ROSS-PCA: Robust Sparse PCA via Fully Differentiable Penalties

Anonymous authors
Paper under double-blind review

Abstract

Sparse PCA finds low-dimensional structure that loads on few features. Existing methods couple learning and feature selection by applying non-differentiable penalties that force retain-or-zero decisions during optimization. This eliminates features before the optimizer has established which ones matter, limits scalability to serial coordinate-update solvers, and fails when components share support. We introduce ROSS-PCA, which decouples learning and feature selection. It learns by optimizing a fully differentiable objective combining robust reconstruction, a smooth sparsity penalty, and an orthogonality term. It selects features post-hoc via cosine-preserving pruning. ROSS-PCA is GPU-accelerated, scaling sparse PCA to currently inaccessible large datasets. All loss terms are calibrated to be independent of data dimensionality and scale, giving penalty weights consistent meaning across datasets. On synthetic benchmarks, ROSS-PCA matches or outperforms established methods on support recovery with particular advantage under severe outlier contamination. The smooth formulation of ROSS-PCA enables stability analysis via random initializations since no feature is eliminated during training. We find that the method's feature selection is stable when the generating basis is theoretically identifiable and shows increased uncertainty precisely when it is not. On the Human Lung Cell Atlas ($584{,}944 \times 27{,}402$), gene selection is 97% consistent across independent random initializations, and consensus gene modules identified from reproducibility across runs are enriched for known biological pathways.

1 Introduction

Principal component analysis reduces dimensionality by finding directions of maximum variance, but every feature may contribute to every component. In high-dimensional settings such as transcriptomics (Wolf et al., 2018; Stuart et al., 2019) and population genetics (Patterson et al., 2006; Price et al., 2006), this makes components difficult to interpret. A component that loads on thousands of genes identifies a direction of variation without identifying which genes define it. Sparse PCA methods modify the objective so that each component loads on few features, combining dimensionality reduction with feature selection.

Dimensionality reduction and feature selection are two distinct tasks. Every existing sparse PCA method performs both simultaneously. The optimization objective enforces sparsity so that the solver produces loadings with exact zeros when it terminates. For exact zeros to be stationary points, the penalty must have a kink at the origin, a non-vanishing force toward zero that holds coefficients there once they arrive. This mechanism underlies every penalty designed for exact sparsity, including the lasso (Tibshirani, 1996), SCAD (Fan and Li, 2001), MCP (Zhang, 2010), and hard thresholding (Ma, 2013). Their solvers implement this geometry by applying a binary retain-or-zero decision to each coefficient at each iteration via thresholding or proximal operators (Parikh and Boyd, 2014). Learning and selection therefore happen simultaneously, coupled by the mechanics of the solver.

This coupling has two consequences. First, features are selected or discarded before the optimizer has established which ones matter. Correlated features compete for shared variance, and which survive thresholding depends on solver internals (update order, step size) rather than solely on the data (Zou and Hastie, 2005; Freijeiro-González et al., 2022). The problem is compounded when components share support. A feature

that contributes to multiple components may be thresholded to zero in some before the optimizer has found a good allocation. Camacho et al. (2020) showed that existing methods fail to recover known sparse structure even on noise-free data when components share features. Second, element-wise coordinate updates and proximal steps are inherently serial, leaving existing implementations CPU-bound and limited to moderate dimensionality (Pedregosa et al., 2011).

We introduce ROSS-PCA, which decouples learning and feature selection. ROSS-PCA optimizes a fully smooth objective via minibatch SGD, producing a loading matrix in which signal and noise entries separate by magnitude. Features are then selected post-hoc. No coefficient is set to zero during training, so the optimizer explores the full loading space without premature feature elimination. The smooth objective admits standard GPU-accelerated autodifferentiation, scaling linearly in all problem dimensions.

The objective combines reconstruction, sparsity, and orthogonality terms. All three are normalized so that the sparsity and orthogonality hyperparameters have consistent meaning independent of data scale and dimensionality, allowing the same settings to transfer across datasets. Robust sample weighting is integrated directly into the reconstruction loss, providing outlier resistance without separate preprocessing. Post-hoc feature selection uses cosine-preserving pruning, which controls the angular deviation from each learned direction rather than applying magnitude thresholds.

The smooth formulation enables a stability analysis that existing methods cannot provide. With ℓ_1 penalties, thresholding makes binary feature decisions at every iteration, so features eliminated early cannot be recovered. This makes random initialization impractical and ties existing methods to PCA initialization. With a smooth penalty, no feature is irrecoverably eliminated during training, so different random initializations can meaningfully explore the solution space. By fitting from many independent initializations, we measure which structure is consistent across solutions and which varies. On synthetic benchmarks where the identifiability of the generating structure is controlled by design, we observe that empirical stability mirrors theoretical identifiability (Lei and Vu, 2015). Feature selection is near-deterministic when the generating basis is identifiable (disjoint support) and degrades precisely when shared support makes the basis non-identifiable. This correspondence, observed but not formally proven, suggests that the variation across initializations reflects properties of the data rather than artifacts of the optimizer.

We validate on synthetic benchmarks using the spiked covariance model (Johnstone and Lu, 2009) in a factorial design that crosses support overlap with eigenvalue separation at three signal-to-noise levels. The method stably identifies active features across all configurations but shows increased uncertainty in component assignments when the generating basis is theoretically non-unique. On the Human Lung Cell Atlas ($584{,}944 \times 27{,}402$), a scale inaccessible to all existing sparse PCA methods, gene selection is 97% consistent across independent random initializations, and consensus gene modules identified from reproducibility across runs are enriched for known biological pathways.

