method is trained directly with an unsupervised SPCA objective, avoiding the need to inherit the behavior of a particular labeling algorithm.
- **Statistical and Generalization Guarantees:** We decompose the recovery error into approximation, statistical, and generalization components, keeping the optimization error explicit as is common in Transformer expressivity analyses. Our framework applies to a broad class of SPCA losses and gives high-probability generalization bounds. Under standard learning-theoretic assumptions and an additional restricted sparse eigengap condition, we quantify the approximation–estimation tradeoff in terms of model capacity, sample size, and noise.
- **Empirical Validation on Noisy Data:** We conduct extensive experiments on both noisy synthetic datasets and high-dimensional gene expression datasets with input dimension 2000 and sample size 62. The empirical results demonstrate that Transformers consistently recover sparse principal components comparably to classical baselines, and in some settings achieve better performance. We further report runtime and memory results, showing efficient batched inference after training.

1.1 Related Works

SPCA Algorithms. Classical approaches of SPCA include the *Lasso*-based regression formulation [29], greedy-based algorithms such as PathSPCA [10], and the TPower algorithm [27]. Recent progress on efficient block-diagonalization [20] and fast approximation algorithms [6] further demonstrates ongoing interest in designing scalable and accurate methods for SPCA. Our work connects SPCA with the emerging view of Transformers as statistical algorithm approximators.

Transformers as Algorithm Approximators. A growing theoretical literature studies Transformers as algorithm emulators. The "Transformers-as-algorithms" perspective was advanced by Bai et al. [5], showing that multi-layer Transformers can implement gradient descent, ridge regression, Lasso, and in-context algorithm selection. Subsequent works extended this perspective to Newton-type updates, supervised N-classification, nearest-neighbor classification, and randomized iterative algorithms [2, 16, 19, 22, 26, 28]. While these prior studies mainly focus on supervised or semi-supervised

settings, our work addresses fully unsupervised SPCA, providing approximation and generalization guarantees under noisy conditions.

Unsupervised Learning in Transformers. Recent work has explored unsupervised capabilities, showing that large language models can perform meta-learning from task context, act as unsupervised in-context learners, and enable few-shot clustering [14, 24, 25]. On the theoretical side, He et al. [17] proved that pretrained Transformers can implement spectral methods, including PCA and two-component Gaussian Mixture Models (GMMs). However, their approach relies on pretraining with labels. Chen et al. [8] extended this line of research to multi-class GMMs and tensor power iterations, but only conducted experiments on low-dimensional synthetic datasets. These works demonstrate Transformer capabilities on unsupervised or weakly supervised tasks. Our work builds upon these results by moving from dense spectral methods to sparse feature extraction: we provide a theoretical analysis of Transformer-based SPCA under noise, establishing approximation and generalization guarantees for TPower-style sparse spectral updates under explicit assumptions. We further conduct experiments on high-dimensional real-world data, showing that Transformers trained with unsupervised objectives can closely match classical SPCA baselines.

2 Preliminaries

2.1 Notation

We use bold lowercase letters for vectors and bold uppercase letters for matrices. For a vector $\boldsymbol{v}$, $\|\boldsymbol{v}\|_2$ and $\|\boldsymbol{v}\|_0$ denote its Euclidean norm and sparsity level. For a matrix $\boldsymbol{A}$, let $\|\boldsymbol{A}\|_2$ and $\|\boldsymbol{A}\|_\infty := \max_{i,j} |A_{ij}|$ denote the operator norm and the entry-wise infinity norm, respectively. For a positive integer k, $[k]$ denotes the set $\{1, 2, \ldots, k\}$. We use $O(\cdot)$ in the standard asymptotic sense and $\widetilde{O}(\cdot)$ to suppress logarithmic factors.

2.2 Transformer Architecture

We use the rectified linear unit (ReLU) Transformer architecture as in [5, 17]. Given an input sequence $\boldsymbol{H} \in \mathbb{R}^{D\times N}$, an M-head self-attention layer and an MLP layer are defined as

$$\mathrm{Attn}_{\boldsymbol{\theta}_1}(\boldsymbol{H}) = \boldsymbol{H} + \frac{1}{N}\sum_{m=1}^{M} (\boldsymbol{V}_m\boldsymbol{H})\sigma\left((\boldsymbol{Q}_m\boldsymbol{H})^\top(\boldsymbol{K}_m\boldsymbol{H})\right), \qquad \mathrm{MLP}_{\boldsymbol{\theta}_2}(\boldsymbol{H}) = \boldsymbol{H} + \boldsymbol{W}_2\sigma(\boldsymbol{W}_1\boldsymbol{H}).$$

Here $\boldsymbol{\theta}_1 = \{(\boldsymbol{V}_m, \boldsymbol{Q}_m, \boldsymbol{K}_m)\}_{m\in[M]}$, $\boldsymbol{\theta}_2 = (\boldsymbol{W}_1, \boldsymbol{W}_2)$, and σ is the element-wise ReLU activation. An L-layer Transformer is a composition of attention and MLP layers followed by linear readout matrices:

$$TF_{\boldsymbol{\theta}}(\boldsymbol{H}) = \tilde{\boldsymbol{W}}_0 \left(\mathrm{MLP}_{\boldsymbol{\theta}_2^L} \circ \mathrm{Attn}_{\boldsymbol{\theta}_1^L} \circ \cdots \circ \mathrm{MLP}_{\boldsymbol{\theta}_2^1} \circ \mathrm{Attn}_{\boldsymbol{\theta}_1^1}\right)(\boldsymbol{H})\tilde{\boldsymbol{W}}_1,$$

where $\tilde{\boldsymbol{W}}_0$ and $\tilde{\boldsymbol{W}}_1$ are parameters that adjust the dimension of the Transformer output. The full parameter collection is denoted by $\boldsymbol{\theta}$.

2.3 SPCA and TPower

Assume that the data matrix $\boldsymbol{X} \in \mathbb{R}^{d\times N}$ is centered, where d is the number of features and N is the number of samples. Let $\boldsymbol{A} = \boldsymbol{X}\boldsymbol{X}^\top \in \mathbb{R}^{d\times d}$ denote the unnormalized empirical covariance matrix. For notational simplicity, we refer to $\boldsymbol{A}$ as the covariance matrix throughout, with the understanding that the scaling factor $\frac{1}{N}$ is absorbed into the analysis.

SPCA can be performed sequentially by recovering one sparse principal component at a time. In the single-unit SPCA problem, the goal is to find a unit vector $\boldsymbol{v} \in \mathbb{R}^d$ that maximizes the quadratic form associated with $\boldsymbol{A}$ under a sparsity constraint r:

$$\boldsymbol{v} = \arg\max_{\|\boldsymbol{v}\|_2=1,\ \|\boldsymbol{v}\|_0\le r} \boldsymbol{v}^\top\boldsymbol{A}\boldsymbol{v}. \tag{1}$$

To recover the top k sparse principal components, we adopt a deflation approach. Specifically, after obtaining the leading $i-1$ sparse eigenvectors $\{\boldsymbol{v}_j\}_{j=1}^{i-1}$, we construct a deflated matrix $\boldsymbol{A}_i$, where

Algorithm 1: TPower

Input : matrix $\boldsymbol{X} \in \mathbb{R}^{d\times N}$, initial vectors $\boldsymbol{v}_i^{(0)} \in \mathbb{R}^d$, $i = 1, 2, \ldots, k$, iterations τ
Output : set of leading k eigenvectors $\mathcal{V}$
Parameters : cardinality $r \in \{1, \ldots, p\}$, in which p denotes the rank of $\boldsymbol{X}$

Compute $\boldsymbol{A} = \boldsymbol{X}\boldsymbol{X}^\top \in \mathbb{R}^{d\times d}$. Initialize $\boldsymbol{A}_1 \leftarrow \boldsymbol{A}$. Let the set of sparse eigenvectors be $\mathcal{V} = \{\}$;
for $i \leftarrow 1$ **to** k **do**
  for $t \leftarrow 1$ **to** τ **do**
    Compute $\boldsymbol{v}_i'^{(t)} = \boldsymbol{A}_i \boldsymbol{v}_i^{(t-1)}$;
    Compute $\hat{\boldsymbol{v}}_i^{(t)} = \mathrm{Trunc}_r(\boldsymbol{v}_i'^{(t)})$, where $\mathrm{Trunc}_r(\boldsymbol{x})$ retains the r largest entries of $\boldsymbol{x}$ in absolute magnitude and sets all remaining entries to zero;
    Normalize $\boldsymbol{v}_i^{(t)} = \hat{\boldsymbol{v}}_i^{(t)} / \|\hat{\boldsymbol{v}}_i^{(t)}\|$;
  Let $\mathcal{V} \leftarrow \mathcal{V} \cup \{\boldsymbol{v}_i^{(\tau)}\}$;
  Update covariance matrix: $\boldsymbol{A}_{i+1} = \left(I_{d\times d} - \boldsymbol{v}_i^{(\tau)} \boldsymbol{v}_i^{(\tau)\top}\right) \boldsymbol{A}_i \left(I_{d\times d} - \boldsymbol{v}_i^{(\tau)} \boldsymbol{v}_i^{(\tau)\top}\right)$;

$\boldsymbol{A}_1 = \boldsymbol{A}$ and each subsequent matrix $\boldsymbol{A}_{i+1}$ is updated to remove the contribution of the previously recovered sparse components. The i-th sparse PC is then computed by solving the same single-unit SPCA problem in (1) with $\boldsymbol{A}$ replaced by the deflated matrix $\boldsymbol{A}_i$.

TPower can be viewed as a sparse extension of the classical power iteration method for SPCA. As shown in Algorithm 1, each inner iteration alternates among matrix–vector multiplication, truncation, and normalization to recover sparse leading principal components. Multiple sparse principal components are obtained sequentially through matrix deflation. The original TPower algorithm also includes a normalization step immediately after matrix–vector multiplication. Since this does not affect the iterations, we omit it for theoretical simplicity.

3 Transformer for SPCA

Figure 1 provides an overview of our Transformer method for SPCA. For theoretical analysis, the input is formed by augmenting $\boldsymbol{X}$ with auxiliary tokens that encode identity features, initialization vectors, and intermediate iterates. We design an auxiliary matrix $\boldsymbol{P}$ consisting of three parts:

1. *Identity Matrix.* We let $[\tilde{\boldsymbol{p}}_{2,1} \quad \ldots \quad \tilde{\boldsymbol{p}}_{2,N}] = \begin{bmatrix}\boldsymbol{I}_d & \boldsymbol{0}_{d\times(N-d)}\end{bmatrix}$. The identity matrix in $\boldsymbol{P}$ enables our construction to form and store the covariance $\boldsymbol{X}\boldsymbol{X}^\top$ in the forward propagation.
2. *Random Samples on the Hypersphere.* We let $\tilde{\boldsymbol{p}}_{2m+1,1}, m \in [k]$ be i.i.d. samples distributed on $\mathbb{S}^{d-1}$. The random samples on the sphere correspond to the initial vectors $\boldsymbol{v}_i^{(0)}$ for $i \in [k]$ in Algorithm 1.
3. *Placeholder.* For all the remaining vectors, we let them be $\boldsymbol{0} \in \mathbb{R}^{d\times 1}$. The placeholders in $\boldsymbol{P}$ record the intermediate results in the forward propagation.

Then we can construct the input matrix as $\boldsymbol{H} = \begin{bmatrix}\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N \\ \tilde{\boldsymbol{p}}_{1,1}, \ldots, \tilde{\boldsymbol{p}}_{1,N} \\ \vdots \\ \tilde{\boldsymbol{p}}_{\ell,1}, \ldots, \tilde{\boldsymbol{p}}_{\ell,N}\end{bmatrix} = \begin{bmatrix}\boldsymbol{X} \\ \boldsymbol{P}\end{bmatrix} \in \mathbb{R}^{D\times N}$, where $D = (2k+2)d$. The auxiliary matrix is designed for purely technical reasons. Moreover, our experiments suggest that such an auxiliary matrix is not necessary for the task, which is detailed in Section 5.

Remark 1. *Throughout the construction, we assume without loss of generality that the sequence length is at least d. Indeed, if $N < d$, we pad the input with zero dummy tokens by setting*

$$\bar{N} = \max\{N, d\}, \qquad \bar{\boldsymbol{X}} = [\boldsymbol{X}, 0_{d\times(\bar{N}-N)}] \in \mathbb{R}^{d\times\bar{N}}.$$

Since $\bar{\boldsymbol{X}}\bar{\boldsymbol{X}}^\top = \boldsymbol{X}\boldsymbol{X}^\top$, this padding does not change the covariance matrix, the SPCA objective, or the TPower iterates. Hence, in the auxiliary matrix construction, we use $\bar{N}$ tokens and write N for $\bar{N}$ for notational simplicity. Our empirical results suggest that the Transformer can perform well on a high-dimensional dataset with a small sample size.

Without access to raw data, let $\boldsymbol{H} = \begin{bmatrix} \boldsymbol{X}\boldsymbol{X}^\top, \boldsymbol{0} \\ \tilde{\boldsymbol{p}}_{2,1}, \ldots, \tilde{\boldsymbol{p}}_{2,N} \\ \vdots \\ \tilde{\boldsymbol{p}}_{\ell,1}, \ldots, \tilde{\boldsymbol{p}}_{\ell,N} \end{bmatrix} \in \mathbb{R}^{D-d\times N}$, and our theorem still holds.

The iterative nature of TPower and its matrix multiplications has a natural connection to Transformers. Each TPower step can be implemented using one or more Transformer layers, while the iterative updates are encoded through auxiliary tokens in forward propagation. This structural alignment forms the basis of our main result in Theorem 4.1, where we show that a suitably constructed ReLU Transformer can approximate TPower with provable error guarantees.

We train Transformers by minimizing an empirical SPCA loss over training instances $\{\boldsymbol{Z}_i\}_{i=1}^n$. Let

$$\mathcal{S}_{r,k} := \{\boldsymbol{V} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k] \in \mathbb{R}^{d\times k} : \boldsymbol{V}^\top \boldsymbol{V} = I_k,\ \|\boldsymbol{v}_j\|_0 \le r,\ j \in [k]\}, \quad \Phi(\boldsymbol{A}) := \max_{\boldsymbol{V}\in\mathcal{S}_{r,k}} \mathrm{Tr}(\boldsymbol{V}^\top \boldsymbol{A}\boldsymbol{V}).$$

We define the SPCA loss by $\ell(\boldsymbol{\theta}; \boldsymbol{Z}) = \Phi(\boldsymbol{A}) - \mathrm{Tr}(\hat{\boldsymbol{V}}_{\boldsymbol{\theta}}^\top \boldsymbol{A}\hat{\boldsymbol{V}}_{\boldsymbol{\theta}})$. The corresponding population and empirical risks are given by

$$R(\boldsymbol{\theta}) := \mathbb{E}_{\boldsymbol{Z}\sim\mathscr{P}}\ell(\boldsymbol{\theta}; \boldsymbol{Z}), \qquad \hat{R}_n(\boldsymbol{\theta}) := \frac{1}{n}\sum_{i=1}^n \ell(\boldsymbol{\theta}; \boldsymbol{Z}_i), \qquad \text{where } \boldsymbol{Z}_i \overset{i.i.d.}{\sim} \mathscr{P}.$$

Since Transformer outputs are continuous-valued, we apply a hard-thresholding post-processing step to enforce exact sparsity. For $\boldsymbol{\theta} \in \Theta$, we denote the resulting estimator by $\hat{\boldsymbol{V}}_{\boldsymbol{\theta}} = \mathrm{Post}(TF_{\boldsymbol{\theta}}(\boldsymbol{H}))$.

4 Theoretical Results

We assume a spiked covariance structure of the form

$$\boldsymbol{A} = \boldsymbol{A}^* + \boldsymbol{E} \in \mathbb{R}^{d\times d}, \quad \boldsymbol{A}^* = \sum_{i=1}^d \lambda_i^* \boldsymbol{v}_i^* (\boldsymbol{v}_i^*)^\top, \tag{2}$$

where $\boldsymbol{A}^*$ denotes the true covariance matrix, $\boldsymbol{E}$ is a symmetric noise matrix, and λ_i^* and $\boldsymbol{v}_i^*$ denote the corresponding eigenvalues and vectors with eigengap Δ. This formulation captures both sampling noise in empirical covariance estimation and model misspecification effects.

Let $\bar{r}$ denote the true sparsity level and s be the effective sparsity level. To quantify the noise level under sparsity constraints, we adopt the restricted perturbation error in [27], defined as:

$$\rho(\boldsymbol{E}, s) := \max_{\|\boldsymbol{v}\|_2=1, \|\boldsymbol{v}\|_0\le s} |\boldsymbol{v}^\top \boldsymbol{E}\boldsymbol{v}|.$$

We briefly introduce several auxiliary quantities that appear in our theoretical analysis. Their precise definitions are deferred to Appendix A.1. Let $\delta(s)$ denote the sparse noise-to-signal ratio, which depends on the restricted perturbation error $\rho(\boldsymbol{E}, s)$ and the eigengap of the population covariance matrix $\boldsymbol{A}^*$. The quantities $\mu_i(r)$ and $\mu_i^{(2)}(r)$ are contraction and stability factors taking values in $(0, 1)$ and they characterize the geometric decay of the finite-iteration error. Finally, let Λ be an upper bound on the spectral norm of $\boldsymbol{A}$ in equation 2.

4.1 Approximation Error

Theorem 4.1 shows that a sufficiently deep and wide Transformer can approximate TPower for sparse PCA, with an explicit error bound that accounts for truncation approximation, normalization approximation, and the finite-iteration TPower error under noisy settings.

Theorem 4.1 (Transformer Approximation of TPower)**.** *Under Assumptions 1-6 and auxiliary variables defined in Appendix A.1, for all $\epsilon_1, \epsilon_2 > 0$, there exist constants C, C', and a Transformer model with the number of attention layers $L = \big(6 + O(log(d)^2)\big)\,\tau k + 3k + 1$ with width polynomial in k, d, the number of MLP layers $k\tau$ with MLP width polynomial in $(d, 1/\epsilon_1, \Lambda)$, and the number*

of heads $M \le \max\{(\frac{d}{r})^d \frac{C'}{\epsilon_2^2}\log(1+\frac{\Lambda}{\epsilon_2}), d\}$ *such that after* τ *iterations, the final output* $\hat{\boldsymbol{V}} = [\hat{\boldsymbol{v}}_1, \ldots, \hat{\boldsymbol{v}}_k]$ *given by the Transformer model achieves*

$$\left\|\hat{\boldsymbol{v}}_j - \boldsymbol{v}_j^*\right\|_2 \le \underbrace{C\tau k\epsilon_1}_{:=\varepsilon_{trunc}} + \underbrace{C\tau k\Lambda\max\{|\Lambda-1|,1\}\epsilon_2}_{:=\varepsilon_{normalize}} + \underbrace{\sum_{i=1}^{k}\left(\mu_i(r)^\tau\sqrt{1-|\boldsymbol{v}_i^{(0),\top}\boldsymbol{v}_i^*|} + \frac{\sqrt{5}\delta(s)}{1-\mu_i^{(2)}(r)}\right)}_{:=\varepsilon_{tpower}}$$
$$:= \varepsilon_{\mathrm{app}}, \quad \forall j \in [k].$$

Moreover, consider all components jointly, let $\boldsymbol{V}^* = [\boldsymbol{v}_1^*, \ldots, \boldsymbol{v}_k^*]$. *There exists* $\boldsymbol{\theta}_{alg} \in \Theta$ *that satisfies* $d_{\mathrm{sub}}(\hat{\boldsymbol{V}}_{\boldsymbol{\theta}_{\mathrm{alg}}}, \boldsymbol{V}^*) \le k\varepsilon_{\mathrm{app}}$. *Here,* $d_{\mathrm{sub}}(\boldsymbol{V}, \boldsymbol{V}^*)$ *denotes the Frobenius norm of the difference between the projection matrices.*

Proof sketch. Using the input construction in Section 3, we construct a fixed-parameter Transformer that emulates TPower (Algorithm 1). A key observation in the construction is $\sigma(x) - \sigma(-x) \equiv x$, which enables ReLU Transformers to perform linear operations. The auxiliary tokens $\boldsymbol{P}$ serve as memory slots for storing the covariance matrix, initialization vectors, and intermediate iterates.

First, using the identity block in $\boldsymbol{P}$, a small number of attention heads writes the empirical covariance matrix $\boldsymbol{X}\boldsymbol{X}^\top$ into designated hidden-state coordinates and thus subsequent layers can operate on it directly. Then, for each component and the t-th iteration, attention layers implement the covariance-vector multiplication $\boldsymbol{A}_i\boldsymbol{v}_i^{(t-1)}$. The main technical challenge is approximating the truncation operator. Under the top-r separation condition, the support selected by $\mathrm{Trunc}_r(\cdot)$ is locally stable, enabling a ReLU Transformer block to uniformly approximate the truncation step. Additional layers approximate the normalization operation by an approximation of the inverse norm. Repeating these blocks for τ iterations yields an approximation of the TPower updates for a single component. The procedure is then sequentially applied to recover the first k sparse eigenvectors, with the Transformer implementing the same deflation update that maps $\boldsymbol{A}_i$ to $\boldsymbol{A}_{i+1}$ after each component is extracted. Finally, the readout matrices $\widetilde{\boldsymbol{W}}_0$ and $\widetilde{\boldsymbol{W}}_1$ select the tokens containing the final normalized sparse iterates. Summing the truncation approximation error and the normalization approximation error across all iterations and combining them with the finite-iteration TPower error yields the stated result.

The approximation bound consists of three components. The first term, ε_{trunc}, is the cumulative error from approximating the truncation operator by Transformer blocks. The second term, $\varepsilon_{normalize}$, comes from the approximation of the normalization step in each emulated power iteration. The last term, $\varepsilon_{\mathrm{tpower}}$, is the intrinsic finite-iteration TPower error under noise, which contains both the optimization bias from using τ iterations and the statistical perturbation effect through $\delta(s)$.

Under the assumptions, for all $i \in [k]$, $\mu_i(r) \in (0,1)$ and $\mu_i^{(2)}(r) \in (\sqrt{0.55}, 1)$. Hence $\mu_i(r)^\tau\sqrt{1-|\boldsymbol{v}_i^{(0)\top}\boldsymbol{v}_i^*|}$ decays geometrically as $\tau \to \infty$, while $\sqrt{5}\delta(s)/\left(1-\mu_i^{(2)}(r)\right) = O\left(\rho(\boldsymbol{E}, s)\right)$. Therefore, the TPower error term is uniformly bounded, and for sufficiently large τ it is dominated by $O(\rho(\boldsymbol{E}, s))$. The constants C, C' may hide dimension dependence, mainly due to the challenge of approximating high-dimensional functions by ReLU networks.

Example 4.1 (Noiseless SPCA)**.** *When the noise* $\boldsymbol{E} = 0$, $\delta(s) = 0$. ε_{tpower} *only contains the geometrically decaying term. When* $r = d$, $\bar{r} = d$, *noiseless SPCA further degenerates to the noiseless dense PCA setting studied in [17] with pretrained Transformers. However, we adopt a different method of iterative deflation which does not depend on the estimation of eigenvalues. This not only simplifies theoretical analysis but also avoids accumulating the estimation error of eigenvalues. Furthermore, approximating the discontinuous truncation step requires a more carefully designed worst-case construction.*

4.2 Generalization Error and Risk Tradeoff

We now convert the approximation result of Theorem 4.1 into a population risk bound. The overall error consists of two competing parts: an approximation error from the Transformer class and an estimation error driven by its complexity. Let P_Θ denote the pseudo-dimension of the Transformer class in the sense of [4]. Following the standard approach in the Transformer theory literature [5, 8, 17], we treat the optimization error $\varepsilon_{\mathrm{opt}}$ separately and focus on the approximation–estimation

tradeoff. The converged training loss curves in Fig. 5(b) and Appendix B.3.2 provide empirical support for this perspective.

Corollary 4.1.1 (Approximation–estimation tradeoff)**.** *Under additional assumptions 1-3 in Appendix A.3.2 and the conditions of Theorem 4.1, suppose the Transformer class is indexed by an approximation precision $\varepsilon > 0$, and write the full approximation error from Theorem 4.1 as*

$$\varepsilon_{\text{app}}(\varepsilon) \leq \varepsilon_{\text{tpower}} + C_{\text{app}}\varepsilon,$$

where $C_{\text{app}}\varepsilon$ is the tunable network approximation error. Suppose further that $P_\Theta(\varepsilon) \lesssim \varepsilon^{-q}$ for some $q > 0$. The tunable approximation–estimation tradeoff gives

$$R(\hat{\theta}_n) = \widetilde{O}(\varepsilon^2_{\text{tpower}}) + \widetilde{O}(n^{-2/(q+4)}) + Ck\mathbb{E}_Z[\rho(E, s)] + \epsilon_{\text{opt}}.$$

We can omit the last two terms since they are not controlled by Transformer approximation precision.

Furthermore, if the restricted sparse eigengap condition (additional assumption 4 in Appendix A.3.2) holds with parameter κ, then

$$\mathbb{E}_{\boldsymbol{Z}} d^2_{\text{sub}}(\hat{\boldsymbol{V}}_{\hat{\theta}_n}, \boldsymbol{V}^*) = \widetilde{O}(\varepsilon^2_{\text{tpower}}) + \widetilde{O}(n^{-2/(q+4)}),$$

where q represents the complexity of the model, and is therefore dependent on d.

This corollary highlights the fundamental approximation–estimation tradeoff in Transformer-based SPCA. The error decomposes into an irreducible statistical term and a tunable approximation term induced by the network class. Balancing these two terms yields the rate $\widetilde{O}(n^{-2/(q+4)})$, and under a restricted sparse eigengap condition, the same tradeoff transfers to the subspace estimation error.

5 Empirical Results

In synthetic settings where the latent structure and noise levels are fully controlled, we precisely assess the model's ability to capture the true sparse eigenvectors. We then study real-world benchmarks to verify Transformers' performance in practical high-dimensional, noisy scenarios. Details on experimental setup and additional empirical results are provided in Appendix B, including synthetic ablations (Appendix B.2.1), computational cost (Appendix B.2.2); as well as Pitprops loadings prediction and training loss curves (Appendix B.3). Throughout this section, PCs refer to the sparse eigenvectors $\boldsymbol{v}_i^*$ themselves instead of projected variables $\boldsymbol{X}\boldsymbol{v}_i^*$. The source code and reproduction instructions are publicly available at https://github.com/qinyanliu/transformer_spca.

5.1 Synthetic Data

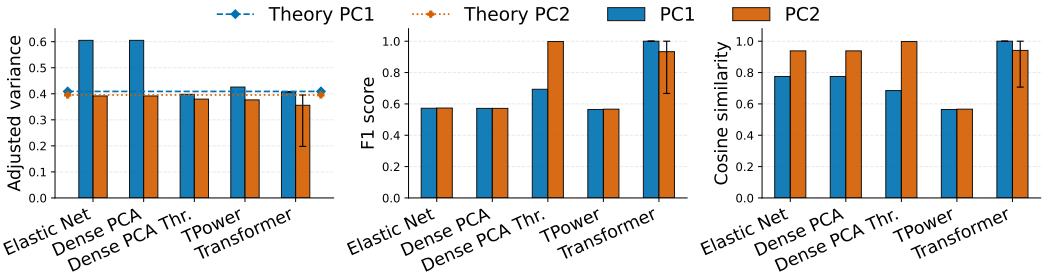

Figure 2: Comparison of Transformer SPCA with baselines on synthetic data. Dense PCA Thr. denotes Dense PCA Thresholding. Left: per-component adjusted variance for PC1 and PC2, with theory values shown as dashed lines. Middle: support F1, Right: Cosine similarity. Error bars denote the min-max range across five random seeds. Transformer SPCA achieves top performance across both PCs, closely matching theoretical values.

As shown in Figure 2, Transformer SPCA recovers PC1 and PC2 with adjusted variance (as defined in [29]) close to the theoretical values. It achieves higher support F1 and cosine similarity than the tested baselines in this synthetic setting. Despite multiple initialization strategies (as suggested in [27]),

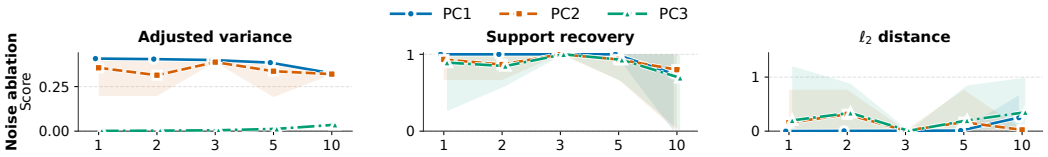

Figure 3: Noise ablation on the synthetic SPCA benchmark. Adjusted variance, support F1, and ℓ_2 distance to true loadings are shown per component (PC1–PC3) across varying levels of strong noise. Solid lines indicate mean over 5 model seeds; shaded regions denote min–max range. Experimental setup: eigengap=10. PC1 and PC2 are robust to noise until 5, while PC3 is more sensitive.

Table 1: Time cost comparison for batch prediction on 512 test samples. Values are reported as mean (std) over five seeds and 3 alternating repetitions. "inf. time" is short for inference time.

Method	Training time	Total inf. time	Per-sample inf. time
TF (train1024-L4-H4-E64)	14.4622 (0.8203) s	0.0364 (0.0019) s	0.0710 (0.0036) ms
Dense PCA Thresholding	–	0.0732 (0.0026) s	0.1430 (0.0050) ms
TPower (Combo R100)	–	55.1215 (0.4332) s	107.6592 (0.8460) ms
Elastic Net	–	560.7947 (28.8177) s	1095.3022 (56.2846) ms

TPower exhibits suboptimal recovery of PC1 and PC2, highlighting the challenge of distinguishing nearby sparse components. In contrast, Transformer SPCA robustly recovers both PCs close to theoretical values, demonstrating the practical advantage of the Transformer architecture in SPCA. An intrinsic property of TPower is that the errors for different sparse eigenvectors accumulate because of its sequential and iterative design. Therefore, for larger k, it is harder to achieve high precision. This is reflected in our theoretical bound ϵ_{tpower} and also explains the larger deviation of PC2.

As shown in Figure 3, under moderate noise levels, Transformer SPCA maintains adjusted variance close to theoretical values, high support recovery rates, and small ℓ_2 distance to true loadings. This suggests Transformers' ability to recover sparse PCs despite noisy settings. When the noise level exceeds half of the eigengap, the dominance of PC1 is damaged, and degradation occurs as expected. We further conduct ablation experiments to study architectural effects and examine their consistency with the trends predicted by our theory. See Appendix B.2. Note that the depth, embedding width and number of attention heads in Theorem 4.1 represent a worst-case constructive upper bound. Our ablation study shows that Transformers with depth 4, 4 attention heads, and embedding width 64 achieve strong SPCA recovery, suggesting that gradient-based training finds efficient parameter configurations far below the theoretical complexity ceiling.

Table 1 reports the computational cost of the main synthetic setting. Additional results of the training time, inference time, and GPU memory usage of Transformers with various configurations, as well as comparison with TPower of various initialization strategies, are reported in Appendix B.2.2. The results show that Transformer SPCA incurs a one-time training cost, but after training it supports efficient batched inference. This complements our empirical recovery results and shows that Transformers benefit significantly from amortized inference compared with classical CPU baselines.

5.2 Real-World Data

We evaluate our method on the 2000-dimensional Colon Cancer gene-expression dataset of Alon et al. [3], a widely used benchmark containing expression measurements from 62 colon tissue samples with both tumor and normal cases. Following the experimental setup in TPower [27] and PathSPCA [9], we consider the 500 genes with the largest variances, and study the resulting Proportion of Explained Variance (PEV) versus cardinality tradeoff of the first extracted PC. As illustrated in Figure 4, selecting the top 500 high-variance genes provides a straightforward unsupervised baseline. Despite removing 75% of the original features, the reduced dataset maintains nearly the same classification performance as the full dataset. This suggests that our pre-processing method effectively captures significant informative signals without relying on label information.

Figure 5 reports the PEV–cardinality trade-off and training convergence on the Colon Cancer dataset under unsupervised variance-based preprocessing. For each cardinality level, the Transformer curve is

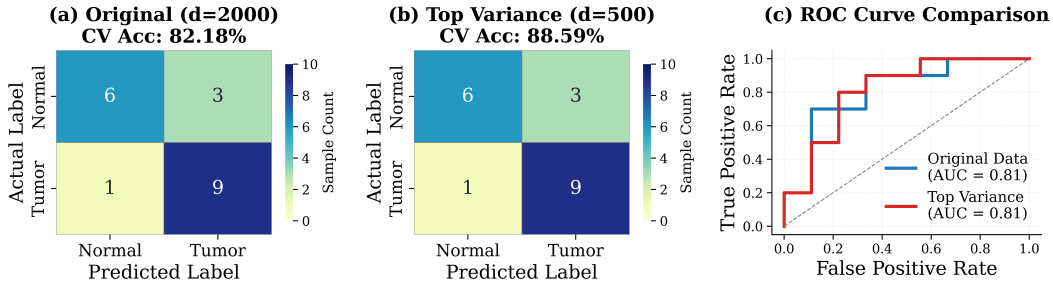

Figure 4: Classification performance comparison between original and top-variance selected datasets. (a) Confusion matrix for original data ($d = 2000$) with a 5-fold cross-validation accuracy of 82.18%. (b) Confusion matrix for top-variance data ($d = 500$), with a 5-fold cross-validation accuracy of 88.59%. (c) ROC curve comparison shows that the reduced dataset achieves essentially the same discriminative performance as the original dataset (AUC = 0.81 for both), while reducing the feature dimension from 2000 to 500.

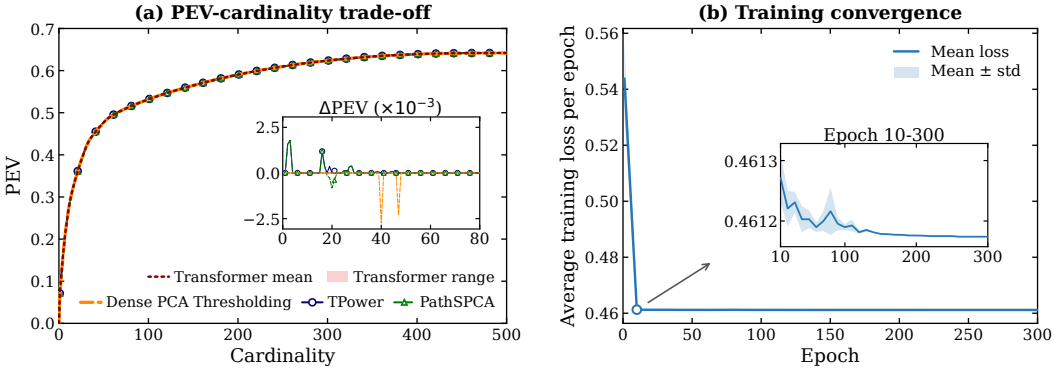

Figure 5: Results on the Colon Cancer dataset under unsupervised variance-based preprocessing. (a) PEV–cardinality trade-off for the first sparse PC. The Transformer curve reports the mean over five random seeds (42–46), with the shaded region denoting the min–max range. The inset shows ΔPEV relative to the Transformer mean for cardinality ≤ 80. The Transformer nearly overlaps with TPower, PathSPCA, and dense PCA thresholding across the full cardinality range. (b) Training convergence of the Transformer. The average training loss per epoch decreases rapidly in the early epochs and remains stable afterwards; the inset zooms in on the loss trajectory after epoch 10.

obtained by thresholding the learned loading vector and refitting PCA on the selected support. Across five random seeds, the Transformer curve is stable and nearly overlaps with TPower, PathSPCA, and dense PCA thresholding, indicating that the learned loading vector induces a support ordering comparable to classical SPCA solvers. This suggests that Transformers can function as SPCA solvers under high-dimensional real-world datasets with a small sample size. The training curve further shows that the unsupervised objective decreases rapidly in the first few epochs and remains stable afterwards, providing a convergence diagnostic for the learned Transformer estimator.

6 Conclusion

We provide both theoretical and empirical evidence that Transformers can serve as SPCA solvers. Unlike previous works, our framework is fully unsupervised and does not require labeled pretraining. The approximation theorem shows that TPower provides a constructive algorithmic template for Transformers to implement sparse spectral updates. Experiments show that trained models can match classical SPCA baselines on synthetic and real-world benchmarks. We further report runtime and memory results, showing efficient batched inference after training. These results support the view of Transformers as amortized approximators of sparse spectral algorithms. Extending the framework to data-adaptive choices of k and r is an interesting future direction, especially in light of Transformers' in-context algorithm selection ability [5].

Acknowledgment

We thank Haojun Wu for helpful discussions and careful checking of the theoretical derivations. We are grateful to Guangyu Li for valuable suggestions on experimental design and evaluation. We also thank Yiqi Wang for assistance with configuring the computational environment.

References

[1] Akyürek, E., Schuurmans, D., Andreas, J., Ma, T., and Zhou, D. (2023). What learning algorithm is in-context learning? Investigations with linear models. In *The Eleventh International Conference on Learning Representations*.

[2] Alcalde, A., Fantuzzi, G., and Zuazua, E. (2025). Exact sequence classification with hardmax Transformers. *arXiv preprint arXiv:2502.02270*.

[3] Alon, U., Barkai, N., Notterman, D. A., Gish, K., Ybarra, S., Mack, D., and Levine, A. J. (1999). Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of the National Academy of Sciences*, 96(12):6745–6750.

[4] Anthony, M. and Bartlett, P. L. (1999). *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, Cambridge.

[5] Bai, Y., Chen, F., Wang, H., Xiong, C., and Mei, S. (2023). Transformers as statisticians: Provable in-context learning with in-context algorithm selection. In *Thirty-seventh Conference on Neural Information Processing Systems*.

[6] Cai, J.-F., Xian, Z., and Ying, J. (2025). Fast and provable algorithms for sparse PCA with improved sample complexity. In *Forty-second International Conference on Machine Learning*.

[7] Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. (2021). Decision Transformer: Reinforcement learning via sequence modeling. In *Advances in Neural Information Processing Systems*, volume 34, pages 15084–15097.

[8] Chen, Z., Wu, R., and Fang, G. (2026). Transformers as unsupervised learning algorithms: A study on gaussian mixtures. In *The Fourteenth International Conference on Learning Representations*.

[9] d'Aspremont, A., Bach, F., and Ghaoui, L. E. (2008). Optimal solutions for sparse principal component analysis. *Journal of Machine Learning Research*, 9(42):1269–1294.

[10] d'Aspremont, A., El Ghaoui, L., Jordan, M. I., and Lanckriet, G. R. (2007). A direct formulation for sparse principal component analysis. *SIAM review*, 49(3):434–448.

[11] Devlin, J., Chang, M. W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186.

[12] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*.

[13] Fan, J., Rosu, P., Wang, A. T., Carin, L., and Cheng, X. (2026). In context semi-supervised learning. In *The Fourteenth International Conference on Learning Representations*.

[14] Gadetsky, A., Atanov, A., Jiang, Y., Gao, Z., Mighan, G. H., Zamir, A., and Brbic, M. (2025). Large (vision) language models are unsupervised in-context learners. In *The Thirteenth International Conference on Learning Representations*.

[15] Garg, S., Tsipras, D., Liang, P. S., and Valiant, G. (2022). What can Transformers learn in-context? A case study of simple function classes. In *Advances in Neural Information Processing Systems*, volume 35, pages 30583–30598.

[16] Giannou, A., Yang, L., Wang, T., Papailiopoulos, D., and Lee, J. D. (2024). How well can Transformers emulate in-context Newton's method? *arXiv preprint arXiv:2403.03183*.

[17] He, Y., Cao, Y., Chen, H.-Y., Wu, D., Fan, J., and Liu, H. (2025). Learning spectral methods by Transformers. *arXiv preprint arXiv:2501.01312*.

[18] Lee, D., Lee, W., Lee, Y., and Pawitan, Y. (2010). Super-sparse principal component analyses for high-throughput genomic data. *BMC Bioinformatics*, 11(1):296.

[19] Li, Z., Cao, Y., Gao, C., He, Y., Liu, H., Klusowski, J. M., Fan, J., and Wang, M. (2024). One-Layer Transformer provably learns one-nearest neighbor in context. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

[20] Pia, A. D., Zhou, D., and Zhu, Y. (2025). Efficient sparse PCA via block-diagonalization. *arXiv preprint arXiv:2410.14092*.