2 Methods

ROSS-PCA is a two-stage pipeline (Figure 1A). *Learning* (Section 2.1) optimizes a fully smooth objective to produce loadings in which signal and noise entries separate by magnitude, and *feature selection* (Section 2.2) imposes exact zeros post-hoc via cosine-preserving pruning. Because the smooth objective supports arbitrary initialization, fitting the pipeline from multiple independent random initializations provides a stability analysis (Figure 1B, Section 2.3) that identifies empirically stable structure in the decomposition.

2.1 Learning

2.1.1 Objective

Given a data matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$ (rows are samples, columns are features), we seek k loading vectors $\mathbf{W} \in \mathbb{R}^{p \times k}$ that minimize

$$\mathcal{L}(\mathbf{X}, \mathbf{W}) = \mathcal{L}_{\text{recon}}(\mathbf{X}, \mathbf{W}) + \lambda_s \, \mathcal{L}_{\text{sparse}}(\mathbf{W}) + \lambda_{\text{orth}} \, \mathcal{L}_{\text{orth}}(\mathbf{W}), \tag{1}$$

$\mathcal{L}_{\text{recon}}$ measures reconstruction quality. $\mathcal{L}_{\text{sparse}}$ drives small loading entries toward zero. $\mathcal{L}_{\text{orth}}$ penalizes inter-component correlation, maintaining near-orthonormality of $\mathbf{W}$. The objective is optimized by minibatch

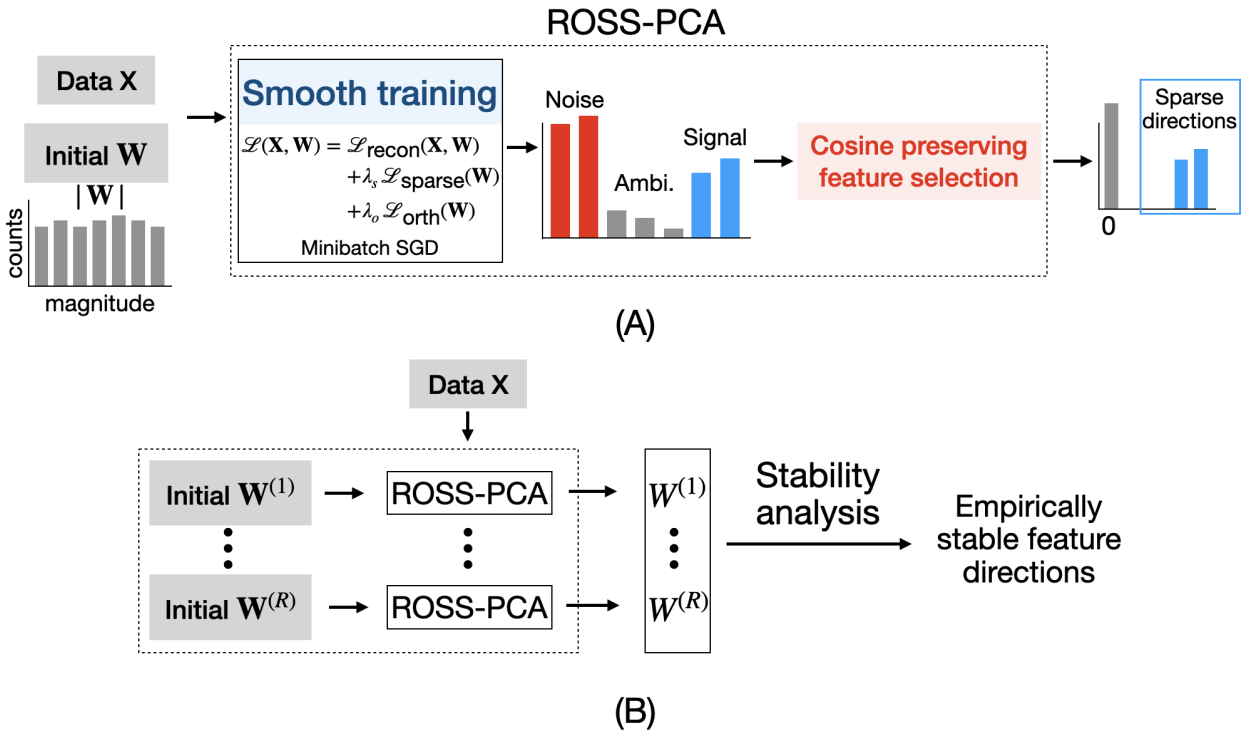

Figure 1: ROSS-PCA overview. **(A)** A single run. Smooth training optimizes a fully differentiable objective via minibatch SGD, producing loadings in which signal, ambiguous, and noise entries separate by magnitude. Cosine-preserving pruning then zeros the smallest entries per component while bounding the angular deviation from the learned direction, yielding sparse directions. No coefficient is set to zero during training. **(B)** Stability analysis. The pipeline is run from R independent random initializations on the same data, producing R pruned loading matrices $\mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(R)}$. Comparing these identifies which features and component assignments are consistent across runs.

stochastic gradient descent over batches of b samples drawn from $\mathbf{X}$ and $\mathbf{W}$ is initialized as either the top-k PCA components or random orthogonal vectors.