[21] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners. Technical report, OpenAI.

[22] Ren, R., Ouyang, S., Tang, H., and Liu, Y. (2025). Transformers as intrinsic optimizers: Forward inference through the energy principle. *arXiv preprint arXiv:2511.00907*.

[23] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30.

[24] Vettoruzzo, A., Braccaioli, L., Vanschoren, J., and Nowaczyk, M. (2025). Unsupervised meta-learning via in-context learning. In *The Thirteenth International Conference on Learning Representations*.

[25] Viswanathan, V., Gashteovski, K., Lawrence, C., Wu, T., and Neubig, G. (2024). Large language models enable few-shot clustering. *Transactions of the Association for Computational Linguistics*, 12:321–333.

[26] von Oswald, J., Kobayashi, S., Akram, Y., and Steger, A. (2025). Learning randomized algorithms with Transformers. In *The Thirteenth International Conference on Learning Representations*.

[27] Yuan, X. and Zhang, T. (2013). Truncated power method for sparse eigenvalue problems. *Journal of Machine Learning Research*, 14:899–925.

[28] Zappala, E. and Bagherian, M. (2025). Universal approximation of operators with Transformers and neural integral operators. *arXiv preprint arXiv:2409.00841*.

[29] Zou, H., Hastie, T., and Tibshirani, R. (2006). Sparse principal component analysis. *Journal of Computational and Graphical Statistics*, 15(2):265–286.

A Theoretical Appendix

A.1 Auxiliary Variables and Key Assumptions

Auxiliary variables. The following auxiliary variables inherited from the TPower paper [11] are used in the theoretical analysis.

Noise-to-signal ratio: $\delta(s) := \frac{\sqrt{2}\rho(\boldsymbol{E},s)}{\sqrt{\rho(\boldsymbol{E},s)^2+(\Delta-2\rho(\boldsymbol{E},s))^2}}$. This measures the relative strength of the noise matrix $\boldsymbol{E}$ against the eigengap Δ, dictating the fundamental resolution limit of sparse recovery.

Adjusted spectral ratio: $\gamma_i(s) := \frac{\lambda_i^*-\Delta+\rho(\boldsymbol{E},s)}{\lambda_i^*-\rho(\boldsymbol{E},s)}, \forall i \in [k]$. This represents the ratio of λ_i^* relative to the noise-affected boundary; values closer to 0 imply faster convergence.

Contraction and stability factors: $\mu_i(r) = \max\{\sqrt{1-\mu_i^{(1)}(r)}, \mu_i^{(2)}(r)\}, \forall i \in [k]$, where $\mu_i^{(1)}(r)$ and $\mu_i^{(2)}(r)$ are defined as:

$$\mu_i^{(1)}(r) = \frac{(1-\gamma_i(s)^2)u(1-u^2)}{2-(2\delta(s)+(\bar{r}/r)^{1/2})}, \quad \mu_i^{(2)}(r) = \sqrt{\left(1+3(\bar{r}/r)^{1/2}\right)\left(1-0.45(1-\gamma_i(s)^2)\right)}.$$

Key assumptions. Theorem 4.1 is stated under the following assumptions.

1. **Spiked Covariance:** Let p denote the rank of $\boldsymbol{A}^*$, therefore we have $k \le p$. Assume that the true eigenvalues of $\boldsymbol{A}^*$ are $\lambda_1^* > \lambda_2^* > \ldots > \lambda_p^* > \lambda_{p+1}^* = \ldots = 0$, $\boldsymbol{v}_i^*$ is the unit true eigenvector corresponding to λ_i^*. Let $\Delta := \min_{1\le i<j\le p+1} |\lambda_i^* - \lambda_j^*| > 0$ denote the eigengap of $\boldsymbol{A}^*$.
2. **Restricted perturbation error constraint:** $\rho(\boldsymbol{E}, s) \le \frac{\Delta}{2}$. This implies that $\delta(s) = O\left(\rho(\boldsymbol{E}, s)\right)$, $\delta(s) \le \sqrt{2}$, and that $\gamma_i(s) \le 1, \forall i \in [k]$.
3. **Sparsity of eigenvectors:** Each true eigenvector $\boldsymbol{v}_i^*$ corresponding to λ_i^* satisfies $\|\boldsymbol{v}_i^*\|_0 = \bar{r}$.
4. **Initialization:** For all $i \in [k]$, initial vectors satisfy $\|\boldsymbol{v}_i^{(0)}\|_2 = 1, \|\boldsymbol{v}_i^{(0)}\|_0 \le r$. Let $s = 2r + \bar{r}$ with $r \ge 4\bar{r}$. Additionally, there exists $u \in (0, 1]$, such that for all $i \in [k]$, $\mu_i^{(1)}(r) \in (0, 1), \mu_i^{(2)}(r) < 1$ and $|\boldsymbol{v}_i^{(0),\top}\boldsymbol{v}_i^*| \ge u + \delta(s)$.
5. **Bounded spectral norm:** There exists a positive constant Λ, such that $\|\boldsymbol{A}\|_2 \le \Lambda$.
6. **Top-r separation:** For any $\hat{\boldsymbol{v}}_i^{(t)}$ as an intermediate vector in TPower right after truncation, there exist Δ_{top}, such that $|\hat{\boldsymbol{v}}_i^{(t)}|_{(r)} - |\hat{\boldsymbol{v}}_i^{(t)}|_{(r+1)} \ge \Delta_{\text{top}}$.

For notational simplicity, we state the assumptions with a common eigengap Δ and sparsity level r. The analysis extends directly to component-specific eigengaps Δ_i and sparsity levels r_i by replacing the corresponding constants in the TPower contraction bounds.

Justifications. Assumptions 1-4 are all inherited from the TPower paper [11]. Assumptions 1-3 are standard in high-dimensional spectral analysis, ensuring a well-defined spiked structure and sufficient signal-to-noise ratio. When the eigengap is small, the target sparse PCs become intrinsically unstable, and both classical SPCA solvers and Transformer-based solvers may exhibit larger variation. When the perturbation error is close to half the eigengap, it becomes hard to identify neighboring eigenvalues and eigenvectors, as is shown in our noise ablation synthetic experiment (Fig. 3).

Assumption 4, regarding initialization, is more stringent under ultra-high dimensions. Since the SPCA objective is nonconvex, a purely arbitrary initialization may converge to an undesired local optimum or a non-leading sparse direction. The condition $|\boldsymbol{v}_i^{(0)^\top}\boldsymbol{v}_i^*| \ge u + \delta(s)$ ensures that the initialization lies in the basin of attraction of the target sparse eigenvector after accounting for the sparse perturbation level. The requirement $r \ge 4\bar{r}$ allows the working support to contain the true support with sufficient slack, which is also common in analyses of iterative sparse spectral algorithms. Our current theory does not remove the warm-start requirement of TPower. Empirically, however, we do not need to construct the augmented input $\boldsymbol{H}$ as in our theoretical analysis. Therefore, tuning the initialization vectors is not required for Transformers.

Assumption 5 is standard in spectral analysis and Transformer approximation studies. Similar bounded-domain or bounded-spectrum assumptions are common in spectral perturbation and neural-network approximation analyses [2, 3, 5]. A sufficient condition of Assumption 5 is that both $\boldsymbol{A}^*$ and $\boldsymbol{E}$ have bounded spectral norm. In practice, preprocessing approaches such as covariance normalization $\tilde{\boldsymbol{A}} = \boldsymbol{A}/\|\boldsymbol{A}\|_2$ can be used to ensure that this assumption is satisfied.

Generalizing our theorem to the diverging-spike setting without the bounded norm assumption is an interesting direction of future work, but it may require a quite different approach of error control and is beyond the scope of our paper.

Assumption 6 is to handle a technical issue specific to our constructive Transformer approximation: the hard top-r truncation operator Trunc_r is discontinuous. In ordinary PCA, the eigengap in Assumption 1 makes the target eigenspace identifiable and, through Davis–Kahan type perturbation bounds, ensures that small perturbations of the covariance matrix lead to small perturbations of the recovered eigenspace [4]. In sparse PCA, however, the estimator also depends on a discrete support selection step. Hence eigenvalue separation alone does not preclude ties or near-ties between the r-th and $(r+1)$-th largest coordinates of an intermediate iterate.

Therefore, for the worst-case approximation argument, we impose the coordinate-level separation $|\widehat{\boldsymbol{v}}_i^{(t)}|_{(r)} - |\widehat{\boldsymbol{v}}_i^{(t)}|_{(r+1)} \geq \Delta_{\mathrm{top}}$ which keeps the iterate away from the discontinuity boundary of Trunc_r. This guarantees that the selected support is locally stable under small approximation errors. Practically, the condition means that the sparse support is not ambiguous at the truncation threshold; if many coordinates are nearly tied around the threshold, exact support recovery is unstable for any hard-thresholding method. We use this assumption only as a sufficient condition for the uniform ReLU-Transformer approximation of the truncation step. It is not claimed to be necessary for the empirical Transformer solver, and our experiments do not explicitly enforce it.

A.2 Proof of Theorem 4.1

Given the above design, we are ready to prove the theorem.

Proof. Our proof can be dissected into the following steps: We construct a Transformer with fixed parameters that performs:

1. Computing the covariance matrix;
2. Approximating the truncated power iteration;
3. Iterative Deflation: Removing the truncated PCs (if $k \geq 1$);
4. Adjusting the dimensions of the final output through multiplying the two matrices $\tilde{\boldsymbol{W}}_0$ and $\tilde{\boldsymbol{W}}_1$ on the left and right.

If we do not have access to raw data, we can construct the input matrix as $\boldsymbol{H} = \begin{bmatrix} \boldsymbol{X}\boldsymbol{X}^\top, \boldsymbol{0} \\ \tilde{\boldsymbol{p}}_{2,1}, \ldots, \tilde{\boldsymbol{p}}_{2,N} \\ \vdots \\ \tilde{\boldsymbol{p}}_{\ell,1}, \ldots, \tilde{\boldsymbol{p}}_{\ell,N} \end{bmatrix} \in \mathbb{R}^{D-d\times N}$, and skip the first step.

In the following proof, $\tilde{\boldsymbol{H}}$ denotes the output of Transformer layers, while $\boldsymbol{H}$ denotes the approximation of Transformer outputs. When we write $f \leq Cg$, the constant C can be different each time. The $\frac{1}{N}$ in attention layers can be absorbed into the constructed parameter matrices.

Step 1. Computing the covariance matrix. To compute the covariance matrix $\boldsymbol{X}\boldsymbol{X}^\top$, we construct $\boldsymbol{H} = \begin{bmatrix} \boldsymbol{X}_1, \ldots, \boldsymbol{X}_N \\ \tilde{\boldsymbol{p}}_{1,1}, \ldots, \tilde{\boldsymbol{p}}_{1,N} \\ \tilde{\boldsymbol{p}}_{2,1}, \ldots, \tilde{\boldsymbol{p}}_{2,N} \\ \vdots \\ \tilde{\boldsymbol{p}}_{\ell,1}, \ldots, \tilde{\boldsymbol{p}}_{\ell,N} \end{bmatrix} = \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{P} \end{bmatrix} \in \mathbb{R}^{D\times N}$. We let the number of heads $M = 2$ and construct the

first covariance layer as follows,

$$\boldsymbol{V}_1^{cov} = -\boldsymbol{V}_2^{cov} = \begin{bmatrix} \mathbf{0}_{d\times D} \\ I_d \\ \mathbf{0} \end{bmatrix} \in \mathbb{R}^{D\times D},$$

$$\boldsymbol{Q}_1^{cov,\top}\boldsymbol{K}_1^{cov} = -\boldsymbol{Q}_2^{\top}\boldsymbol{K}_2 = \begin{bmatrix} \mathbf{0}_{d\times 2d} & I_d & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \in \mathbb{R}^{D\times D},$$

$$p_{1,j,m} = 0, \qquad p_{2,j,m} = \begin{cases} \mathbb{1}_{j=m} & \text{when } j \le d \\ 0 & \text{when } j > d \end{cases}, \quad \forall j \in [N] \text{ and } m \in [d],$$

$$p_{i,j,m} \text{ denotes the } m\text{-th component of vector } \tilde{\boldsymbol{p}}_{i,j}. \tag{3}$$

Under the above construction, we obtain that

$$\boldsymbol{Q}_1^{\top}\boldsymbol{K}_1\boldsymbol{H} = \begin{bmatrix} I_d & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \in \mathbb{R}^{D\times N}, \quad \boldsymbol{Q}_2^{\top}\boldsymbol{K}_2\boldsymbol{H} = \begin{bmatrix} -I_d & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \in \mathbb{R}^{D\times N},$$
$$\sigma(\boldsymbol{H}^{\top}\boldsymbol{Q}_1^{\top}\boldsymbol{K}_1\boldsymbol{H}) - \sigma(\boldsymbol{H}^{\top}\boldsymbol{Q}_2^{\top}\boldsymbol{K}_2\boldsymbol{H}) = \left[\boldsymbol{X}^{\top}, \mathbf{0}\right] \in \mathbb{R}^{N\times N}.$$

We further obtain that

$$\sum_{m=1}^{M} (\boldsymbol{V}_m\boldsymbol{H}) \times \sigma\left((\boldsymbol{Q}_m\boldsymbol{H})^{\top}(\boldsymbol{K}_m\boldsymbol{H})\right) = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \boldsymbol{X}\boldsymbol{X}^{\top} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \in \mathbb{R}^{D\times N}.$$

Therefore, the output is given by $\tilde{\boldsymbol{H}}^{cov} = \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{X}\boldsymbol{X}^{\top}, \mathbf{0} \\ \tilde{\boldsymbol{p}}_{2,1}, \ldots, \tilde{\boldsymbol{p}}_{2,N} \\ \vdots \\ \tilde{\boldsymbol{p}}_{\ell,1}, \ldots, \tilde{\boldsymbol{p}}_{\ell,N} \end{bmatrix}$.

Step 2. Approximating the truncated power iteration. Then we consider constructing a single attention layer that approximates the truncated power iteration. This step involves three important operations: (1) Obtaining the vector given by $\boldsymbol{X}\boldsymbol{X}^{\top}\boldsymbol{v}$. (2) Perform truncation $\mathrm{Trunc}_r(\boldsymbol{X}\boldsymbol{X}^{\top}\boldsymbol{v})$. (3) Approximation of the value of the inverse norm given by $1/\|\mathrm{Trunc}_r(\boldsymbol{X}\boldsymbol{X}^{\top}\boldsymbol{v})\|_2$. We show that one can use the multi-head ReLU Transformer to achieve both goals simultaneously.
Let $F = \mathrm{supp}(\boldsymbol{X}\boldsymbol{X}^{\top}\tilde{\boldsymbol{p}}_{3,1}, r)$ be the indices of $\boldsymbol{X}\boldsymbol{X}^{\top}\tilde{\boldsymbol{p}}_{3,1}$ with the largest r absolute values. We construct a Transformer layer whose parameters are given by

$$\boldsymbol{V}_1^{pow,1} = -\boldsymbol{V}_2^{pow,1} = \begin{bmatrix} \mathbf{0}_{4d\times 2d} & \mathbf{0} & \mathbf{0} \\ \mathbf{0}_{d\times 2d} & I_d & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix},$$

$$\boldsymbol{Q}_1^{pow,1} = -\boldsymbol{Q}_2^{pow,1} = \begin{bmatrix} \mathbf{0}_{3d\times d} & \mathbf{0} & \mathbf{0} \\ \mathbf{0}_{d\times d} & I_d & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix},$$

$$\boldsymbol{K}_1^{pow,1} = \boldsymbol{K}_2^{pow,1} = \begin{bmatrix} \mathbf{0}_{3d\times 3d} & \mathbf{0} & \mathbf{0} \\ \mathbf{0}_{d\times 3d} & I_d & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix},$$

$$\tilde{\boldsymbol{p}}_{4,j} = \mathbf{0}, \quad \forall j \in [N].$$

Note that auxiliary vectors $\tilde{\boldsymbol{p}}_{3,j}$ are drawn once at initialization and fixed throughout the construction. Therefore they can be used safely in the construction. Under the above constructions we have

$$\boldsymbol{Q}_1^{pow,1}\tilde{\boldsymbol{H}}^{cov} = -\boldsymbol{Q}_2^{pow,1}\tilde{\boldsymbol{H}}^{cov} = \begin{bmatrix} \mathbf{0}_{d\times N} \\ \boldsymbol{X}\boldsymbol{X}^{\top}, \mathbf{0} \\ \mathbf{0} \end{bmatrix},$$

$$\boldsymbol{K}_1^{pow,1}\tilde{\boldsymbol{H}}^{cov} = \boldsymbol{K}_2^{pow,1}\tilde{\boldsymbol{H}}^{cov} = \begin{bmatrix} \mathbf{0}_{3d\times N} \\ \tilde{\boldsymbol{p}}_{3,1}, \mathbf{0} \\ \mathbf{0} \end{bmatrix},$$

which implies that

$$\sum_{m\in\{1,2\}}(-1)^{m-1}\sigma\left((\boldsymbol{Q}_m^{pow,1}\tilde{\boldsymbol{H}}^{cov})^\top(\boldsymbol{K}_m^{pow,1}\tilde{\boldsymbol{H}}^{cov})\right)=\begin{bmatrix}\boldsymbol{X}\boldsymbol{X}^\top\tilde{\boldsymbol{p}}_{3,1} & \boldsymbol{0}_{d\times(N-1)}\\ \boldsymbol{0} & \boldsymbol{0}\end{bmatrix}.$$

Then we can show that

$$\begin{aligned}\tilde{\boldsymbol{H}}^{pow,1}-\tilde{\boldsymbol{H}}^{cov}&=\sum_{m\in\{1,2\}}\boldsymbol{V}_m^{pow,1}\tilde{\boldsymbol{H}}^{cov}\times\sigma\left((\boldsymbol{Q}_m^{pow,1}\tilde{\boldsymbol{H}}^{cov})^\top(\boldsymbol{K}_m^{pow,1}\tilde{\boldsymbol{H}}^{cov})\right)\\&=\begin{bmatrix}\boldsymbol{0}_{4d}\\ \boldsymbol{X}\boldsymbol{X}^\top\tilde{\boldsymbol{p}}_{3,1},\boldsymbol{0}_{d\times(N-1)}\\ \boldsymbol{0}\end{bmatrix}.\end{aligned}$$

Therefore, we conclude that the output of the first truncated power iteration layer is given by

$$\tilde{\boldsymbol{H}}^{pow,1}=\begin{bmatrix}\boldsymbol{X}\\ \boldsymbol{X}\boldsymbol{X}^\top,\boldsymbol{0}\\ \tilde{\boldsymbol{p}}_{2,1},\ldots,\tilde{\boldsymbol{p}}_{2,N}\\ \tilde{\boldsymbol{p}}_{3,1},\ldots,\tilde{\boldsymbol{p}}_{3,N}\\ \boldsymbol{X}\boldsymbol{X}^\top\tilde{\boldsymbol{p}}_{3,1},\boldsymbol{0}\\ \tilde{\boldsymbol{p}}_{5,1},\ldots,\tilde{\boldsymbol{p}}_{5,N}\\ \vdots\\ \tilde{\boldsymbol{p}}_{\ell,1},\ldots,\tilde{\boldsymbol{p}}_{\ell,N}\end{bmatrix}.$$

Since $\tilde{\boldsymbol{p}}_{3,1}$ is a unit vector,

$$\|\boldsymbol{X}\boldsymbol{X}^\top\tilde{\boldsymbol{p}}_{3,1}\|_2\le\|\boldsymbol{X}\boldsymbol{X}^\top\|_2\le\Lambda.$$

Using Lemma A.6, for any given $\epsilon_1>0$, there exists a Transformer block $\mathrm{TF}_{\boldsymbol{\theta}}(\cdot)$ with $3+S_d$ attention layers with number of heads $M\le d$ and 1 MLP layer with MLP width polynomial in $(d,1/\epsilon_1,\Lambda)$ such that

$$\left\|\mathrm{TF}_{\boldsymbol{\theta}}(\boldsymbol{X}\boldsymbol{X}^\top\tilde{\boldsymbol{p}}_{3,1})-\mathrm{Trunc}_r(\boldsymbol{X}\boldsymbol{X}^\top\tilde{\boldsymbol{p}}_{3,1})\right\|_\infty\le\epsilon_1.$$