2.1.2 Robust Reconstruction Loss

Let $\bar{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i \in \mathbb{R}^p$ denote the column mean. For a centered observation $\tilde{\mathbf{x}}_i = \mathbf{x}_i - \bar{\mathbf{x}}$, the scores are $\mathbf{z}_i = \mathbf{W}^\top \tilde{\mathbf{x}}_i$ and the residual is $\mathbf{r}_i = \tilde{\mathbf{x}}_i - \mathbf{W}\mathbf{z}_i$. In matrix form, the reconstruction of the centered data is $\tilde{\mathbf{X}}\mathbf{W}\mathbf{W}^\top$, which equals the optimal rank-k projection when $\mathbf{W}$ has orthonormal columns.

The reconstruction loss is the weighted mean squared residual normalized by a robust variance estimate $\hat{\sigma}^2_{\text{rob}}$:

$$\mathcal{L}_{\text{recon}}(\mathbf{X},\mathbf{W}) = \frac{1}{\hat{\sigma}^2_{\text{rob}}} \cdot \frac{1}{bp} \sum_{i=1}^{b} \sum_{j=1}^{p} \bar{w}_i \, r_{ij}^2, \tag{2}$$

where $\bar{w}_i$ downweights outlying samples and the division by $\hat{\sigma}^2_{\text{rob}}$ normalizes the loss to $O(1)$ regardless of data scale. We define $\hat{\sigma}_{\text{rob}} = 1.4826 \cdot \text{median}_{j:\,\text{MAD}_j>0}(\text{MAD}_j)$, where $\text{MAD}_j = \text{median}_i\, |x_{ij} - \bar{x}_j|$ (see Appendix A.2 for details).

We define the per-sample robustness weight $\bar{w}_i$ as follows. Within each minibatch, we compute the root-mean-square residual e_i and derive weights via a tanh function:

$$e_i = \sqrt{\frac{1}{p}\sum_{j=1}^{p} r_{ij}^2}, \qquad w_i = \frac{\tanh(e_i)}{e_i}, \qquad \bar{w}_i = \frac{w_i}{\frac{1}{b}\sum_{i=1}^{b} w_i}. \tag{3}$$

Samples with small residuals receive $w_i \approx 1$ (recovering squared-error behavior), while samples with large residuals receive $w_i \approx 0$ (effectively discarded).

2.1.3 Smooth Sparsity Penalty

The sparsity and orthogonality penalties operate on the column-normalized loading matrix $\hat{\mathbf{W}}$, where $\hat{\mathbf{w}}_\ell = \mathbf{w}_\ell / \|\mathbf{w}_\ell\|_2$, decoupling the direction of each component from its magnitude.

The sparsity loss is

$$\mathcal{L}_{\text{sparse}}(\hat{\mathbf{W}}) = \frac{1}{pk} \sum_{j=1}^{p} \sum_{\ell=1}^{k} \phi(\hat{W}_{j\ell};\, \varepsilon), \tag{4}$$

where

$$\phi(x; \varepsilon) = \log\left(1 + \frac{x^2}{\varepsilon^2}\right) \tag{5}$$

is a fully smooth (C^∞) version of the log penalty $\log(|x| + \varepsilon)$ of Candès et al. (2008). The gradient increases from zero at $x = 0$ to a maximum of $1/\varepsilon$ at $|x| = \varepsilon$, then decays as $2/|x|$ for $|x| \gg \varepsilon$. This provides intrinsic adaptive lasso behavior (Zou, 2006) without explicit reweighting. We set $\varepsilon_0 = 1/\sqrt{p}$ since entries of $\hat{\mathbf{W}}$ are $O(1/\sqrt{p})$ at initialization.

2.1.4 Orthogonality Penalty

The orthogonality loss is

$$\mathcal{L}_{\text{orth}}(\hat{\mathbf{W}}) = \frac{1}{k^2} \left\| \hat{\mathbf{W}}^\top \hat{\mathbf{W}} - \mathbf{I}_k \right\|_F^2, \tag{6}$$

which penalizes the off-diagonal entries of $\hat{\mathbf{W}}^\top \hat{\mathbf{W}}$ (the cosines of inter-component angles).

2.1.5 Optimization

PCA baseline calibration. The reconstruction loss of the top-k PCA solution is a lower bound on $\mathcal{L}_{\text{recon}}$. When this floor is large, the values of λ_s and λ_{orth} needed to make the sparsity and orthogonality penalties compete with reconstruction depend on the data. Without calibration, finding effective penalty weights would require dataset-specific exploration. We eliminate this by scaling the penalties by the PCA reconstruction loss $\mathcal{L}_{\text{recon}}^{\text{PCA}} = \mathcal{L}_{\text{recon}}(\mathbf{X}, \mathbf{W}_{\text{PCA}})$, evaluated on a small subsample of batches before training. The calibrated objective is

$$\mathcal{L}(\mathbf{X}, \mathbf{W}) = \mathcal{L}_{\text{recon}}(\mathbf{X}, \mathbf{W}) + \lambda_s \, \mathcal{L}_{\text{recon}}^{\text{PCA}} \, \mathcal{L}_{\text{sparse}}(\hat{\mathbf{W}}) + \lambda_{\text{orth}} \, \mathcal{L}_{\text{recon}}^{\text{PCA}} \, \mathcal{L}_{\text{orth}}(\hat{\mathbf{W}}). \tag{7}$$

Computation details are in Appendix A.1.