Define

$$\boldsymbol{H}^{trunc,1}:=\begin{bmatrix}\boldsymbol{X}\\ \boldsymbol{X}\boldsymbol{X}^\top,\boldsymbol{0}\\ \tilde{\boldsymbol{p}}_{2,1},\ldots,\tilde{\boldsymbol{p}}_{2,N}\\ \tilde{\boldsymbol{p}}_{3,1},\ldots,\tilde{\boldsymbol{p}}_{3,N}\\ \mathrm{Trunc}_r(\boldsymbol{X}\boldsymbol{X}^\top\tilde{\boldsymbol{p}}_{3,1}),\boldsymbol{0}\\ \tilde{\boldsymbol{p}}_{5,1},\ldots,\tilde{\boldsymbol{p}}_{5,N}\\ \vdots\\ \tilde{\boldsymbol{p}}_{\ell,1},\ldots,\tilde{\boldsymbol{p}}_{\ell,N}\end{bmatrix}.$$

Therefore, we have

$$\left\|\mathrm{TF}_{\boldsymbol{\theta}}(\tilde{\boldsymbol{H}}^{pow,1})-\boldsymbol{H}^{trunc,1}\right\|_2\le\epsilon_1.$$

Then, using Lemma A.5, we design an additional attention layer that performs the normalization procedure. For any given $\epsilon_2>0$, there exists $M<(\frac{d}{r})^d\frac{C'(d)}{\epsilon_2^2}log(1+\frac{\Lambda}{\epsilon_2})$, such that there exists constants $\{\boldsymbol{a}_m\}_{m\in[M]}\subset\mathbb{S}^{N-1}$ and $\{c_m\}_{m\in[M]}\subset\mathbb{R}$ with

$$\left|\sum_{m=1}^M c_m\sigma\left(\boldsymbol{a}_m^\top\mathrm{Trunc}_r(\boldsymbol{X}\boldsymbol{X}^\top\tilde{\boldsymbol{p}}_{3,1})\right)-\frac{1}{\|\mathrm{Trunc}_r(\boldsymbol{X}\boldsymbol{X}^\top\tilde{\boldsymbol{p}}_{3,1})\|_2}+1\right|\le\epsilon_2.$$

We design a single attention layer with the following parameters $\forall m\in[M]$,

$$\boldsymbol{V}_m^{pow,2}=\begin{bmatrix}\boldsymbol{0}_{4d\times4d} & c_m\boldsymbol{I}_d & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0}\end{bmatrix},\qquad\boldsymbol{Q}_m^{pow,2}=\begin{bmatrix}\boldsymbol{0}_{d\times2d} & \boldsymbol{I}_d & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0}\end{bmatrix},$$

$$\boldsymbol{K}_m^{pow,2}=\begin{bmatrix}\boldsymbol{0}_{1\times4d} & \boldsymbol{a}_m^\top & \boldsymbol{0}\\ \vdots & & \\ \boldsymbol{0}_{1\times4d} & \boldsymbol{a}_m^\top & \boldsymbol{0}\\ \boldsymbol{0}_{(D-d)\times4d} & \boldsymbol{0} & \boldsymbol{0}\end{bmatrix}.$$

Under the above construction, we obtain

$$(\boldsymbol{Q}_m^{pow,2}\boldsymbol{H}^{trunc,1})^\top = \begin{bmatrix} I_{d\times d} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \qquad \boldsymbol{K}_m^{pow,2}\boldsymbol{H}^{trunc,1} = \begin{bmatrix} \boldsymbol{a}_m^\top \operatorname{Trunc}_r(\boldsymbol{X}\boldsymbol{X}^\top\tilde{\boldsymbol{p}}_{3,1}) & \mathbf{0} \\ \vdots & \\ \boldsymbol{a}_m^\top \operatorname{Trunc}(\boldsymbol{X}\boldsymbol{X}^\top\tilde{\boldsymbol{p}}_{3,1}) & \mathbf{0} \\ \mathbf{0}_{(D-d)\times 1} & \mathbf{0} \end{bmatrix}.$$

Then we have

$$\begin{aligned}\left\| \sum_{m=1}^M \boldsymbol{V}_m^{pow,2}\boldsymbol{H}^{trunc,1}\sigma\left((\boldsymbol{Q}_m^{pow,2}\boldsymbol{H}^{trunc,1})^\top(\boldsymbol{K}_m^{pow,2}\boldsymbol{H}^{trunc,1})\right) - \begin{bmatrix} \mathbf{0}_{4d\times N} \\ \frac{\operatorname{Trunc}_r(\boldsymbol{X}\boldsymbol{X}^\top\tilde{\boldsymbol{p}}_{3,1})}{\|\operatorname{Trunc}_r(\boldsymbol{X}\boldsymbol{X}^\top\tilde{\boldsymbol{p}}_{3,1})\|_2} - \operatorname{Trunc}_r(\boldsymbol{X}\boldsymbol{X}^\top\tilde{\boldsymbol{p}}_{3,1}), \mathbf{0} \\ \mathbf{0} \end{bmatrix} \right\|_2 \\ < \epsilon_2\|\operatorname{Trunc}_r(\boldsymbol{X}\boldsymbol{X}^\top\tilde{\boldsymbol{p}}_{3,1})\|_2 \le \epsilon_2\|\boldsymbol{X}\boldsymbol{X}^\top\tilde{\boldsymbol{p}}_{3,1}\|_2.\end{aligned}$$

Hence, using the fact that

$$\tilde{\boldsymbol{H}}^{pow,2} = \mathrm{TF}_\Theta(\tilde{\boldsymbol{H}}^{pow,1}) + \sum_{m=1}^M \boldsymbol{V}_m^{pow,2}\mathrm{TF}_\Theta(\tilde{\boldsymbol{H}}^{pow,1})\sigma\left(\left(\boldsymbol{Q}_m^{pow,2}\mathrm{TF}_\Theta(\tilde{\boldsymbol{H}}^{pow,1})\right)^\top\left(\boldsymbol{K}_m^{pow,2}\mathrm{TF}_\Theta(\tilde{\boldsymbol{H}}^{pow,1})\right)\right),$$

and applying the triangle inequality of the 2-norm, we obtain

$$\begin{aligned}&\left\|\tilde{\boldsymbol{H}}^{pow,2} - \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{X}\boldsymbol{X}^\top, \mathbf{0} \\ \tilde{\boldsymbol{p}}_{2,1},\dots,\tilde{\boldsymbol{p}}_{2,N} \\ \tilde{\boldsymbol{p}}_{3,1},\dots,\tilde{\boldsymbol{p}}_{3,N} \\ \frac{\operatorname{Trunc}_r(\boldsymbol{X}\boldsymbol{X}^\top\tilde{\boldsymbol{p}}_{3,1})}{\|\operatorname{Trunc}_r(\boldsymbol{X}\boldsymbol{X}^\top\tilde{\boldsymbol{p}}_{3,1})\|_2},\dots \mathbf{0} \\ \tilde{\boldsymbol{p}}_{5,1},\dots,\tilde{\boldsymbol{p}}_{5,N} \\ \vdots \\ \tilde{\boldsymbol{p}}_{\ell,1},\dots,\tilde{\boldsymbol{p}}_{\ell,N} \end{bmatrix}\right\|_2 \\ \le&\left\| \sum_{m=1}^M \boldsymbol{V}_m^{pow,2}\boldsymbol{H}^{trunc,1}\sigma\left((\boldsymbol{Q}_m^{pow,2}\boldsymbol{H}^{trunc,1})^\top(\boldsymbol{K}_m^{pow,2}\boldsymbol{H}^{trunc,1})\right) - \begin{bmatrix} \mathbf{0}_{4d\times N} \\ \frac{\operatorname{Trunc}_r(\boldsymbol{X}\boldsymbol{X}^\top\tilde{\boldsymbol{p}}_{3,1})}{\|\operatorname{Trunc}_r(\boldsymbol{X}\boldsymbol{X}^\top\tilde{\boldsymbol{p}}_{3,1})\|_2} - \operatorname{Trunc}_r(\boldsymbol{X}\boldsymbol{X}^\top\tilde{\boldsymbol{p}}_{3,1}), \mathbf{0} \\ \mathbf{0} \end{bmatrix}\right\|_2 \\ +&\left\| \sum_{m=1}^M \boldsymbol{V}_m^{pow,2}\mathrm{TF}_\Theta(\tilde{\boldsymbol{H}}^{pow,1})\sigma\left(\left(\boldsymbol{Q}_m^{pow,2}\mathrm{TF}_\Theta(\tilde{\boldsymbol{H}}^{pow,1})\right)^\top\left(\boldsymbol{K}_m^{pow,2}\mathrm{TF}_\Theta(\tilde{\boldsymbol{H}}^{pow,1})\right)\right)\right. \\ &\left. - \sum_{m=1}^M \boldsymbol{V}_m^{pow,2}\boldsymbol{H}^{trunc,1}\sigma\left((\boldsymbol{Q}_m^{pow,2}\boldsymbol{H}^{trunc,1})^\top(\boldsymbol{K}_m^{pow,2}\boldsymbol{H}^{trunc,1})\right)\right\|_2 \\ +&\left\|\mathrm{TF}_{\boldsymbol{\theta}}(\tilde{\boldsymbol{H}}^{pow,1}) - \boldsymbol{H}^{trunc,1}\right\|_2 \\ \le& C\epsilon_1 + \epsilon_2\|\boldsymbol{X}\boldsymbol{X}^\top\tilde{\boldsymbol{p}}_{3,1}\|_2.\end{aligned}$$

Then we construct another attention layer, which performs similar calculations as that of $pow, 1$ but switch the rows of $\tilde{\boldsymbol{p}}_{3,1}$ with that of $\frac{\operatorname{Trunc}_r(\boldsymbol{X}\boldsymbol{X}^\top\tilde{\boldsymbol{p}}_{3,1})}{\|\operatorname{Trunc}_r(\boldsymbol{X}\boldsymbol{X}^\top\tilde{\boldsymbol{p}}_{3,1})\|_2}$. Our construction for the next layer is given by

$$\begin{aligned}\boldsymbol{V}_1^{pow,3} &= -\boldsymbol{V}_2^{pow,3} = \begin{bmatrix} \mathbf{0}_{4d\times 2d} & \mathbf{0} & \mathbf{0} \\ \mathbf{0}_{d\times 2d} & I_d & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}, \\ \boldsymbol{Q}_1^{pow,3} &= -\boldsymbol{Q}_2^{pow,3} = \begin{bmatrix} \mathbf{0}_{3d\times d} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0}_{d\times d} & I_d & -I_d & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}, \\ \boldsymbol{K}_1^{pow,3} &= \boldsymbol{K}_2^{pow,3} = \begin{bmatrix} \mathbf{0}_{4d\times 4d} & \mathbf{0} & \mathbf{0} \\ \mathbf{0}_{d\times 4d} & I_d & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}, \quad \tilde{\boldsymbol{p}}_{4,j} = \mathbf{0}, \quad \forall j \in [N].\end{aligned}$$

Given the above construction, we can show that

$$\boldsymbol{Q}_2^{pow,3}\tilde{\boldsymbol{H}}^{pow,2} = -\boldsymbol{Q}_1^{pow,3}\tilde{\boldsymbol{H}}^{pow,2} = \begin{bmatrix} \mathbf{0}_{4d\times N} \\ \boldsymbol{X}\boldsymbol{X}^\top - I_d \quad \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \quad \boldsymbol{K}_2^{pow,3}\tilde{\boldsymbol{H}}^{pow,2} = \boldsymbol{K}_1^{pow,3}\tilde{\boldsymbol{H}}^{pow,2},$$

$$\left\| \boldsymbol{K}_2^{pow,3}\tilde{\boldsymbol{H}}^{pow,2} - \begin{bmatrix} \mathbf{0}_{4d\times N} \\ \frac{\mathrm{Trunc}_r(\boldsymbol{X}\boldsymbol{X}^\top\tilde{\boldsymbol{p}}_{3,1})}{\|\mathrm{Trunc}_r(\boldsymbol{X}\boldsymbol{X}^\top\tilde{\boldsymbol{p}}_{3,1})\|_2}, \mathbf{0} \\ \mathbf{0} \end{bmatrix} \right\|_2 \le C\epsilon_1 + \epsilon_2\|\boldsymbol{X}\boldsymbol{X}^\top\tilde{\boldsymbol{p}}_{3,1}\|_2.$$

Then, using the fact that given $\boldsymbol{X}_1, \boldsymbol{X}_2$ with $\|\boldsymbol{X}_1 - \boldsymbol{X}_2\|_2 \le \delta_0$, we have $\|(\boldsymbol{X}\boldsymbol{X}^\top - I_d)(\boldsymbol{X}_1 - \boldsymbol{X}_2)\|_2 \le \|(\boldsymbol{X}\boldsymbol{X}^\top - I_d)\|_2\delta_0$. Hence, collecting the above pieces, we have

$$\left\| \sum_{m=1}^{2} \boldsymbol{V}_m^{pow,3}\tilde{\boldsymbol{H}}^{pow,2}\sigma\left((\boldsymbol{Q}_2^{pow,3}\tilde{\boldsymbol{H}}^{pow,2})^\top(\boldsymbol{K}_2^{pow,3}\tilde{\boldsymbol{H}}^{pow,2}\right) - \begin{bmatrix} \mathbf{0}_{4d\times N} \\ \frac{(\boldsymbol{X}\boldsymbol{X}^\top)\,\mathrm{Trunc}_r(\boldsymbol{X}\boldsymbol{X}^\top\tilde{\boldsymbol{p}}_{3,1})}{\|\mathrm{Trunc}_r(\boldsymbol{X}\boldsymbol{X}^\top\tilde{\boldsymbol{p}}_{3,1})\|_2} - \frac{\mathrm{Trunc}_r(\boldsymbol{X}\boldsymbol{X}^\top\tilde{\boldsymbol{p}}_{3,1})}{\|\mathrm{Trunc}_r(\boldsymbol{X}\boldsymbol{X}^\top\tilde{\boldsymbol{p}}_{3,1})\|_2}, \mathbf{0} \\ \mathbf{0} \end{bmatrix} \right\|_2$$

$$\le C\epsilon_1 + \epsilon_2\|(\boldsymbol{X}\boldsymbol{X}^\top - I_d)\|_2 \left\|\boldsymbol{X}\boldsymbol{X}^\top\tilde{\boldsymbol{p}}_{3,1}\right\|_2.$$

Henceforth, one can further show that

$$\left\| \tilde{\boldsymbol{H}}^{pow,3} - \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{X}\boldsymbol{X}^\top, \mathbf{0} \\ \tilde{\boldsymbol{p}}_{2,1}, \ldots, \tilde{\boldsymbol{p}}_{2,N} \\ \tilde{\boldsymbol{p}}_{3,1}, \ldots, \tilde{\boldsymbol{p}}_{3,N} \\ \frac{\boldsymbol{X}\boldsymbol{X}^\top\mathrm{Trunc}_r(\boldsymbol{X}\boldsymbol{X}^\top\tilde{\boldsymbol{p}}_{3,1})}{\|\mathrm{Trunc}_r(\boldsymbol{X}\boldsymbol{X}^\top\tilde{\boldsymbol{p}}_{3,1})\|_2}, \mathbf{0} \\ \tilde{\boldsymbol{p}}_{5,1}, \ldots, \tilde{\boldsymbol{p}}_{5,N} \\ \vdots \\ \tilde{\boldsymbol{p}}_{\ell,1}, \ldots, \tilde{\boldsymbol{p}}_{\ell,N} \end{bmatrix} \right\|_2 \le C\epsilon_1 + \epsilon_2\|(\boldsymbol{X}\boldsymbol{X}^\top - I_d)\|_2\|\boldsymbol{X}\boldsymbol{X}^\top\tilde{\boldsymbol{p}}_{3,1}\|_2.$$

Consider we are doing in total of τ truncated power iterations, we can set $\forall\tau \in \mathbb{N}^*$,

$$\begin{aligned} &\boldsymbol{V}_m^{pow,2\tau} = \boldsymbol{V}_m^{pow,2}, \quad \boldsymbol{Q}_m^{pow,2\tau} = \boldsymbol{Q}_m^{pow,2}, \quad \boldsymbol{K}_m^{pow,2\tau} = \boldsymbol{K}_m^{pow,2}, \\ &\boldsymbol{V}_m^{pow,2\tau+1} = \boldsymbol{V}_m^{pow,3}, \quad \boldsymbol{Q}_m^{pow,2\tau+1} = \boldsymbol{Q}_m^{pow,3}, \quad \boldsymbol{K}_m^{pow,2\tau+1} = \boldsymbol{K}_m^{pow,3}. \end{aligned}$$

Between attention layers $pow, 2\tau$ and $pow, 2\tau+1$, the same Transformer block $\mathrm{TF}_{\boldsymbol{\theta}}(\cdot)$ with $3 + S_d$ attention layers and 1 MLP layer is inserted, which means that a total of $(6 + S_d)\tau$ layers of Transformer layers are constructed in a complete Truncated Power iteration. Recall that for any $t \in \tau$,

$$\tilde{\boldsymbol{p}}_{3,1}^{(t)} = \frac{\mathrm{Trunc}_r(\boldsymbol{X}\boldsymbol{X}^\top\tilde{\boldsymbol{p}}_{3,1}^{(t-1)})}{\left\|\mathrm{Trunc}_r(\boldsymbol{X}\boldsymbol{X}^\top\tilde{\boldsymbol{p}}_{3,1}^{(t-1)})\right\|_2}, \quad \tilde{\boldsymbol{p}}_{3,1}^{(0)} = \tilde{\boldsymbol{p}}_{3,1}$$

Therefore, $\tilde{\boldsymbol{p}}_{3,1}^{(t)}$ is a unit vector, so we always have $\|\boldsymbol{X}\boldsymbol{X}^\top\tilde{\boldsymbol{p}}_{3,1}^{(t)}\|_2 \le \|\boldsymbol{X}\boldsymbol{X}^\top\|_2$ Then, using the triangle inequality of the 2-norm, we can show that for $\tau \in \mathbb{N}$,

$$\left\| \tilde{\boldsymbol{H}}^{pow,2\tau} - \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{X}\boldsymbol{X}^\top, \mathbf{0} \\ \tilde{\boldsymbol{p}}_{2,1}, \ldots, \tilde{\boldsymbol{p}}_{2,N} \\ \tilde{\boldsymbol{p}}_{3,1}, \ldots, \tilde{\boldsymbol{p}}_{3,N} \\ \tilde{\boldsymbol{p}}_{3,1}^{(\tau)}, \mathbf{0} \\ \tilde{\boldsymbol{p}}_{5,1}, \ldots, \tilde{\boldsymbol{p}}_{5,N} \\ \vdots \\ \tilde{\boldsymbol{p}}_{\ell,1}, \ldots, \tilde{\boldsymbol{p}}_{\ell,N} \end{bmatrix} \right\|_2 \le C\tau\epsilon_1 + \tau\epsilon_2\|(\boldsymbol{X}\boldsymbol{X}^\top - I_d)\|_2\|\boldsymbol{X}\boldsymbol{X}^\top\|_2.$$

Step 3. Removing the truncated PCs. After τ iterates on the power method, we need to remove the principal term from the matrix $\boldsymbol{X}\boldsymbol{X}^\top$, achieved through iterative deflation: $\boldsymbol{A}^{(i+1)} = \left(I_{d\times d} - \boldsymbol{v}_\tau^{(i)}\boldsymbol{v}_\tau^{(i)\top}\right)\boldsymbol{A}^{(i)}\left(I_{d\times d} - \boldsymbol{v}_\tau^{(i)}\boldsymbol{v}_\tau^{(i)\top}\right)$.

We start from $i = 1$, where $\boldsymbol{A}^1 = \boldsymbol{X}\boldsymbol{X}^\top$. Define the approximation of the previous layer's output

$$\boldsymbol{H}^{pow,2\tau} := \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{X}\boldsymbol{X}^\top, \boldsymbol{0} \\ \tilde{\boldsymbol{p}}_{2,1}, \ldots, \tilde{\boldsymbol{p}}_{2,N} \\ \tilde{\boldsymbol{p}}_{3,1}, \ldots, \tilde{\boldsymbol{p}}_{3,N} \\ \tilde{\boldsymbol{p}}_{3,1}^{(\tau)}, \boldsymbol{0} \\ \tilde{\boldsymbol{p}}_{5,1}, \ldots, \tilde{\boldsymbol{p}}_{5,N} \\ \vdots \\ \tilde{\boldsymbol{p}}_{\ell,1}, \ldots, \tilde{\boldsymbol{p}}_{\ell,N} \end{bmatrix} = \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{A}^{(1)}, \boldsymbol{0} \\ \tilde{\boldsymbol{p}}_{2,1}, \ldots, \tilde{\boldsymbol{p}}_{2,N} \\ \tilde{\boldsymbol{p}}_{3,1}, \ldots, \tilde{\boldsymbol{p}}_{3,N} \\ \tilde{\boldsymbol{p}}_{3,1}^{(\tau)}, \boldsymbol{0} \\ \tilde{\boldsymbol{p}}_{5,1}, \ldots, \tilde{\boldsymbol{p}}_{5,N} \\ \vdots \\ \tilde{\boldsymbol{p}}_{\ell,1}, \ldots, \tilde{\boldsymbol{p}}_{\ell,N} \end{bmatrix}.$$

First we construct a Transformer layer whose parameters are given by

$$\begin{aligned} \boldsymbol{V}_1^{rpc,1} &= -\boldsymbol{V}_2^{rpc,1} = \begin{bmatrix} \boldsymbol{0}_{d\times 4d} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0}_{d\times 4d} & I_d & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}, \\ \boldsymbol{Q}_1^{rpc,1} &= -\boldsymbol{Q}_2^{rpc,1} = \begin{bmatrix} \boldsymbol{0}_{d\times 4d} & I_d & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}, \\ \boldsymbol{K}_1^{rpc,1} &= \boldsymbol{K}_2^{rpc,1} = \begin{bmatrix} \boldsymbol{0}_{d\times d} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0}_{d\times d} & -I_d & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}. \end{aligned}$$

Under the above constructions we have

$$\begin{aligned} \boldsymbol{V}_1^{rpc,1}\boldsymbol{H}^{pow,2\tau} &= -\boldsymbol{V}_2^{rpc,1}\boldsymbol{H}^{pow,2\tau} = \begin{bmatrix} \boldsymbol{0}_{d\times N} \\ \tilde{\boldsymbol{p}}_{3,1}^{(\tau)}, \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix}, \\ \boldsymbol{Q}_1^{rpc,1}\boldsymbol{H}^{pow,2\tau} &= -\boldsymbol{Q}_2^{rpc,1}\boldsymbol{H}^{pow,2\tau} = \begin{bmatrix} \tilde{\boldsymbol{p}}_{3,1}^{(\tau)}, \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix}, \\ \boldsymbol{K}_1^{rpc,1}\boldsymbol{H}^{pow,2\tau} &= \boldsymbol{K}_2^{rpc,1}\boldsymbol{H}^{pow,2\tau} = \begin{bmatrix} -\boldsymbol{X}\boldsymbol{X}^\top, \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix}, \end{aligned}$$

which implies that

$$\sum_{m\in\{1,2\}} (\boldsymbol{V}_m^{rpc,1}\boldsymbol{H}^{pow,2\tau})\sigma\left((\boldsymbol{Q}_m^{rpc,1}\boldsymbol{H}^{pow,2\tau})^\top(\boldsymbol{K}_m^{rpc,1}\boldsymbol{H}^{pow,2\tau})\right) = \begin{bmatrix} \boldsymbol{0}_{d\times N} \\ -\tilde{\boldsymbol{p}}_{3,1}^{(\tau)}\tilde{\boldsymbol{p}}_{3,1}^{(\tau)\top}\boldsymbol{X}\boldsymbol{X}^\top & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}.$$

Define

$$\boldsymbol{H}^{rpc,1} := \begin{bmatrix} \boldsymbol{X} \\ (I_d - \tilde{\boldsymbol{p}}_{3,1}^{(\tau)}\tilde{\boldsymbol{p}}_{3,1}^{(\tau)\top})\boldsymbol{X}\boldsymbol{X}^\top, \boldsymbol{0} \\ \tilde{\boldsymbol{p}}_{2,1}, \ldots, \tilde{\boldsymbol{p}}_{2,N} \\ \tilde{\boldsymbol{p}}_{3,1}, \ldots, \tilde{\boldsymbol{p}}_{3,N} \\ \tilde{\boldsymbol{p}}_{3,1}^{(\tau)}, \boldsymbol{0} \\ \tilde{\boldsymbol{p}}_{5,1}, \ldots, \tilde{\boldsymbol{p}}_{5,N} \\ \vdots \\ \tilde{\boldsymbol{p}}_{\ell,1}, \ldots, \tilde{\boldsymbol{p}}_{\ell,N} \end{bmatrix}.$$

Recall that

$$\left\| \tilde{\boldsymbol{H}}^{pow,2\tau} - \boldsymbol{H}^{pow,2\tau} \right\|_2 \leq C\tau\epsilon_1 + C\tau\epsilon_2 \|(\boldsymbol{X}\boldsymbol{X}^\top - I_d)\|_2 \|\boldsymbol{X}\boldsymbol{X}^\top\|_2.$$

Therefore, using the triangle inequality of the 2-norm, we conclude that the output of the first removal of truncated PCs satisfies

$$\left\| \tilde{\boldsymbol{H}}^{rpc,1} - \boldsymbol{H}^{rpc,1} \right\|_2 \leq C\tau\epsilon_1 + C\tau\epsilon_2 \|(\boldsymbol{X}\boldsymbol{X}^\top - I_d)\|_2 \|\boldsymbol{X}\boldsymbol{X}^\top\|_2.$$

Then we construct the next layer whose parameters are given by

$$\begin{aligned}
\boldsymbol{V}_1^{rpc,2} &= -\boldsymbol{V}_2^{rpc,2} = \begin{bmatrix} \mathbf{0}_{5d\times 4d} & \mathbf{0} & \mathbf{0} \\ \mathbf{0}_{d\times 4d} & I_d & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}, \\
\boldsymbol{Q}_1^{rpc,2} &= -\boldsymbol{Q}_2^{rpc,2} = \begin{bmatrix} \mathbf{0}_{d\times 4d} & I_d & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}, \\
\boldsymbol{K}_1^{rpc,2} &= \boldsymbol{K}_2^{rpc,2} = \begin{bmatrix} \mathbf{0}_{d\times 2d} & I_d & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}, \quad \tilde{\boldsymbol{p}}_{5,j} = 0, \quad \forall j \in [N].
\end{aligned}$$

Therefore we have

$$\begin{aligned}
\boldsymbol{V}_1^{rpc,2}\boldsymbol{H}^{rpc,1} &= -\boldsymbol{V}_2^{rpc,2}\boldsymbol{H}^{rpc,1} = \begin{bmatrix} \mathbf{0}_{5d\times N} \\ \tilde{\boldsymbol{p}}_{3,1}^{(\tau)}, \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \\
\boldsymbol{Q}_1^{rpc,2}\boldsymbol{H}^{rpc,1} &= -\boldsymbol{Q}_2^{rpc,2}\boldsymbol{H}^{rpc,1} = \begin{bmatrix} \tilde{\boldsymbol{p}}_{3,1}^{(\tau)}, \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \\
\boldsymbol{K}_1^{rpc,2}\boldsymbol{H}^{rpc,1} &= \boldsymbol{K}_2^{rpc,2}\boldsymbol{H}^{rpc,1} = \begin{bmatrix} I_d, \mathbf{0} \\ \mathbf{0} \end{bmatrix},
\end{aligned}$$

which implies that

$$\sum_{m\in\{1,2\}} (\boldsymbol{V}_m^{rpc,2}\boldsymbol{H}^{rpc,1})\sigma\left((\boldsymbol{Q}_m^{rpc,2}\boldsymbol{H}^{rpc,1})^\top(\boldsymbol{K}_m^{rpc,2}\boldsymbol{H}^{rpc,1})\right) = \begin{bmatrix} \mathbf{0}_{5d\times N} \\ \tilde{\boldsymbol{p}}_{3,1}^{(\tau)}\tilde{\boldsymbol{p}}_{3,1}^{(\tau)\top} \quad \mathbf{0} \\ \mathbf{0} \quad \mathbf{0} \end{bmatrix}.$$

Define

$$\boldsymbol{H}^{rpc,2} := \begin{bmatrix} \boldsymbol{X} \\ (I_d - \tilde{\boldsymbol{p}}_{3,1}^{(\tau)}\tilde{\boldsymbol{p}}_{3,1}^{(\tau)\top})\boldsymbol{X}\boldsymbol{X}^\top, \mathbf{0} \\ \tilde{\boldsymbol{p}}_{2,1}, \ldots, \tilde{\boldsymbol{p}}_{2,N} \\ \tilde{\boldsymbol{p}}_{3,1}, \ldots, \tilde{\boldsymbol{p}}_{3,N} \\ \tilde{\boldsymbol{p}}_{3,1}^{(\tau)}, \mathbf{0} \\ \tilde{\boldsymbol{p}}_{3,1}^{(\tau)}\tilde{\boldsymbol{p}}_{3,1}^{(\tau)\top}, \mathbf{0} \\ \tilde{\boldsymbol{p}}_{6,1}, \ldots, \tilde{\boldsymbol{p}}_{6,N} \\ \vdots \\ \tilde{\boldsymbol{p}}_{\ell,1}, \ldots, \tilde{\boldsymbol{p}}_{\ell,N} \end{bmatrix}.$$

Therefore, using the triangle inequality of the 2-norm, we conclude that the output of the first removal of truncated PCs satisfies

$$\left\| \tilde{\boldsymbol{H}}^{rpc,2} - \boldsymbol{H}^{rpc,2} \right\|_2 \leq C\tau\epsilon_1 + C\tau\epsilon_2 \|(\boldsymbol{X}\boldsymbol{X}^\top - I_d)\|_2 \|\boldsymbol{X}\boldsymbol{X}^\top\|_2.$$

Then we construct the third layer whose parameters are given by

$$\begin{aligned}
\boldsymbol{V}_1^{rpc,3} &= -\boldsymbol{V}_2^{rpc,3} = \begin{bmatrix} \mathbf{0}_{d\times d} & \mathbf{0} & \mathbf{0} \\ \mathbf{0}_{d\times d} & I_d & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}, \\
\boldsymbol{Q}_1^{rpc,3} &= -\boldsymbol{Q}_2^{rpc,3} = \begin{bmatrix} \mathbf{0}_{d\times 5d} & I_d & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}, \\
\boldsymbol{K}_1^{rpc,3} &= \boldsymbol{K}_2^{rpc,3} = \begin{bmatrix} \mathbf{0}_{d\times 2d} & -I_d & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}.
\end{aligned}$$

Therefore we have

$$\boldsymbol{V}_1^{rpc,3}\boldsymbol{H}^{rpc,2} = -\boldsymbol{V}_2^{rpc,3}\boldsymbol{H}^{rpc,2} = \begin{bmatrix} \mathbf{0}_{d\times N} \\ (I_d - \tilde{\boldsymbol{p}}_{3,1}^{(\tau)}\tilde{\boldsymbol{p}}_{3,1}^{(\tau)\top})\boldsymbol{X}\boldsymbol{X}^\top, \mathbf{0} \\ \mathbf{0} \end{bmatrix},$$

$$\boldsymbol{Q}_1^{rpc,3}\boldsymbol{H}^{rpc,2} = -\boldsymbol{Q}_2^{rpc,3}\boldsymbol{H}^{rpc,2} = \begin{bmatrix} \tilde{\boldsymbol{p}}_{3,1}^{(\tau)}\tilde{\boldsymbol{p}}_{3,1}^{(\tau)\top}, \mathbf{0} \\ \mathbf{0} \end{bmatrix},$$

$$\boldsymbol{K}_1^{rpc,3}\boldsymbol{H}^{rpc,2} = \boldsymbol{K}_2^{rpc,3}\boldsymbol{H}^{rpc,2} = \begin{bmatrix} -I_d, \mathbf{0} \\ \mathbf{0} \end{bmatrix},$$

which implies that

$$\sum_{m\in\{1,2\}} (\boldsymbol{V}_m^{rpc,3}\boldsymbol{H}^{rpc,2})\sigma\left((\boldsymbol{Q}_m^{rpc,3}\boldsymbol{H}^{rpc,2})^\top(\boldsymbol{K}_m^{rpc,3}\boldsymbol{H}^{rpc,2})\right) = \begin{bmatrix} \mathbf{0}_{d\times N} \\ -(I_d - \tilde{\boldsymbol{p}}_{3,1}^{(\tau)}\tilde{\boldsymbol{p}}_{3,1}^{(\tau)\top})\boldsymbol{X}\boldsymbol{X}^\top\tilde{\boldsymbol{p}}_{3,1}^{(\tau)}\tilde{\boldsymbol{p}}_{3,1}^{(\tau)\top} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}.$$

Define

$$\boldsymbol{H}^{rpc,3} := \begin{bmatrix} \boldsymbol{X} \\ (I_d - \tilde{\boldsymbol{p}}_{3,1}^{(\tau)}\tilde{\boldsymbol{p}}_{3,1}^{(\tau)\top})\boldsymbol{X}\boldsymbol{X}^\top(I_d - \tilde{\boldsymbol{p}}_{3,1}^{(\tau)}\tilde{\boldsymbol{p}}_{3,1}^{(\tau)\top}), \mathbf{0} \\ \tilde{\boldsymbol{p}}_{2,1}, \ldots, \tilde{\boldsymbol{p}}_{2,N} \\ \tilde{\boldsymbol{p}}_{3,1}, \ldots, \tilde{\boldsymbol{p}}_{3,N} \\ \tilde{\boldsymbol{p}}_{3,1}^{(\tau)}, \mathbf{0} \\ \tilde{\boldsymbol{p}}_{3,1}^{(\tau)}\tilde{\boldsymbol{p}}_{3,1}^{(\tau)\top}, \mathbf{0} \\ \tilde{\boldsymbol{p}}_{6,1}, \ldots, \tilde{\boldsymbol{p}}_{6,N} \\ \vdots \\ \tilde{\boldsymbol{p}}_{\ell,1}, \ldots, \tilde{\boldsymbol{p}}_{\ell,N} \end{bmatrix}.$$

Therefore, using the triangle inequality of the 2-norm, we conclude that the output of the first removal of truncated PCs satisfies

$$\left\|\tilde{\boldsymbol{H}}^{rpc,3} - \boldsymbol{H}^{rpc,3}\right\|_2 \leq C\tau\epsilon_1 + C\tau\epsilon_2\|(\boldsymbol{X}\boldsymbol{X}^\top - I_d)\|_2\|\boldsymbol{X}\boldsymbol{X}^\top\|_2.$$

Then we construct the next layer to eliminate $\tilde{\boldsymbol{p}}_{3,1}^{(\tau)}\tilde{\boldsymbol{p}}_{3,1}^{(\tau)\top}$ in $\boldsymbol{H}$, this can not only reduce the cost of calculation but also make the proof in the finishing up step more concise. Its parameters are given by

$$\boldsymbol{V}_1^{rpc,4} = -\boldsymbol{V}_2^{rpc,4} = \begin{bmatrix} \mathbf{0}_{5d\times 4d} & \mathbf{0} & \mathbf{0} \\ \mathbf{0}_{d\times 4d} & -I_d & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix},$$

$$\boldsymbol{Q}_1^{rpc,4} = -\boldsymbol{Q}_2^{rpc,4} = \begin{bmatrix} \mathbf{0}_{d\times 4d} & I_d & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix},$$

$$\boldsymbol{K}_1^{rpc,4} = \boldsymbol{K}_2^{rpc,4} = \begin{bmatrix} \mathbf{0}_{d\times 2d} & I_d & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}, \quad \tilde{\boldsymbol{p}}_{5,j} = 0, \quad \forall j \in [N].$$

Therefore we have

$$\boldsymbol{V}_1^{rpc,4}\boldsymbol{H}^{rpc,3} = -\boldsymbol{V}_2^{rpc,4}\boldsymbol{H}^{rpc,3} = \begin{bmatrix} \mathbf{0}_{5d\times N} \\ -\tilde{\boldsymbol{p}}_{3,1}^{(\tau)}, \mathbf{0} \\ \mathbf{0} \end{bmatrix},$$

$$\boldsymbol{Q}_1^{rpc,4}\boldsymbol{H}^{rpc,3} = -\boldsymbol{Q}_2^{rpc,4}\boldsymbol{H}^{rpc,3} = \begin{bmatrix} \tilde{\boldsymbol{p}}_{3,1}^{(\tau)}, \mathbf{0} \\ \mathbf{0} \end{bmatrix},$$

$$\boldsymbol{K}_1^{rpc,4}\boldsymbol{H}^{rpc,3} = \boldsymbol{K}_2^{rpc,4}\boldsymbol{H}^{rpc,3} = \begin{bmatrix} I_d, \mathbf{0} \\ \mathbf{0} \end{bmatrix},$$

which implies that

$$\sum_{m\in\{1,2\}} (\boldsymbol{V}_m^{rpc,4}\boldsymbol{H}^{rpc,3})\sigma\left((\boldsymbol{Q}_m^{rpc,4}\boldsymbol{H}^{rpc,3})^\top(\boldsymbol{K}_m^{rpc,4}\boldsymbol{H}^{rpc,3})\right) = \begin{bmatrix} \mathbf{0}_{5d\times N} \\ -\tilde{\boldsymbol{p}}_{3,1}^{(\tau)}\tilde{\boldsymbol{p}}_{3,1}^{(\tau)\top} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}.$$

Recall that

$$\boldsymbol{A}^{(i+1)} = \left(I_{d\times d} - \boldsymbol{v}_\tau^{(i)}\boldsymbol{v}_\tau^{(i)\top}\right)\boldsymbol{A}^{(i)}\left(I_{d\times d} - \boldsymbol{v}_\tau^{(i)}\boldsymbol{v}_\tau^{(i)\top}\right).$$

Define

$$\boldsymbol{H}^{rpc,4} := \begin{bmatrix} \boldsymbol{X} \\ (I_d - \tilde{\boldsymbol{p}}_{3,1}^{(\tau)}\tilde{\boldsymbol{p}}_{3,1}^{(\tau)\top})\boldsymbol{X}\boldsymbol{X}^\top(I_d - \tilde{\boldsymbol{p}}_{3,1}^{(\tau)}\tilde{\boldsymbol{p}}_{3,1}^{(\tau)\top}), \boldsymbol{0}_d \\ \tilde{\boldsymbol{p}}_{2,1},\ldots,\tilde{\boldsymbol{p}}_{2,N} \\ \tilde{\boldsymbol{p}}_{3,1},\ldots,\tilde{\boldsymbol{p}}_{3,N} \\ \tilde{\boldsymbol{p}}_{3,1}^{(\tau)}, \boldsymbol{0} \\ \boldsymbol{0} \\ \tilde{\boldsymbol{p}}_{6,1},\ldots,\tilde{\boldsymbol{p}}_{6,N} \\ \vdots \\ \tilde{\boldsymbol{p}}_{\ell,1},\ldots,\tilde{\boldsymbol{p}}_{\ell,N} \end{bmatrix} = \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{A}^{(2)} \\ \tilde{\boldsymbol{p}}_{2,1},\ldots,\tilde{\boldsymbol{p}}_{2,N} \\ \tilde{\boldsymbol{p}}_{3,1},\ldots,\tilde{\boldsymbol{p}}_{3,N} \\ \tilde{\boldsymbol{p}}_{3,1}^{(\tau)}, \boldsymbol{0} \\ \boldsymbol{0} \\ \tilde{\boldsymbol{p}}_{6,1},\ldots,\tilde{\boldsymbol{p}}_{6,N} \\ \vdots \\ \tilde{\boldsymbol{p}}_{\ell,1},\ldots,\tilde{\boldsymbol{p}}_{\ell,N} \end{bmatrix}.$$

Therefore, using the triangle inequality of the 2-norm, we conclude that the output of the first removal of truncated PCs satisfies

$$\left\|\tilde{\boldsymbol{H}}^{rpc,4} - \boldsymbol{H}^{rpc,4}\right\|_2 \le C\tau\epsilon_1 + C\tau\epsilon_2\|(\boldsymbol{X}\boldsymbol{X}^\top - I_d)\|_2\|\boldsymbol{X}\boldsymbol{X}^\top\|_2.$$