Solver and schedule. The objective (7) is minimized by minibatch stochastic gradient descent using AdamW (Loshchilov and Hutter, 2019). Training is steps-based (not epoch-based), so all hyperparameters are expressed in gradient steps, making them independent of dataset size and batch size. The learning rate η follows a two-phase schedule, with linear warmup from $\eta_{\max}/100$ to $\eta_{\max}$, then cosine decay to $\eta_{\max}/10$. After the scheduled T steps, training optionally continues at $\eta_{\max}/10$ with patience-based stopping. Each minibatch gradient step requires $O(Bpk)$ operations, linear in all problem dimensions, and is JIT-compiled for GPU execution. The training procedure is summarized in Algorithm 1 (Appendix A.7).

Sparsity warmup. Applying full sparsity pressure from initialization can trap the optimizer in poor local minima before the reconstruction term has established a good subspace. The sparsity term is therefore multiplied by a cosine annealing factor $\alpha(t) = \frac{1}{2}(1 - \cos(\pi t / T_{\text{warm}}))$, which ramps from 0 to 1 over T_{warm} warmup steps. During warmup, ε is held fixed at $\varepsilon_0 = 1/\sqrt{p}$.

Adaptive threshold via Otsu's method. As training progresses and the sparsity penalty drives small entries toward zero, the distribution of $|\hat{W}_{j\ell}|$ develops a concentrated mass near zero separated from a tail of signal entries. After warmup, we adapt ε to track the boundary between these populations at intervals of T_ε steps using Otsu's method (Otsu, 1979), which finds the threshold maximizing between-class variance on the histogram of $|\hat{W}_{j\ell}|$ values. The update is floored at the initial value to prevent ε from collapsing below the natural scale of $\hat{W}$ entries:

$$\varepsilon \leftarrow \max\bigl(\varepsilon_{\text{Otsu}},\ 1/\sqrt{p}\bigr). \tag{8}$$

The floor at $1/\sqrt{p}$ also bounds the peak sparsity gradient at $\sqrt{p}$, preventing the sparsity term from dominating reconstruction.

Gradient propagation. Both the sparsity and orthogonality terms operate on $\hat{\mathbf{W}}$ with columns $\hat{\mathbf{w}}_\ell = \mathbf{w}_\ell/\|\mathbf{w}_\ell\|_2$, but they differ in how gradients propagate to $\mathbf{W}$. For the sparsity term, the column norms $\|\mathbf{w}_\ell\|_2$ are treated as constants (stop-gradient), so $\partial\mathcal{L}_{\text{sparse}}/\partial\mathbf{W}$ acts only on the direction of each column and the optimizer cannot inflate $\|\mathbf{w}_\ell\|_2$ to shrink $|\hat{W}_{j\ell}|$ and weaken the penalty. For the orthogonality term, gradients flow through the full normalization $\hat{\mathbf{w}}_\ell = \mathbf{w}_\ell/\|\mathbf{w}_\ell\|_2$, allowing the optimizer to rotate component directions to reduce inter-component correlation. For the reconstruction term, the per-sample weights $\bar{w}_i$ are detached from the computational graph (stop-gradient on $\bar{w}_i$) to prevent the optimizer from minimizing the loss by manipulating the weights rather than improving the reconstruction (Holland and Welsch, 1977).

Initialization. Initialization defaults to the top-k right singular vectors of the centered data via randomized SVD (Halko et al., 2011) (PCA initialization). We also provide random initialization. Draw a random $p \times k$ Gaussian matrix, orthonormalize via QR decomposition, and scale by $\hat{\sigma}_{\text{rob}}$ to match the data's robust variance scale.

2.2 Feature Selection

Cosine-preserving pruning (CPP). The learning stage produces near-orthonormal loading vectors whose directions encode the subspace structure (Figure 1A). Pruning strategies based on magnitude thresholding or percentile cutoffs may deviate the directions from the learned ones. We designed CPP to control the angular deviation directly. For each loading vector, we zero out the smallest-magnitude entries while bounding the cosine distance from the learned direction by a threshold θ (Algorithm 2, Appendix A.8). Sweeping θ on a uniform grid from $0°$ to $\theta_{\max}$ (default $10°$, 100 points) yields a one-parameter family of solutions from unpruned to maximally sparse without retraining. Evaluating sparsity and variance explained (VE) at each θ traces a Pareto frontier. To select a single operating point, we identify the knee of this frontier.

Operating point selection. We normalize both axes to $[0, 1]$ per curve and smoothen the curve via a Savitzky–Golay filter (Savitzky and Golay, 1964). We compute the rate of change of the tangent angle $\vartheta = \arctan(d\,\text{VE}/d\sigma)$ with respect to normalized sparsity σ:

$$\frac{d\vartheta}{d\sigma} = \frac{d^2\text{VE}/d\sigma^2}{1 + (d\,\text{VE}/d\sigma)^2}. \tag{9}$$

The operating point is the sparsity at which $d\vartheta/d\sigma$ is most negative, the sharpest transition from flat to steep on the frontier. Compared to arc-length curvature (which uses a $3/2$ exponent in the denominator), this criterion is less sensitive to early gentle bends and selects the onset of substantial degradation rather than the first detectable change.

2.3 Stability Analysis via Multiple Initializations

The sparse decomposition may not be uniquely identifiable from the data. With ℓ_1 methods, thresholding eliminates features irrecoverably at each iteration, making random initialization impractical and tying these methods to PCA initialization. The smooth ROSS-PCA objective has no such constraint, so different random initializations can meaningfully explore the solution space. We fit the pipeline R times from independent random orthonormal initializations, each followed by CPP (Figure 1B), producing R pruned loading matrices

$\mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(R)}$. Variation across these matrices reflects two sources that we do not attempt to disentangle: the non-convex optimization may converge to different local minima, and CPP's angular threshold may prune differently depending on the learned loadings. Structure that is consistent across solutions is robustly determined by the data. The consensus procedure below is robust to both sources because features at the pruning boundary are inconsistent across runs and are filtered out.

To measure consistency, we match components across runs via optimal one-to-one assignment (Hungarian algorithm on the Jaccard similarity matrix of component supports). For each component j, we define a *stability frequency* f_{jg} as the fraction of runs in which feature g is active in the matched partner of component j. The *consensus core* at threshold τ is the set of features with $f_{jg} \geq \tau$, and the *stable support* is the union of all consensus cores. At the global level, the pairwise Jaccard similarity of active feature sets across all $\binom{R}{2}$ pairs quantifies overall selection consistency. The full matching procedure is detailed in Appendix D.

Consensus cores provide a per-feature, per-component confidence annotation. Core features ($f_{jg} \geq \tau$) are stably co-assigned across independent solutions, while peripheral features ($f_{jg} < \tau$) reflect ambiguity in the decomposition. This extends stability selection (Meinshausen and Bühlmann, 2010) from binary feature inclusion to component-level assignment. Methods that produce a unique solution by construction cannot provide this diagnostic.

2.4 Evaluation Metrics

We evaluate three aspects of each method's output. *Reconstruction quality* is measured by variance explained (VE) at a given sparsity level. *Feature recovery* is measured by global area under the precision–recall curve (AUPRC), which ranks features by their row norms across all components and tests whether active features are ranked above inactive ones. Because it is invariant to component permutation and rotation within the loading subspace, global AUPRC is the primary recovery metric. *Orthogonality* is the mean absolute off-diagonal of the direction-normalized Gram matrix $\hat{\mathbf{W}}^\top \hat{\mathbf{W}}$, with lower values indicating less inter-component correlation. Additional metrics including per-component recovery and subspace angles are defined in Appendix B.2.

3 Results

We evaluate ROSS-PCA on synthetic benchmarks with known generating structure, testing feature recovery, reconstruction quality, stability, robustness, and computational scaling, and on the Human Lung Cell Atlas (HLCA), a single-cell transcriptomics dataset at a scale inaccessible to existing sparse PCA methods. Experimental details for synthetic benchmarks are in Appendix B and for the HLCA in Appendix D.

We use the spiked covariance model (Johnstone and Lu, 2009) ($p = 500$, $k = 5$, $n = 1{,}000$):

$$X \sim \mathcal{N}(0, \Sigma), \qquad \Sigma = \mathbf{V}\mathbf{D}\mathbf{V}^\top + \sigma^2 \mathbf{I}_p, \tag{10}$$

where $\mathbf{V} \in \mathbb{R}^{p \times k}$ has sparse, unit-norm columns, $\mathbf{D} = \mathrm{diag}(d_1, \ldots, d_k)$ contains spike eigenvalues, and σ^2 is isotropic noise variance. We refer to $\mathbf{V}$ as the *generating basis*, the set of features with at least one nonzero entry in $\mathbf{V}$ as the *generating support*, and columns of $\mathbf{V}$ as the *generating components*. Features nonzero in exactly one generating component are *private* and those in two or more are *shared*.

Baselines include Zou–Hastie SparsePCA (Zou and Hastie, 2005), MiniBatch SparsePCA (Pedregosa et al., 2011), VarPro SPCA (Erichson et al., 2020), and PMD/SPC (Witten et al., 2009), each sweeping their sparsity parameter over a grid. All methods use PCA initialization. For baselines, each hyperparameter configuration produces a model with native sparsity. For ROSS-PCA, results reflect the full pipeline, training followed by CPP operating point selection.

3.1 Factorial Design (S1–S4)

A 2×2 factorial design crosses support overlap (0% vs. 25%) with eigenvalue separation (well-separated vs. degenerate), yielding four structural configurations (S1–S4). This design is motivated by the identifiability

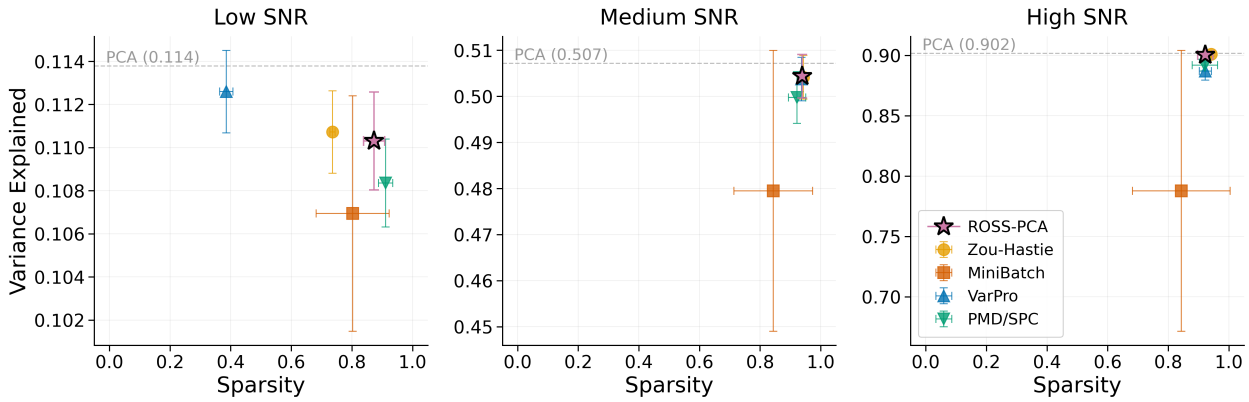

Figure 2: Sparsity vs. variance explained for S1 (disjoint, separated eigenvalues) at three SNR levels. PCA dense ceiling shown as dashed line. ROSS-PCA ($\star$) matches the PCA ceiling at above 90% sparsity across all SNR levels. Each point represents one hyperparameter configuration. Error bars show ± 1 std over 100 replicates.