By this time we have replaced $\boldsymbol{A}^{(1)}$ with $\boldsymbol{A}^{(2)}$. To recover the second to the k-th truncated PC, we can repeat a similar process of Step 2 and Step 3. For any $i \in [k]$, after the i-th complete cycle, $\tilde{\boldsymbol{p}}_{3,i}^{(\tau)}$ will appear in the matrix and $\boldsymbol{A}^{(i)}$ will be replaced by $\boldsymbol{A}^{(i+1)}$. Applying the sub-additivity of the 2-norm, we can show that

$$\left\|\tilde{\boldsymbol{H}}^{rpc,4,k} - \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{A}^{(k+1)}, \boldsymbol{0} \\ \tilde{\boldsymbol{p}}_{2,1},\ldots,\tilde{\boldsymbol{p}}_{2,N} \\ \tilde{\boldsymbol{p}}_{3,1},\ldots,\tilde{\boldsymbol{p}}_{3,N} \\ \tilde{\boldsymbol{p}}_{3,1}^{(\tau)}, \boldsymbol{0} \\ \tilde{\boldsymbol{p}}_{3,2}^{(\tau)}, \boldsymbol{0} \\ \vdots \\ \tilde{\boldsymbol{p}}_{3,k}^{(\tau)}, \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix}\right\|_2 \le Ck\tau\epsilon_1 + Ck\tau\epsilon_2\|(\boldsymbol{X}\boldsymbol{X}^\top - I_d)\|_2\|\boldsymbol{X}\boldsymbol{X}^\top\|_2.$$

Step 4. Adjusting the dimensions of the final output. The finishing up phase considers constructing $\tilde{\boldsymbol{W}}_0$ and $\tilde{\boldsymbol{W}}_1$ that adjust the final output format. Our construction gives the following

$$\tilde{\boldsymbol{W}}_0 = [\boldsymbol{0}_{4d}, I_{kd}, 0], \qquad \tilde{\boldsymbol{W}}_1 = \begin{bmatrix} 1 \\ \boldsymbol{0}_{N-1} \end{bmatrix}.$$

Then we have

$$\tilde{\boldsymbol{W}}_0 \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{A}^{(k+1)}, \boldsymbol{0} \\ \tilde{\boldsymbol{p}}_{2,1},\ldots,\tilde{\boldsymbol{p}}_{2,N} \\ \tilde{\boldsymbol{p}}_{3,1},\ldots,\tilde{\boldsymbol{p}}_{3,N} \\ \tilde{\boldsymbol{p}}_{3,1}^{(\tau)}, \boldsymbol{0} \\ \tilde{\boldsymbol{p}}_{3,2}^{(\tau)}, \boldsymbol{0} \\ \vdots \\ \tilde{\boldsymbol{p}}_{3,k}^{(\tau)}, \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix} \tilde{\boldsymbol{W}}_1 = \begin{bmatrix} \tilde{\boldsymbol{p}}_{3,1}^{(\tau)} \\ \tilde{\boldsymbol{p}}_{3,2}^{(\tau)} \\ \vdots \\ \tilde{\boldsymbol{p}}_{3,k}^{(\tau)} \end{bmatrix}.$$

and therefore

$$\left\|\tilde{\boldsymbol{W}}_0\tilde{\boldsymbol{H}}^{rpc,4,k}\tilde{\boldsymbol{W}}_1 - \begin{bmatrix} \tilde{\boldsymbol{p}}_{3,1}^{(\tau)} \\ \tilde{\boldsymbol{p}}_{3,2}^{(\tau)} \\ \vdots \\ \tilde{\boldsymbol{p}}_{3,k}^{(\tau)} \end{bmatrix}\right\|_2 \le Ck\tau\epsilon_1 + Ck\tau\epsilon_2\|(\boldsymbol{X}\boldsymbol{X}^\top - I_d)\|_2\|\boldsymbol{X}\boldsymbol{X}^\top\|_2.$$

Recall that all norms defined in a finite-dimensional linear space are equivalent. Considering the spectral decomposition of $\boldsymbol{X}\boldsymbol{X}^\top$, we can obtain

$$\|\boldsymbol{X}\boldsymbol{X}^\top\|_2 \le C\Lambda, \qquad \|(\boldsymbol{X}\boldsymbol{X}^\top - I_d)\|_2 \le C\max\{|\Lambda - 1|, 1\}.$$

TPower has convergence rate of

$$\|\tilde{\boldsymbol{p}}_{3,i}^{(\tau)} - \boldsymbol{v}_i^*\|_2 \le \mu_i(r)^\tau \sqrt{1 - |\boldsymbol{v}_i^{(0),\top}\boldsymbol{v}_i^*|} + \sqrt{5}\delta(s)/(1 - \mu_i^{(2)}(r)), \quad \forall i \in [k].$$

Collecting the above pieces, and we obtain

$$\begin{aligned} d_{\mathrm{sub}}\left(\hat{\boldsymbol{V}}, \boldsymbol{V}^*\right) &\le C\tau k\epsilon_1 + C\tau k\Lambda\max\{|\Lambda - 1|, 1\}\epsilon_2 \\ &+ \sum_{i=1}^{j}\left(\mu_i(r)^\tau\sqrt{1 - |\boldsymbol{v}_i^{(0),\top}\boldsymbol{v}_i^*|} + \sqrt{5}\delta(s)/\left(1 - \mu_i^{(2)}(r)\right)\right). \end{aligned}$$

□

Theorem A.1 (Generalization and recovery guarantees)**.** *Under additional assumptions 1-3 in Appendix A.3.2 and the conditions of Theorem 4.1, there exists a universal constant $C > 0$ such that for every $\delta \in (0, 1)$, with probability at least $1 - \delta$,*

$$R(\hat{\theta}_n) \le C\Lambda\mathbb{E}_{\boldsymbol{Z}}[\varepsilon_{\mathrm{app}}^2] + Ck\mathbb{E}_{\boldsymbol{Z}}\rho(\boldsymbol{E}, s) + C\Lambda k\sqrt{\frac{P_\Theta\log(en/P_\Theta) + \log(1/\delta)}{n}} + \varepsilon_{\mathrm{opt}}.$$

If the restricted sparse eigengap condition holds with parameter κ, then on the same event,

$$\mathbb{E}_{\boldsymbol{Z}}d_{\mathrm{sub}}^2(\hat{\boldsymbol{V}}_{\hat{\theta}_n}, \boldsymbol{V}^*) \le \frac{C}{\kappa}\left\{\Lambda\mathbb{E}_{\boldsymbol{Z}}[\varepsilon_{\mathrm{app}}^2] + k\mathbb{E}_{\boldsymbol{Z}}\rho(\boldsymbol{E}, s) + \Lambda k\sqrt{\frac{P_\Theta\log(en/P_\Theta) + \log(1/\delta)}{n}} + \varepsilon_{\mathrm{opt}}\right\}.$$

A.3 Proof of Theorem A.1

A.3.1 Expression of the loss function

Let $\mathcal{P}$ be a distribution over SPCA task instances. For $\boldsymbol{Z} \sim \mathcal{P}$, write $\boldsymbol{A} = \boldsymbol{A}(\boldsymbol{Z}) = \boldsymbol{X}(\boldsymbol{Z})\boldsymbol{X}(\boldsymbol{Z})^\top = \boldsymbol{A}^* + \boldsymbol{E}$. When no confusion arises, we suppress the dependence on $\boldsymbol{Z}$. Define

$$\mathcal{S}_{r,k} := \{\boldsymbol{V} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k] \in \mathbb{R}^{d\times k} : \boldsymbol{V}^\top\boldsymbol{V} = I_k,\ \|\boldsymbol{v}_j\|_0 \le r,\ j \in [k]\}, \quad \Phi(\boldsymbol{A}) := \max_{\boldsymbol{V}\in\mathcal{S}_{r,k}} \mathrm{Tr}(\boldsymbol{V}^\top\boldsymbol{A}\boldsymbol{V}).$$

For $\boldsymbol{\theta} \in \Theta$, let $\hat{\boldsymbol{V}}_{\boldsymbol{\theta}} = \mathrm{Post}(TF_{\boldsymbol{\theta}}(\boldsymbol{H}))$ be the post-processed Transformer output. We define the unsupervised excess SPCA loss by $\ell(\boldsymbol{\theta}; \boldsymbol{Z}) := \Phi(\boldsymbol{A}) - \mathrm{Tr}(\hat{\boldsymbol{V}}_{\boldsymbol{\theta}}^\top\boldsymbol{A}\hat{\boldsymbol{V}}_{\boldsymbol{\theta}})$. Throughout this section, $C > 0$ denotes a universal constant whose value may change from line to line.

Remark 2. *While our main results are stated for the unsupervised explained-variance loss $\ell(\boldsymbol{\theta}; \boldsymbol{Z}) = \Phi(\boldsymbol{A}) - \mathrm{Tr}(\hat{\boldsymbol{V}}_{\boldsymbol{\theta}}^\top\boldsymbol{A}\hat{\boldsymbol{V}}_{\boldsymbol{\theta}})$, our proof skeleton is applicable to a broader class of loss functions. Specifically, any loss function $\ell(\boldsymbol{\theta}; \boldsymbol{Z})$ that satisfies the following properties can be substituted without altering the main structure of the bounds: (i)* boundedness*, i.e., $0 \le \ell(\boldsymbol{\theta}; \boldsymbol{Z}) \le L$ for all $\boldsymbol{\theta}$ and $\boldsymbol{Z}$, or more generally, Lipschitz continuity with respect to the empirical covariance $\boldsymbol{A}$; (ii) the associated hypothesis class $\mathcal{L}_\Theta = \{\boldsymbol{Z} \mapsto \ell(\boldsymbol{\theta}; \boldsymbol{Z}) : \boldsymbol{\theta} \in \Theta\}$ has finite pseudo-dimension P_Θ. Under these conditions, the uniform convergence argument (Lemma A.2) directly extends, and the subsequent combination with the approximation error $\varepsilon_{\mathrm{app}}$ and restricted perturbation $\rho(\boldsymbol{E}, s)$ yields population risk and subspace recovery bounds analogous to those in Theorem A.1. This includes, for instance, mean-squared-error-based losses, sparsity-penalized objectives, or orthogonality-regularized losses, as long as the boundedness and pseudo-dimension conditions are verified. Therefore, the theoretical guarantees are not limited to the explained-variance objective and can accommodate a variety of practical SPCA loss functions used in training Transformers.*

A.3.2 Additional assumptions for generalization guarantees

In addition to the key assumptions in Appendix A.1, we also introduce the following assumptions for generalization guarantees.

1. $\boldsymbol{Z}_i$ are i.i.d samples.
2. $\hat{\boldsymbol{V}}_{\boldsymbol{\theta}} \in \mathcal{S}_{r,k}$.
3. **Transformer parameter space:** Θ has bounded pseudo-dimension $\mathrm{Pdim}(\Theta)$.
4. (Optional) **Restricted Sparse Eigengap Condition**: Under the eigengap condition in Appendix A.1, the target eigenvectors of $\boldsymbol{A}^*$ are identifiable up to sign. In ordinary PCA, Davis–Kahan type bounds convert operator perturbation into subspace perturbation at a rate controlled by the inverse eigengap [4, 10]. In the sparse constrained setting, we assume the analogous restricted curvature condition over $\mathcal{S}_{r,k}$: there exists $\kappa > 0$ such that for all $\boldsymbol{V} \in \mathcal{S}_{r,k}$,

   $$\mathrm{Tr}(\boldsymbol{V}^{*\top}\boldsymbol{A}^*\boldsymbol{V}^*) - \mathrm{Tr}(\boldsymbol{V}^\top\boldsymbol{A}^*\boldsymbol{V}) \geq \kappa d_{\mathrm{sub}}^2(\boldsymbol{V}, \boldsymbol{V}^*), \qquad d_{\mathrm{sub}}(\boldsymbol{V}, \boldsymbol{V}^*) := \|\boldsymbol{V}\boldsymbol{V}^\top - \boldsymbol{V}^*\boldsymbol{V}^{*\top}\|_F.$$

   This condition is imposed directly as a restricted identifiability assumption for SPCA and plays the role of a sparse eigengap.

Lemma A.1 (Bounded unsupervised loss)**.** *If $\|\boldsymbol{A}\|_2 \leq \Lambda$ and $\hat{\boldsymbol{V}}_{\boldsymbol{\theta}} \in \mathcal{S}_{r,k}$, then $0 \leq \ell(\boldsymbol{\theta}; \boldsymbol{Z}) \leq k\Lambda$ for all $\boldsymbol{\theta} \in \Theta$ and all task instances $\boldsymbol{Z}$.*

Proof. Since $\hat{\boldsymbol{V}}_{\boldsymbol{\theta}} \in \mathcal{S}_{r,k}$ and $\Phi(\boldsymbol{A})$ is the maximum of $\mathrm{Tr}(\boldsymbol{V}^\top\boldsymbol{A}\boldsymbol{V})$ over $\mathcal{S}_{r,k}$, $\ell(\boldsymbol{\theta}; \boldsymbol{Z}) \geq 0$. For any $\boldsymbol{V} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k] \in \mathcal{S}_{r,k}$, $\mathrm{Tr}(\boldsymbol{V}^\top\boldsymbol{A}\boldsymbol{V}) = \sum_{j=1}^k \boldsymbol{v}_j^\top\boldsymbol{A}\boldsymbol{v}_j \leq \sum_{j=1}^k \|\boldsymbol{A}\|_2\|\boldsymbol{v}_j\|_2^2 \leq k\Lambda$. Hence $\Phi(\boldsymbol{A}) \leq k\Lambda$. Since $\boldsymbol{A} = \boldsymbol{X}\boldsymbol{X}^\top$ is positive semi-definite, $\mathrm{Tr}(\hat{\boldsymbol{V}}_{\boldsymbol{\theta}}^\top\boldsymbol{A}\hat{\boldsymbol{V}}_{\boldsymbol{\theta}}) \geq 0$, and therefore $\ell(\boldsymbol{\theta}; \boldsymbol{Z}) \leq k\Lambda$. □

Lemma A.2 (Uniform convergence)**.** *Under the Assumptions 1-6 in Appendix A.1 and Additional Assumptions 1-3 in Appendix A.3.2, with probability at least $1 - \delta$,*

$$R(\hat{\theta}_n) \leq \inf_{\boldsymbol{\theta}\in\Theta} R(\boldsymbol{\theta}) + C\Lambda k\sqrt{\frac{P_\Theta \log(en/P_\Theta) + \log(1/\delta)}{n}} + \varepsilon_{\mathrm{opt}}.$$

Proof. By Lemma A.1, $\mathcal{L}_\Theta$ is uniformly bounded in $[0, k\Lambda]$. The standard uniform convergence theorem for bounded real-valued classes with pseudo-dimension P_Θ [1] gives that, with probability at least $1 - \delta$,

$$\sup_{\boldsymbol{\theta}\in\Theta} |R(\boldsymbol{\theta}) - \hat{R}_n(\boldsymbol{\theta})| \leq C\Lambda k\sqrt{\frac{P_\Theta \log(en/P_\Theta) + \log(1/\delta)}{n}} =: \varepsilon_{\mathrm{gen}}.$$

On this event,

$$R(\hat{\theta}_n) \leq \hat{R}_n(\hat{\theta}_n) + \varepsilon_{\mathrm{gen}} \leq \inf_{\boldsymbol{\theta}\in\Theta} \hat{R}_n(\boldsymbol{\theta}) + \varepsilon_{\mathrm{opt}} + \varepsilon_{\mathrm{gen}} \leq \inf_{\boldsymbol{\theta}\in\Theta} R(\boldsymbol{\theta}) + 2\varepsilon_{\mathrm{gen}} + \varepsilon_{\mathrm{opt}}.$$

Absorbing the factor 2 into C proves the lemma. □

Lemma A.3 (Restricted perturbation)**.** *For any symmetric $\boldsymbol{E}$, any $\boldsymbol{V} \in \mathcal{S}_{r,k}$, and any $s \geq r$,*

$$|\mathrm{Tr}(\boldsymbol{V}^\top\boldsymbol{E}\boldsymbol{V})| \leq k\rho(\boldsymbol{E}, s), \qquad |\Phi(\boldsymbol{A}^* + \boldsymbol{E}) - \Phi(\boldsymbol{A}^*)| \leq k\rho(\boldsymbol{E}, s).$$

Proof. The first inequality follows from $|\mathrm{Tr}(\boldsymbol{V}^\top\boldsymbol{E}\boldsymbol{V})| \leq \sum_{j=1}^k |\boldsymbol{v}_j^\top\boldsymbol{E}\boldsymbol{v}_j| \leq k\rho(\boldsymbol{E}, s)$, since each $\boldsymbol{v}_j$ is unit norm and r-sparse. For the second inequality, let $\boldsymbol{A} = \boldsymbol{A}^* + \boldsymbol{E}$, $\boldsymbol{V}_A \in \arg\max_{\boldsymbol{V}\in\mathcal{S}_{r,k}} \mathrm{Tr}(\boldsymbol{V}^\top\boldsymbol{A}\boldsymbol{V})$, and $\boldsymbol{V}_{A^*} \in \arg\max_{\boldsymbol{V}\in\mathcal{S}_{r,k}} \mathrm{Tr}(\boldsymbol{V}^\top\boldsymbol{A}^*\boldsymbol{V})$. Since $\boldsymbol{V}_{A^*}$ maximizes the $\boldsymbol{A}^*$ objective, $\Phi(\boldsymbol{A}) - \Phi(\boldsymbol{A}^*) \leq \mathrm{Tr}(\boldsymbol{V}_A^\top\boldsymbol{A}\boldsymbol{V}_A) - \mathrm{Tr}(\boldsymbol{V}_A^\top\boldsymbol{A}^*\boldsymbol{V}_A) = \mathrm{Tr}(\boldsymbol{V}_A^\top\boldsymbol{E}\boldsymbol{V}_A) \leq k\rho(\boldsymbol{E}, s)$. Similarly, since $\boldsymbol{V}_A$ maximizes the $\boldsymbol{A}$ objective, $\Phi(\boldsymbol{A}^*) - \Phi(\boldsymbol{A}) \leq \mathrm{Tr}(\boldsymbol{V}_{A^*}^\top\boldsymbol{A}^*\boldsymbol{V}_{A^*}) - \mathrm{Tr}(\boldsymbol{V}_{A^*}^\top\boldsymbol{A}\boldsymbol{V}_{A^*}) = -\mathrm{Tr}(\boldsymbol{V}_{A^*}^\top\boldsymbol{E}\boldsymbol{V}_{A^*}) \leq k\rho(\boldsymbol{E}, s)$. Combining the two one-sided bounds proves the result. □

Lemma A.4 (Oracle risk)**.** *Under the conditions of Theorem 4.1, let $\boldsymbol{\theta}_{\mathrm{alg}} \in \Theta$ be the constructive Transformer parameter. If $\|\boldsymbol{A}^*\|_2 \leq \Lambda$ almost surely, then*

$$R(\boldsymbol{\theta}_{\mathrm{alg}}) \leq C\Lambda\mathbb{E}_{\boldsymbol{Z}}[\varepsilon_{\mathrm{app}}^2] + Ck\mathbb{E}_{\boldsymbol{Z}}\rho(\boldsymbol{E}, s).$$

Proof. Fix a task instance and write $\hat{\boldsymbol{V}} = \hat{\boldsymbol{V}}_{\boldsymbol{\theta}_{\mathrm{alg}}}$. We decompose

$$\ell(\boldsymbol{\theta}_{\mathrm{alg}}; \boldsymbol{Z}) = \Phi(\boldsymbol{A}) - \mathrm{Tr}(\hat{\boldsymbol{V}}^\top \boldsymbol{A} \hat{\boldsymbol{V}}) = [\Phi(\boldsymbol{A}) - \Phi(\boldsymbol{A}^*)] + [\Phi(\boldsymbol{A}^*) - \mathrm{Tr}(\hat{\boldsymbol{V}}^\top \boldsymbol{A}^* \hat{\boldsymbol{V}})] + [\mathrm{Tr}(\hat{\boldsymbol{V}}^\top \boldsymbol{A}^* \hat{\boldsymbol{V}}) - \mathrm{Tr}(\hat{\boldsymbol{V}}^\top \boldsymbol{A} \hat{\boldsymbol{V}})].$$

By Lemma A.3, the first and third terms are bounded by $Ck\rho(\boldsymbol{E}, s)$. For the middle term, by the definition of $\Phi(\boldsymbol{A}^*)$ and the standard eigenspace objective bound,

$$\Phi(\boldsymbol{A}^*) - \mathrm{Tr}(\hat{\boldsymbol{V}}^\top \boldsymbol{A}^* \hat{\boldsymbol{V}}) \le C\|\boldsymbol{A}^*\|_2 d_{\mathrm{sub}}^2(\hat{\boldsymbol{V}}, \boldsymbol{V}^*) \le C\Lambda d_{\mathrm{sub}}^2(\hat{\boldsymbol{V}}, \boldsymbol{V}^*).$$

By Theorem 4.1, $d_{\mathrm{sub}}(\hat{\boldsymbol{V}}, \boldsymbol{V}^*) \le C\varepsilon_{\mathrm{app}}$, where $\varepsilon_{\mathrm{app}}$ denotes the full output error bound in Theorem 4.1. Therefore

$$\ell(\boldsymbol{\theta}_{\mathrm{alg}}; \boldsymbol{Z}) \le C\Lambda\varepsilon_{\mathrm{app}}^2 + Ck\rho(\boldsymbol{E}, s).$$

Taking expectation over $\boldsymbol{Z} \sim \mathcal{P}$ proves the lemma. □

Proof of Theorem A.1. By Lemma A.2, with probability at least $1 - \delta$,

$$R(\hat{\theta}_n) \le \inf_{\boldsymbol{\theta} \in \Theta} R(\boldsymbol{\theta}) + C\Lambda k \sqrt{\frac{P_\Theta \log(en/P_\Theta) + \log(1/\delta)}{n}} + \varepsilon_{\mathrm{opt}}.$$

Since $\boldsymbol{\theta}_{\mathrm{alg}} \in \Theta$, $\inf_{\boldsymbol{\theta} \in \Theta} R(\boldsymbol{\theta}) \le R(\boldsymbol{\theta}_{\mathrm{alg}})$. Applying Lemma A.4 proves the risk bound.