conditions of Lei and Vu (2015), who showed that support overlap and eigenvalue separation jointly determine whether the sparse generating basis is recoverable. Each configuration is crossed with three SNR levels ($\sim$10%, $\sim$50%, $\sim$90% variance explained) and 100 replicates, for 1,200 datasets. See Appendix Table 4 for full specifications.

Across all four configurations at high SNR, every method except MiniBatch SparsePCA achieves sparsity above 90% with variance explained within 1% of the PCA ceiling. Global AUPRC is $\sim$0.97 for ROSS-PCA, Zou–Hastie, and PMD/SPC, dropping to 0.92 for VarPro under overlap (S3, S4). MiniBatch SparsePCA consistently underperforms on all metrics. Results at medium and low SNR follow the same ranking with smaller effect sizes (Appendix Figures 7–13). Figure 2 shows the VE–sparsity frontier for S1, the identifiable baseline (Lei and Vu, 2015).

S1c: robustness to outlier contamination. To isolate the contribution of robust sample weighting, we contaminate S1 with a 2×2 factorial crossing contamination fraction $\{2\%, 5\%\}$ with outlier magnitude $\{4\times, 8\times\}$ the data's standard deviation (Appendix Table 5). Since the only difference from S1 is contamination, any performance gap reflects how each method handles corrupted samples.

At mild contamination (2% at 4×), all methods recover well at medium and high SNR (global AUPRC above 0.93). The robust weighting is unnecessary. As contamination increases, baselines degrade while ROSS-PCA maintains recovery. At 5% at 8× and low SNR, ROSS-PCA achieves global AUPRC 0.71 while all baselines fall below 0.32, more than a 2× improvement. At medium SNR under the same contamination, ROSS-PCA achieves 0.89 versus the best baseline at 0.82 (PMD/SPC). Figure 3 shows the contamination grid.

S2: disjoint support and degenerate eigenvalues. S2 retains the disjoint support of S1 but makes all spike eigenvalues equal. Any rotation within the k-dimensional eigenspace is equally optimal for reconstruction, so individual generating components are not identifiable. The appropriate metric is subspace angles, which measure recovery of the column space of $\mathbf{V}$ independently of basis choice. On this metric, ROSS-PCA and Zou–Hastie are equivalent (Figure 9). Apparent differences in component-level metrics reflect basis non-identifiability rather than recovery quality (Appendix B.2).

S3/S4: overlapping support. S3 and S4 are nearly indistinguishable on all metrics, indicating that eigenvalue degeneracy adds little cost on top of overlap. Full per-configuration figures are in Appendix (Figures 10–13). Under overlapping support, the generating basis is not uniquely determined by the data (Vu and Lei, 2013; Camacho et al., 2020). We explore this ambiguity empirically via stability analysis in the next section.