It remains to prove the recovery bound. Let $\boldsymbol{V} = \hat{\boldsymbol{V}}_{\hat{\theta}_n}$. By the restricted sparse eigengap condition,

$$\kappa d_{\mathrm{sub}}^2(\boldsymbol{V}, \boldsymbol{V}^*) \le \mathrm{Tr}(\boldsymbol{V}^{*\top} \boldsymbol{A}^* \boldsymbol{V}^*) - \mathrm{Tr}(\boldsymbol{V}^\top \boldsymbol{A}^* \boldsymbol{V}) = \Phi(\boldsymbol{A}^*) - \mathrm{Tr}(\boldsymbol{V}^\top \boldsymbol{A}^* \boldsymbol{V}).$$

Adding and subtracting empirical quantities gives

$$\Phi(\boldsymbol{A}^*) - \mathrm{Tr}(\boldsymbol{V}^\top \boldsymbol{A}^* \boldsymbol{V}) = [\Phi(\boldsymbol{A}^*) - \Phi(\boldsymbol{A})] + \ell(\hat{\theta}_n; \boldsymbol{Z}) + [\mathrm{Tr}(\boldsymbol{V}^\top \boldsymbol{A} \boldsymbol{V}) - \mathrm{Tr}(\boldsymbol{V}^\top \boldsymbol{A}^* \boldsymbol{V})].$$

By Lemma A.3, the first and third terms are bounded by $Ck\rho(\boldsymbol{E}, s)$. Hence $\kappa d_{\mathrm{sub}}^2(\boldsymbol{V}, \boldsymbol{V}^*) \le \ell(\hat{\theta}_n; \boldsymbol{Z}) + Ck\rho(\boldsymbol{E}, s)$. Taking expectation over $\boldsymbol{Z}$ yields

$$\kappa \mathbb{E}_{\boldsymbol{Z}} d_{\mathrm{sub}}^2(\hat{\boldsymbol{V}}_{\hat{\theta}_n}, \boldsymbol{V}^*) \le R(\hat{\theta}_n) + Ck\mathbb{E}_{\boldsymbol{Z}} \rho(\boldsymbol{E}, s).$$

Substituting the risk bound already proved gives the claimed recovery bound. □

A.4 Proof of Corollary 4.1.1

Proof of Corollary 4.1.1. Write $\varepsilon_{\mathrm{app}}(\varepsilon) \le \varepsilon_{\mathrm{tpower}} + C_{\mathrm{app}}\varepsilon$, where only the second term is reduced by increasing the Transformer complexity. Then $\varepsilon_{\mathrm{app}}(\varepsilon)^2 \le C\varepsilon_{\mathrm{tpower}}^2 + C\varepsilon^2$. The leading tunable approximation and generalization terms in Theorem A.1 are therefore

$$\varepsilon^2 + \sqrt{\frac{P_\Theta(\varepsilon)}{n}}.$$

Under $P_\Theta(\varepsilon) \lesssim \varepsilon^{-q}$, these terms are bounded by $\varepsilon^2 + \varepsilon^{-q/2} n^{-1/2}$ up to logarithmic factors. Balancing the two terms gives $\varepsilon^2 \asymp \varepsilon^{-q/2} n^{-1/2}$, or equivalently $\varepsilon^{2+q/2} \asymp n^{-1/2}$. Thus $\varepsilon \asymp n^{-1/(q+4)}$, and substituting this into ε^2 gives $n^{-2/(q+4)}$. The term $\varepsilon_{\mathrm{tpower}}^2$ is carried through unchanged.

The exponent q describes how the statistical complexity of the Transformer class grows as the tunable approximation precision is improved. In the current construction, the normalization approximation uses a multi-head ReLU attention approximation whose number of heads scales at least as ϵ_2^{-2} up to logarithmic factors. Thus, if the pseudo-dimension grows at least linearly with the number of heads, the normalization part alone suggests $q \ge 2$. The truncation block has width polynomial in the inverse approximation precision; if this polynomial dependence is ϵ_1^{-a} and it dominates the head complexity, then q should be taken at least as large as a. More generally, q should be interpreted as the architecture-dependent exponent satisfying $P_\Theta(\varepsilon) \lesssim \varepsilon^{-q}$, so the corollary states the rate in terms of a general q rather than asserting a universal numerical exponent. □

A.5 Lemma A.5

This is Lemma B.2 from [5], we restate it here.

Lemma A.5 (Approximation of norm by sum of ReLU activations by Transformer networks)**.** *Assume that there exists a constant C with $\|\boldsymbol{v}\|_2 \leq C$. There exists a multi-head ReLU attention layer with number of heads $M < \left(\frac{\overline{R}}{\underline{R}}\right)^d \frac{C(d)}{\epsilon^2} \log(1 + C/\epsilon)$ such that there exists $\{\boldsymbol{a}_m\}_{m\in[M]} \subset \mathbb{S}^{d-1}$ and $\{c_m\}_{m\in[M]} \subset \mathbb{R}$ where for all $\boldsymbol{v}$ with $\overline{R} \geq \|\boldsymbol{v}\|_2 \geq \underline{R}$, we have*

$$\left| \sum_{m=1}^{M} c_m \sigma(\boldsymbol{a}_m^\top \boldsymbol{v}) - \frac{1}{\|\boldsymbol{v}\|_2} + 1 \right| \leq \epsilon.$$

Remark 3. *Using the definition of the truncation operation, we have for any unit vector $\boldsymbol{w} \in \mathbb{R}^d$, $\frac{r}{d}\|\boldsymbol{X}\boldsymbol{X}^\top \boldsymbol{w}\|_2 \leq \|\operatorname{Trunc}_r(\boldsymbol{X}\boldsymbol{X}^\top \boldsymbol{w})\|_2 \leq \|\boldsymbol{X}\boldsymbol{X}^\top \boldsymbol{w}\|_2 \leq \|\boldsymbol{X}\boldsymbol{X}^\top\|_2 \leq \Lambda$. Therefore, when we apply this lemma in our proof, we can always take $\frac{\overline{R}}{\underline{R}} = \frac{d}{r}$ and Λ as the constant C in this lemma.*

A.6 Proof of Lemma A.6

Lemma A.6 (Stable top-r truncation by a fixed Transformer)**.** *Fix $d \geq 1$, $1 \leq r < d$, $B > 0$, and $\Delta_{\mathrm{top}} > 0$. Let*

$$\mathcal{K}^{(r)}_{B,\Delta_{\mathrm{top}}} := \left\{ \boldsymbol{v} \in [-B, B]^d : |\boldsymbol{v}|_{(r)} - |\boldsymbol{v}|_{(r+1)} \geq \Delta_{\mathrm{top}} \right\}.$$

Then there exists a ReLU Transformer block TF_Θ whose parameters depend only on $(d, r, B, \Delta_{\mathrm{top}})$. Moreover, if an approximate multiplier is used, then for every $\varepsilon > 0$ there exists such a block with width polynomial in $(d, B, \Delta_{\mathrm{top}}^{-1}, \varepsilon^{-1})$ satisfying

$$\sup_{\boldsymbol{v}\in\mathcal{K}^{(r)}_{B,\Delta_{\mathrm{top}}}} \|\mathrm{TF}_\Theta(\boldsymbol{v}) - \operatorname{Trunc}_r(\boldsymbol{v})\|_\infty \leq \varepsilon.$$

Proof. We construct an L-layer ReLU Transformer with

$$L = S_d + 3,$$

where S_d is the depth of a fixed sorting network on d inputs (for example, one may take $S_d = O((\log d)^2)$ using a bitonic sorting network). The parameters depend only on $(d, r, B, \Delta_{\mathrm{top}})$.

Input embedding. We identify the input vector $\boldsymbol{v} = (v_1, \ldots, v_d) \in \mathbb{R}^d$ with a sequence

$$\boldsymbol{H}^{(0)}(\boldsymbol{v}) \in \mathbb{R}^{D\times d},$$

with $N = d$ tokens, where the ith token stores:

- the raw value v_i in one designated coordinate;
- a one-hot positional code $e_i \in \mathbb{R}^d$ in a designated block;
- several auxiliary work coordinates initialized at zero.

The ambient token dimension D is chosen large enough, polynomial in d, so that all intermediate quantities can be stored in separate coordinates.

Layer 1: exact absolute values. In the first Transformer layer, the attention part is taken to be the identity (all heads zero), and the MLP part acts tokenwise. Using the ReLU identity

$$|x| = \sigma(x) + \sigma(-x),$$

the MLP writes

$$a_i := |v_i|$$

into a designated work coordinate of the ith token, while keeping the raw value $\boldsymbol{v}_i$ and the positional code unchanged. Thus after Layer 1, each token stores (v_i, a_i, e_i).

Layers $2, \dots, S_d + 1$: exact sorting of $(a_1, \dots, a_d)$. Fix a sorting network on d inputs with depth S_d. For each stage $s \in [S_d]$, let $\mathscr{P}_s$ denote the collection of disjoint comparator pairs used at that stage. We claim that one Transformer layer with attention heads no more than d can realize one entire stage.

Indeed, because the positional codes are fixed and the pairing pattern in $\mathscr{P}_s$ depends only on d, the attention block can be chosen so that, for every comparator pair $(i, j) \in \mathscr{P}_s$, the current values stored at tokens i and j are routed into auxiliary coordinates of both participating tokens. Since all pairs in one stage are disjoint, this routing can be performed simultaneously in one attention layer with a polynomial number of heads.

The subsequent tokenwise MLP applies the exact ReLU formulas

$$\max\{x, y\} = \sigma(x - y) + y, \qquad \min\{x, y\} = x - \sigma(x - y),$$

and updates the designated sorted-value coordinate according to whether the token is meant to hold the larger or the smaller output of that comparator. All untouched tokens simply pass through by the residual connection.

Therefore, one Transformer layer implements one sorting stage exactly. After S_d such layers, the tokens store the absolute values in sorted order:

$$a_{(1)} \geq \cdots \geq a_{(d)}, \qquad a_{(k)} = |\boldsymbol{v}|_{(k)}.$$

In particular, token r stores $a_{(r)}$ and token $r + 1$ stores $a_{(r+1)}$.

Layer $S_d + 2$: broadcast the threshold and construct the mask. In the next layer, the attention block uses the positional codes to broadcast the values at tokens r and $r + 1$ to every token. Thus each token receives

$$a_{(r)}, \qquad a_{(r+1)}.$$

The tokenwise MLP then computes

$$\tau := \frac{a_{(r)} + a_{(r+1)}}{2}$$

and

$$m_i := \eta(a_i - \tau),$$

where η is the fixed piecewise linear map

$$\eta(t) := \begin{cases} 0, & t \leq -\Delta_{\mathrm{top}}/4, \\ \dfrac{2}{\Delta_{\mathrm{top}}}\, t + \dfrac{1}{2}, & |t| < \Delta_{\mathrm{top}}/4, \\ 1, & t \geq \Delta_{\mathrm{top}}/4. \end{cases}$$

Since η is piecewise linear, it is representable exactly by a ReLU MLP.

Now, because $\boldsymbol{v} \in \mathcal{K}^{(r)}_{B,\Delta_{\mathrm{top}}}$, we have

$$a_{(r)} - a_{(r+1)} \geq \Delta_{\mathrm{top}},$$

hence

$$a_i - \tau \geq \frac{\Delta_{\mathrm{top}}}{2} \quad \text{for } i \in \mathrm{supp}(\mathrm{Trunc}_r(\boldsymbol{v})),$$

while

$$a_i - \tau \leq -\frac{\Delta_{\mathrm{top}}}{2} \quad \text{for } i \notin \mathrm{supp}(\mathrm{Trunc}_r(\boldsymbol{v})).$$

Therefore

$$m_i = \begin{cases} 1, & i \in \mathrm{supp}(\mathrm{Trunc}_r(\boldsymbol{v})), \\ 0, & i \notin \mathrm{supp}(\mathrm{Trunc}_r(\boldsymbol{v})). \end{cases}$$

Thus this layer constructs the exact top-r support mask.

Layer $S_d + 3$: exact gated output. In the final layer, the attention part is again taken to be the identity, and the tokenwise MLP applies the exact ReLU gate

$$T(v, m) := \sigma(v - C(1 - m)) - \sigma(-v - C(1 - m)),$$

where $C > B$ is any fixed constant.

Since $|v_i| \le B < C$, we have:

$$T(v_i, 1) = \sigma(v_i) - \sigma(-v_i) = v_i, \qquad T(v_i, 0) = 0.$$

Hence the output coordinate of the ith token is exactly

$$T(v_i, m_i) = \begin{cases} v_i, & i \in \mathrm{supp}(\mathrm{Trunc}_r(\boldsymbol{v})), \\ 0, & i \notin \mathrm{supp}(\mathrm{Trunc}_r(\boldsymbol{v})). \end{cases}$$

That is, the designated output coordinate across the d tokens is precisely $\mathrm{Trunc}_r(\boldsymbol{v})$.

Finally, choose the output projections $\tilde{\boldsymbol{W}}_0$ and $\tilde{\boldsymbol{W}}_1$ so that they extract this designated coordinate from all d tokens and identify it with a vector in $\mathbb{R}^d$. Therefore

$$\mathrm{TF}_\Theta(\boldsymbol{v}) = \mathrm{Trunc}_r(\boldsymbol{v}), \qquad \forall \boldsymbol{v} \in \mathcal{K}^{(r)}_{B,\Delta_{\mathrm{top}}}.$$

Approximate multiplier version. If, in the last layer, the exact gate $T(v, m)$ is replaced by an approximate multiplier subnetwork, then the preceding layers still construct the exact support mask $(m_i)_{i=1}^d$. Since the multiplication map $(x, m) \mapsto xm$ is continuous on the compact set $[-B, B] \times [0, 1]$, a ReLU MLP can approximate it uniformly to arbitrary precision. Hence, for every $\varepsilon > 0$, there exists a Transformer with the same number of layers $L = S_d + 3$, width polynomial in $(d, B, \Delta_{\mathrm{top}}^{-1}, \varepsilon^{-1})$, and polynomially many heads, such that

$$\sup_{\boldsymbol{v} \in \mathcal{K}^{(r)}_{B,\Delta_{\mathrm{top}}}} \|\mathrm{TF}_\Theta(\boldsymbol{v}) - \mathrm{Trunc}_r(\boldsymbol{v})\|_\infty \le \varepsilon.$$

This proves the lemma. □

Remark 4. *The restriction to $K^{(r)}_{B,\Delta_{\mathrm{top}}}$ is necessary for a uniform approximation statement. Indeed, Trunc_r is discontinuous at points where $|\boldsymbol{v}|_{(r)} = |\boldsymbol{v}|_{(r+1)}$, whereas ReLU Transformers define continuous maps. Hence no continuous Transformer can uniformly approximate Trunc_r on the whole cube $[-B, B]^d$.*

For random inputs with a continuous distribution, exact ties occur with probability zero. However, the absence of exact ties does not by itself imply a uniform separation gap. Our theorem therefore explicitly assumes the stronger condition $|\boldsymbol{v}|_{(r)} - |\boldsymbol{v}|_{(r+1)} \ge \Delta_{\mathrm{top}}$ along the relevant TPower iterates, which guarantees stability of the selected support and permits a fixed, input-independent Transformer construction.

B Empirical Appendix

B.1 Experimental Setup

Our experiments start by first constructing the training set and testing set. Then, we train a Transformer model by modifying the GPT-2 architecture [9] into an encoder-based model with a ReLU activation function. We use unsupervised learning objectives such as explained variance, with a orthogonal loss or sparsity penalty as the tuning parameters. Then we evaluate the model on the testing set and compare its performance with baselines. Each experiment repeats this procedure with 5 model random seeds (42, 43, 44, 45, 46). We record the average performance of Transformers across different seeds in the tasks of predicting the top k sparse PCs. All experiments were conducted on a server equipped with 4 NVIDIA A100 GPUs (80GB PCIe). The NVIDIA driver version is 570.195.03, and the CUDA version is 12.8. We construct a training pipeline using the PyTorch library [7] and generate data using the Scikit-learn library [8].

Unless otherwise mentioned, the default parameters of the Transformers are Training sample size=1024; Testing sample size=512; Layers=4; Heads=4; Embeddings = 64.

B.2 Synthetic Dataset

We follow the synthetic data construction in Section 5.2 of [12]. The synthetic dataset contains three latent factors V1, V2, V3, with $V_3 = -0.3V_1 + 0.925V_2 + \varepsilon$, $\varepsilon \sim N(0, 1)$. Each factor is associated with disjoint sets of observable variables: V1 with X1–X4, V2 with X5–X8, and V3 with X9–X10. Additive Gaussian noise with standard deviation 1.0 is applied.

The model is trained with AdamW using learning rate 5×10^{-4}, batch size 128, and 20 epochs. We use mixed-precision training on GPU when available and clip the gradient norm at 1.0. In the default setting, each experiment uses 1024 training instances and 512 test instances. Since sparse PCA components are identifiable only up to sign and permutation, predicted components are matched to the ground-truth components by maximizing the average absolute cosine similarity.

B.2.1 Ablation Experiments

Figure 6 reports per-component metrics on the synthetic SPCA benchmark. Adjusting network depth, number of heads, and embedding dimension reveals that PC1 and PC2 are robust, while PC3 exhibits higher sensitivity to architectural changes, consistent with its tiny explained variance. Performance peaks around depth=4, heads=4, and embedding width=64, guiding our main experimental configuration.

Figure 7 reports the effect of training sample size. Larger training sets generally improve support recovery and reduce ℓ_2 error, especially up to the intermediate regime, which agrees with the generalization term in Theorem A.1. The remaining non-monotonicity is expected because the total recovery error also contains non-vanishing noise, finite-iteration approximation error, and optimization effects.

Overall, these analyses are qualitatively consistent with the capacity–generalization tradeoffs suggested by Theorem 4.2 and support our architecture choice for the synthetic SPCA experiments. They should not be interpreted as a direct empirical verification of the full theoretical bound, since optimization effects and finite-sample noise remain present.