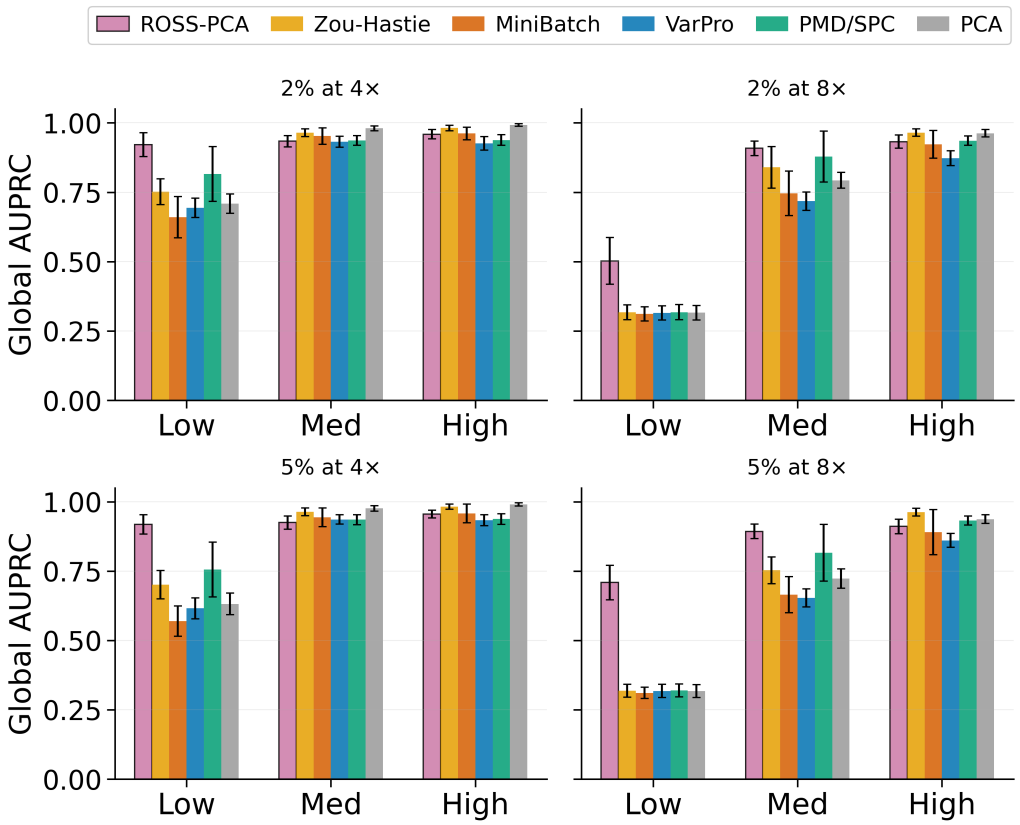

Figure 3: Feature recovery under outlier contamination (S1c). Rows show contamination fraction (2%, 5%) and columns show outlier magnitude (4×, 8×). Each panel shows global AUPRC at three SNR levels. ROSS-PCA's advantage emerges at severe contamination (bottom-right), with global AUPRC 0.71 vs. < 0.32 for all baselines at low SNR.

3.2 Stability across initializations

We apply the multi-initialization procedure (Section 2.3) with $R = 100$ independent random orthonormal initializations per replicate across all four factorial configurations at three SNR levels (Appendix Table 6). S1 and S2 serve as theoretically identifiable controls, while S3 and S4 test the effect of overlap. Figure 4 reports all metrics at high SNR.

Generating support stability. Global AUPRC across the 100 solutions exceeds 0.96 for all configurations at high SNR, showing that the generating support is stably recovered regardless of overlap or eigenvalue structure. This is expected since the generating support is the row support of the top-k eigenspace, which is invariant to basis choice.

Feature assignment stability. The probability that a private feature is placed in exactly one estimated component (Q2) is $\sim$0.95 under S1 and S2, confirming that eigenvalue degeneracy alone does not degrade feature-level assignments. Under overlap, Q2 drops to $\sim$0.71, a $\sim$24 percentage point decrease reflecting the non-uniqueness of the generating basis. Shared features are reliably detected as multi-component (Q3 ≈ 0.89 under S3 and S4), partly explaining the Q2 drop as shared-feature activity spreads into neighboring private features.

Component-level co-assignment. Q4 measures whether features are consistently placed in the same component across runs. Within-component co-assignment (Q4 co) remains above 0.88 across all configurations, indicating that features belonging to the same component are stably grouped together. Cross-

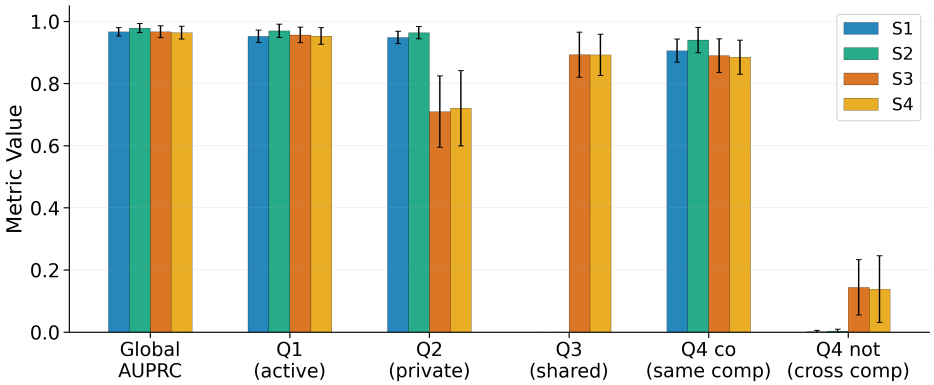

Figure 4: Feature stability at high SNR across the factorial design (mean ± std over 100 replicates, 100 random initializations each). Global AUPRC exceeds 0.96 for all configurations, confirming stable generating support recovery. Q2 (private features placed in exactly one component) drops sharply under overlap (S3, S4) while Q3 (shared features detected as multi-component) is high, indicating that the method trades private-feature precision for shared-feature detection. Q4 co (within-component co-assignment) remains high throughout. Q4 not (cross-component false co-assignment) rises under overlap. S3 and S4 are nearly indistinguishable, indicating that eigenvalue degeneracy adds little ambiguity beyond overlap itself. S1/S2 have no shared features, so Q3 bars are absent. Full SNR breakdown in Appendix Table 7.

component false co-assignment (Q4 not) is negligible under S1 and S2 (< 0.01) but rises to $\sim$0.14 under overlap, meaning features from different generating components are sometimes merged into a single estimated component. This is the component-level signature of the ambiguity that overlap introduces. On the HLCA, where no generating basis is available, we apply the full consensus core procedure (Section 2.3) to identify stably co-assigned feature sets within each component (Section 3.4). Table 7 shows the full SNR $\times$ configuration breakdown.

3.3 Computational scaling

Figure 5 shows wall-clock time on scaling benchmarks where n, p, and k increase jointly (Appendix Table 8). Times are averaged across all hyperparameter configurations per method, and ROSS-PCA time includes both fit and CPP. ROSS-PCA scales from $\sim$10 s at $(n, p, k) = (5000, 500, 10)$ to $\sim$5 min at $(150000, 15000, 300)$, with negligible variance across configurations. Zou–Hastie and PMD/SPC both exceed the 6-hour timeout at $p \geq 10000$. MiniBatch SparsePCA is the only baseline that completes at all sizes, but requires 1.1 h at the largest problem compared to $\sim$5 min for ROSS-PCA, and achieves consistently worse reconstruction and support recovery (Section 3.1).