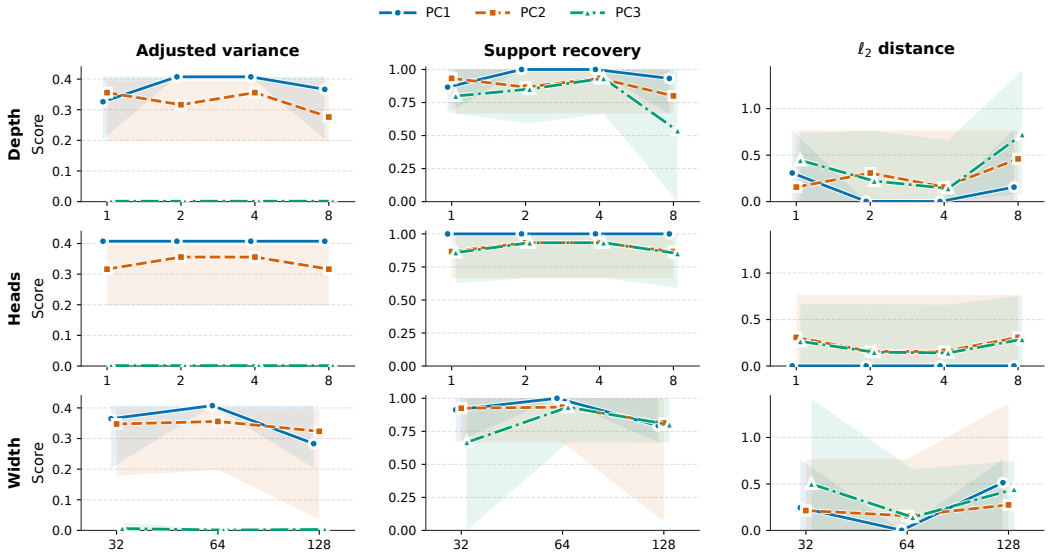

Figure 6: Architecture ablation on the synthetic SPCA benchmark. Adjusted variance, support F1, and ℓ_2 distance to true loadings are shown per component (PC1–PC3) across varying network depth, number of heads, and embedding width. Solid lines indicate mean over 5 model seeds; shaded regions denote min–max range. Experimental setup: $D = 10, N = 200$, fixed training/testing data seed. PC1 and PC2 are robust to architectural changes, while PC3 is more sensitive. Optimal performance occurs around depth=4, heads=4, width=64.

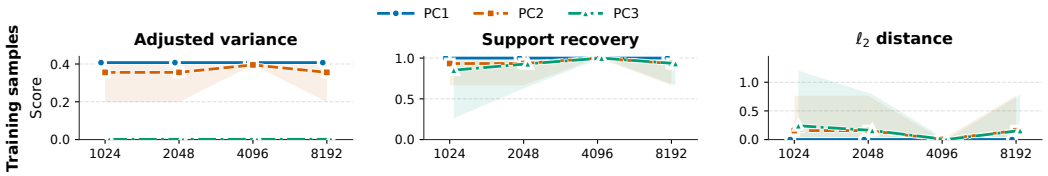

Figure 7: Panels show per-component metrics (PC1–PC3) for adjusted variance (left), support F1 (middle), and ℓ_2 distance (right) across increasing training sample sizes. Increasing the training size generally stabilizes support recovery and reduces recovery error up to the intermediate regime, while remaining fluctuations reflect optimization variability and the difficulty of weak-component recovery.

B.2.2 Computational Cost

Our experiments are conducted using five random seeds (42–46). To reduce the influence of server-side fluctuations, we additionally repeat the entire benchmarking procedure five times while alternating the execution order of Transformer and baselines. We vary one factor at a time around the default Transformer architecture in our synthetic settings.

Inference time and Comparison with Baselines To obtain stable measurements, both Transformer and baseline inference times are measured after 5 warm-up runs and then averaged over 50 repeated executions. The reported numbers are averaged over all seed-repeat combinations. The reported experiments use a single GPU for the Transformer and CPU implementations for the classical baselines. Figure 8 compares the inference time of Transformer variants with classical SPCA baselines. For the Transformer, inference is amortized: after training, the model predicts all $k = 3$ sparse principal components in a single batched forward pass. Thus, the reported time is the total test-set forward time for all components, rather than a per-component sequential extraction time.

For TPower, we include several initialization variants to make the comparison transparent. The "dense" variant initializes TPower from the leading dense PCA direction, while "diag" uses diagonal or coordinate-based initializers. The "combo" variant is the strongest initialization family and combines dense PCA, thresholded dense PCA, diagonal, greedy, and SparsePCA-based initializers. The notation R denotes R additional random restarts for each extracted component. "TPower (Combo R100)" is the TPower setup we use to plot the performance of TPower in Fig. 2. For each initialization, TPower runs the truncated power iterations and selects the solution with the largest sparse Rayleigh quotient. Because TPower is a non-convex iterative method, more initialization candidates and random restarts generally improve robustness but increase runtime substantially.

For the Elastic Net baseline, we follow the Zou-style SPCA formulation and fix the ridge regularization parameter to 10^{-3}. The sparsity parameter is selected from the grid $\{10^{-4}, 3 \times 10^{-4}, 10^{-3}, 3 \times 10^{-3}, 10^{-2}, 3 \times 10^{-2}, 10^{-1}, 3 \times 10^{-1}, 1\}$, and the best solution is chosen according to the objective value. We use the same parameters to plot the performance of Elastic Net in Fig. 2.

The results show that Transformer inference remains fast across model sizes. Changing the number of training instances does not directly affect inference complexity once the model architecture is fixed. Increasing the depth L has the clearest effect among Transformer variants, since more attention–MLP blocks must be evaluated. The effects of the number of heads H and embedding dimension E are moderate in this low-dimensional setting.

Compared with iterative TPower variants, the Transformer benefits significantly from amortized inference. After training, it predicts all components with a single batched GPU forward pass, whereas TPower performs CPU-based per-instance iterative optimization with multiple initializations and random restarts.

Training time and GPU memory usage We obtain the training time and peak allocated GPU memory usage of representative Transformer architectures. Fig. 9(a) shows that training time increases clearly with the number of training instances and model depth, while the effect of model width and the number of heads is mild in this setting. The standard deviation is positively related to the mean value of the training time. Fig. 9(b) shows that the peak allocated GPU memory usage increases clearly with model width and model depth, while training sample size and the number of heads has

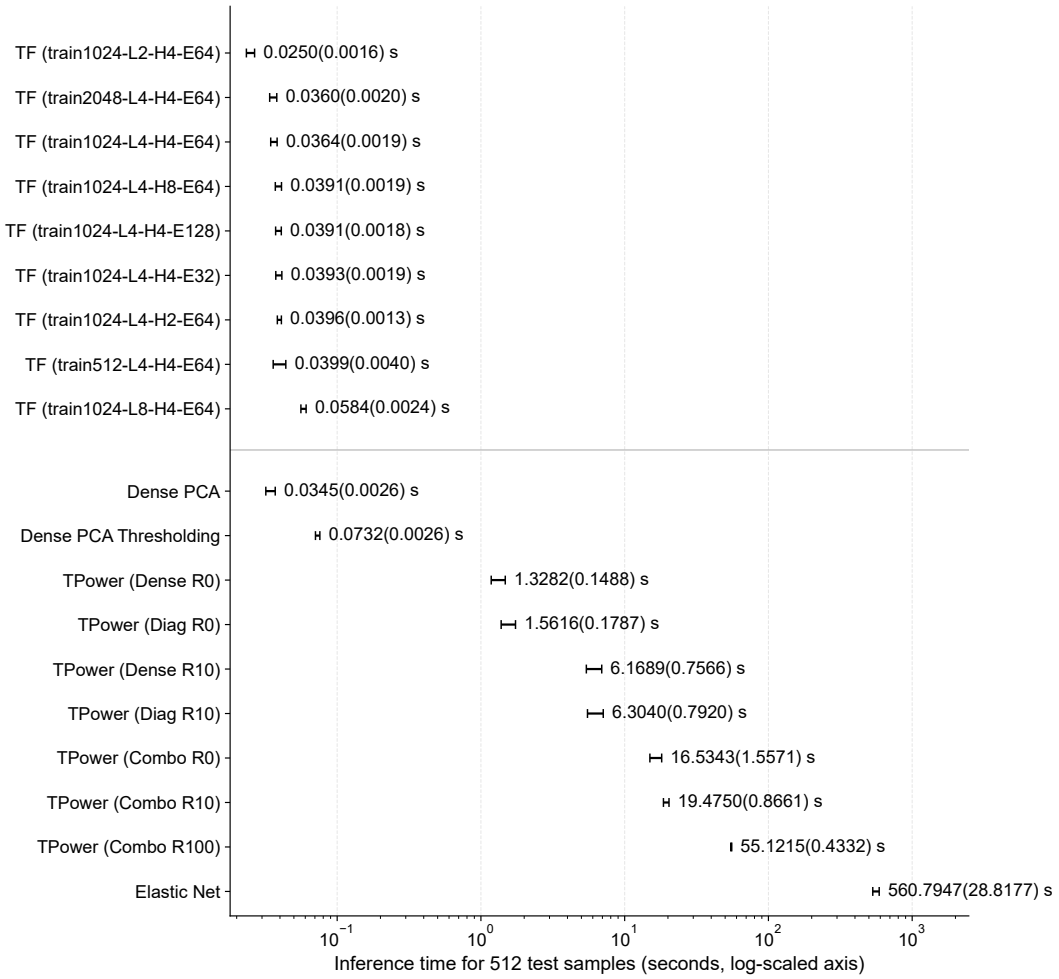

Figure 8: Inference time comparison. For Transformer variants, labels are of the form "TF (train$n$-Lℓ-Hh-Ee)", where n is the number of training instances, L is the number of Transformer layers, H is the number of attention heads, and E is the embedding dimension. For TPower variants, labels are of the form "TPower (initialization RR)", where the initialization field denotes the initialization strategy and R denotes the number of random restarts. The reported Transformer time is the synchronized GPU batched forward time for predicting all $k = 3$ sparse principal components on the test set of 512 samples. The reported baseline time measures the clean CPU algorithm time of the same 512 samples, excluding data loading, data-format conversion, post-processing of the obtained PCs, metric computation, and file writing. The x-axis is shown on a logarithmic scale for visualization, while the values next to the bars report the original untransformed runtime in seconds in the form of mean(std).

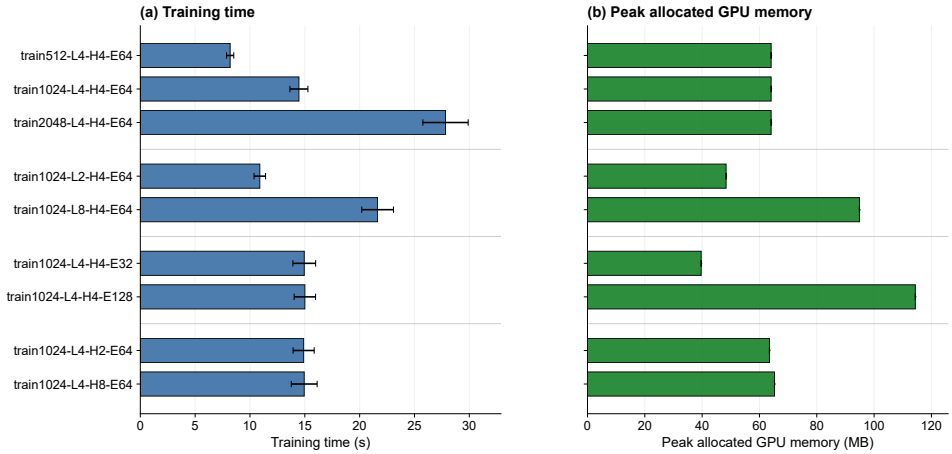

Figure 9: Training time (a) and peak allocated GPU memory usage (b) of different Transformer architectures. Labels are of the form "train$n$-Lℓ-Hh-Ee", where n is the training sample size, L is the number of Transformer layers, H is the number of attention heads, and E is the embedding dimension. Training time increases clearly with training sample size and model depth, while model width and the number of heads has little influence. For the peak allocated GPU memory, memory usage increases clearly with model width and model depth, while training sample size and the number of heads has little influence. Error bars denote standard deviation over repeated runs.

little influence. The standard deviation of peak GPU memory is zero because memory allocation is fully determined by the model architecture, batch size, and input dimensions, which remain fixed across repeated runs.

Across all configurations, the Transformer remains computationally lightweight. Even the largest architecture considered in our experiments can be trained within a few tens of seconds on a single modern GPU while requiring only tens of megabytes of GPU memory. We clarify that Transformer SPCA is not claimed to be uniformly faster when training cost is included; rather, its potential computational advantage lies in amortized and batched inference after training.

B.3 Pitprops Dataset

The pitprops dataset [6], which consists of 180 observations with 13 measured variables, has been a standard benchmark for testing SPCA algorithms. Since real-world datasets do not provide ground-truth sparse PCs, we compare the recovered loadings with those reported by classical SPCA methods. The similarity of the selected variables suggests that the Transformer identifies sparse structures consistent with existing SPCA baselines. Following previous studies, we also consider the first six sparse PCs of the data. When extracting multiple PCs, we can adopt the sequential training trick. First, we train a Transformer to extract PC1 of the input matrix. Then, after performing iterative deflation as in TPower [11], we apply the trained Transformer model to extract PC'1 of the matrix after deflation, and take PC'1 as PC2 of the input matrix. The same process is repeated until k PCs are extracted. The benefit of sequential training is that the dimension of the desired output of the Transformer model only needs to be 1 instead of k. This significantly reduces the GPU memory usage, and suggests that Transformers learn how to exploit the dominant features despite the change of the data matrix. Additionally, we don't need to train a new Transformer even if we need a different choice of k. This trick is helpful on high-dimensional datasets.

Our results across 5 seeds are recorded in Table 2. Note that given the continuous nature of the Transformer network, we have to apply hard thresholding to the Transformer output to obtain the desired sparse vectors. Comparing the results with TPower [11] and Elastic Net [12] (see Table 3 and Table 4), we claim that Transformer has successfully captured the key structures of the Pitprops dataset. Furthermore, the close alignment between the TPower result and the Transformer sequential

training result suggests that Transformers have learned to extract dominant features despite the change of data matrix.

We create the training set by augmenting the empirical covariance matrix of the data. Specifically, we generate 1,024 independent training samples by adding symmetric Gaussian noise to the base covariance matrix, thereby creating a diverse set of noisy covariance structures that represent various perturbations of the underlying signals.

B.3.1 Loading prediction comparison with baselines

Table 2: Transformer SPCA loadings with hard threshold 7-2-1-1-1-1.

PCs	x_1 topd	x_2 length	x_3 moist	x_4 testsg	x_5 ovensg	x_6 ringt	x_7 ringb	x_8 bowm	x_9 bowd	x_{10} whorls	x_{11} clear	x_{12} knots	x_{13} diaknot
PC1	0.420	0.422				0.295	0.415	0.305	0.371	0.393			
PC2			0.765	0.644									
PC3					1.000								
PC4											1.000		
PC5												1.000	
PC6													1.000
PC1(std)	0.003	0.001				0.001	0.003	0.003	0.003				
PC2(std)			0.003	0.004									

The table reports the mean and std of PC1-6 over 5 seeds (42-46); the 0 std of PC3–PC6 are omitted for simplicity.

Table 3: TPower loadings with cardinality setting 7-2-1-1-1-1.

PCs	x_1 topd	x_2 length	x_3 moist	x_4 testsg	x_5 ovensg	x_6 ringt	x_7 ringb	x_8 bowm	x_9 bowd	x_{10} whorls	x_{11} clear	x_{12} knots	x_{13} diaknot
PC1	0.424	0.430				0.268	0.403	0.313	0.379	0.399			
PC2			0.707	0.707									
PC3					1.000								
PC4											1.000		
PC5												1.000	
PC6													1.000

Table 4: Elastic Net loadings.

PCs	x_1 topd	x_2 length	x_3 moist	x_4 testsg	x_5 ovensg	x_6 ringt	x_7 ringb	x_8 bowm	x_9 bowd	x_{10} whorls	x_{11} clear	x_{12} knots	x_{13} diaknot
PC1	-0.477	-0.476			0.177		-0.250	-0.344	-0.416	-0.400			
PC2			0.785	0.620									
PC3					0.640	0.589	0.492						
PC4											-1.000		
PC5												-1.000	
PC6													1.000

B.3.2 Training loss curves

Figure 10 reports the training loss curves on the Pitprops dataset for five random seeds. Across all seeds, the loss decreases rapidly in the early stage and reaches a stable regime after approximately 75 epochs. The zoomed-in inset further shows that the training losses remain stable from epoch 75 onward, with only small transient fluctuations for a few seeds. These results provide empirical convergence diagnostics for the unsupervised training procedure on the real dataset.

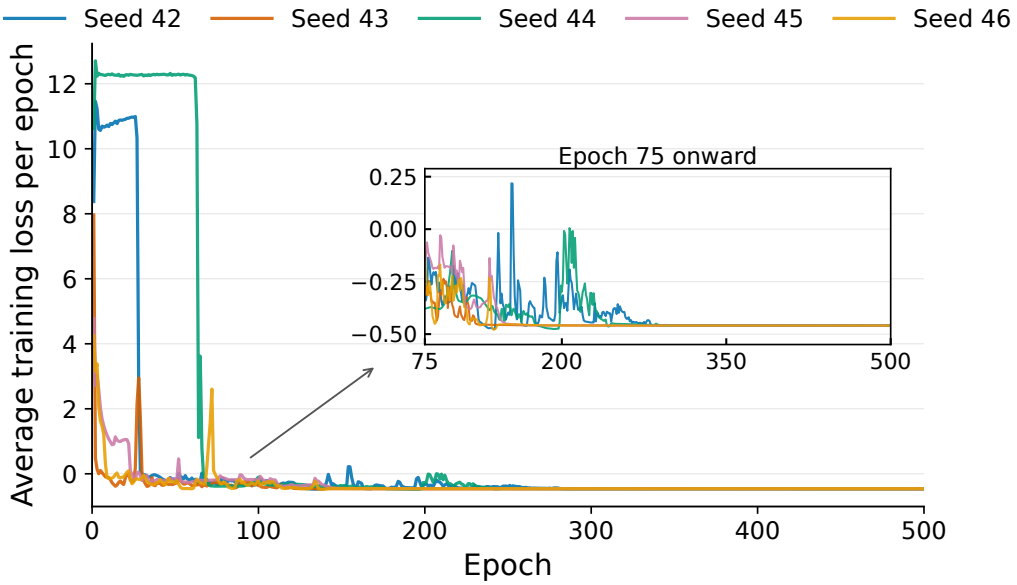

Figure 10: Training loss curves on the Pitprops dataset over five random seeds. Each colored curve corresponds to one random seed. The training loss decreases rapidly during the early epochs and stabilizes after approximately 75 epochs. The inset zooms in on the later stage from epoch 75 to epoch 500, showing that the losses across different seeds remain close to a stable level with only small transient fluctuations.

References

[1] Anthony, M. and Bartlett, P. L. (1999). *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, Cambridge.

[2] Bai, Y., Chen, F., Wang, H., Xiong, C., and Mei, S. (2023). Transformers as statisticians: Provable in-context learning with in-context algorithm selection. In *Thirty-seventh Conference on Neural Information Processing Systems*.

[3] Chen, Z., Wu, R., and Fang, G. (2026). Transformers as unsupervised learning algorithms: A study on gaussian mixtures. In *The Fourteenth International Conference on Learning Representations*.

[4] Davis, C. and Kahan, W. M. (1970). The rotation of eigenvectors by a perturbation. iii. *SIAM Journal on Numerical Analysis*, 7(1):1–46.

[5] He, Y., Cao, Y., Chen, H.-Y., Wu, D., Fan, J., and Liu, H. (2025). Learning spectral methods by Transformers. *arXiv preprint arXiv:2501.01312*.

[6] Jeffers, J. (1967). Two case studies in the application of principal component analysis. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 16(3):225–236.

[7] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E. Z., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché Buc, F., Fox, E. B., and Garnett, R., editors, *NeurIPS*, pages 8024–8035.

[8] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.*, 12(null):2825–2830.

[9] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners. Technical report, OpenAI.

[10] Yu, Y., Wang, T., and Samworth, R. J. (2015). A useful variant of the Davis–Kahan theorem for statisticians. *Biometrika*, 102(2):315–323.

[11] Yuan, X. and Zhang, T. (2013). Truncated power method for sparse eigenvalue problems. *Journal of Machine Learning Research*, 14:899–925.

[12] Zou, H., Hastie, T., and Tibshirani, R. (2006). Sparse principal component analysis. *Journal of Computational and Graphical Statistics*, 15(2):265–286.