3.4 Application: Human Lung Cell Atlas

We apply ROSS-PCA to the Human Lung Cell Atlas (HLCA) core (Sikkema et al., 2023): 584,944 cells $\times$ 27,402 genes, log-normalized single-cell RNA sequencing data. At $k = 5{,}000$ with 100 independent random initializations, gene selection is 97% consistent, consensus gene modules identified from reproducibility across runs are enriched for known biological pathways, and the stable gene support (47% of active genes) carries 98.7% of the model's variance.

This dataset is inaccessible to all existing sparse PCA methods because the dense matrix requires $\sim$64 GB in float32, and all baselines operate on the dense matrix. ROSS-PCA works directly on the sparse representation ($\sim$3 GB), densifying only one minibatch at a time. Full configuration details are in Appendix D.

Gene selection is stable. Figure 6A shows gene selection frequency across 100 random initializations. Of the 27,402 genes, 53% are active in every run, 39% are never active, and fewer than 8% are ambiguous. The mean pairwise Jaccard similarity of active gene sets is 0.97, confirming that gene selection is driven by data

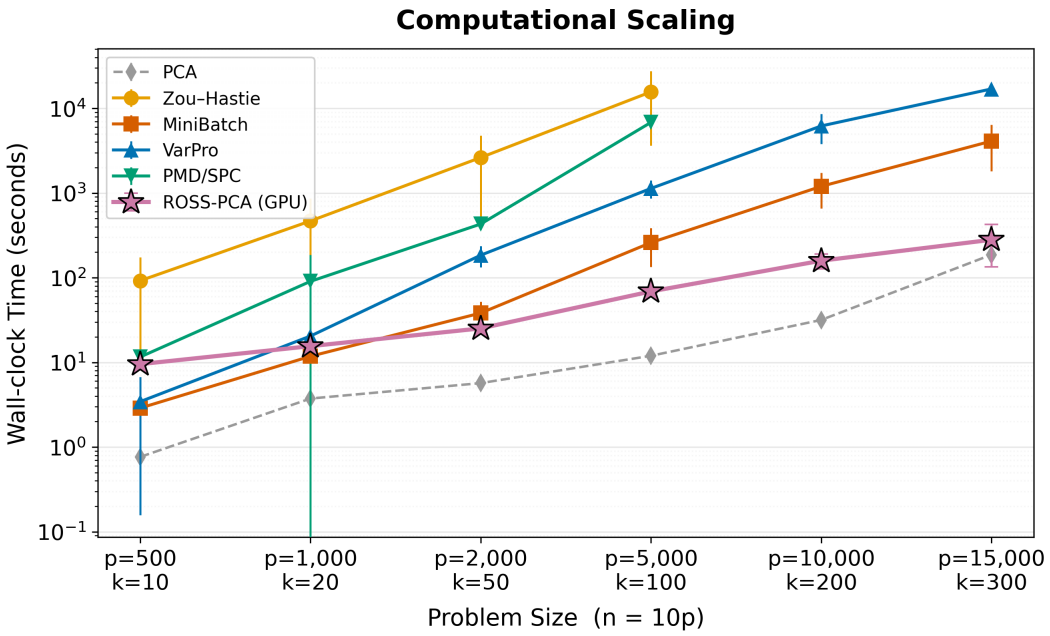

Figure 5: Computational scaling. Wall-clock time (log y-axis) versus problem size. Problem dimensions (p, n, k) increase jointly with $s = 30$ fixed (Table 8). Lines show mean time across hyperparameter configurations. Error bars show ± 1 std. ROSS-PCA time includes fit and CPP. Missing segments indicate timeout ($> 6\,\mathrm{h}$) or failure.

structure rather than initialization. The ambiguous genes reflect two sources of variation that we do not attempt to disentangle: different initializations may converge to different local optima, and CPP's angular threshold may prune differently depending on the learned loadings. Rather than isolating these sources, the consensus core analysis below identifies features that are stable regardless of both.

Component-level stability: consensus cores. Gene-level consistency does not address whether genes are assigned to the *same component* across runs. We apply the consensus core procedure (Section 2.3), matching components across runs via optimal one-to-one assignment on Jaccard similarity (Appendix D).

All 5,000 components have non-empty consensus cores at the 90% threshold (Figure 19). Core size declines gradually from 80% to 90% stability and drops sharply above 90%, indicating a natural boundary between stably co-assigned and peripherally assigned genes (Appendix Figure 20).

Stable cores carry nearly all variance. Restricting loadings to the union of all 90%-consensus-core genes (47% of active genes) and recomputing reconstruction yields 98.7% of the full model's variance explained (Figure 6B). The remaining genes collectively contribute less than 1 percentage point. At the 95% threshold, support shrinks further and VE drops to 47%, indicating a sharp transition between stable and ambiguous signal (Appendix Table 9).

Stable cores are biologically coherent. To validate that the reproducible structure corresponds to real biology, we perform over-representation analysis (Fisher's exact test via GSEApy, Fang et al. 2022) on the 90%-consensus cores of the 50 largest components, testing against GO Biological Process, KEGG, and Reactome. All 50 cores show significant enrichment (adjusted $p < 0.05$), with top-term p-values below 10^{-10}. Notably, the enriched pathways are not generic housekeeping programs but reflect biology specific to lung tissue (Table 1), including cilium assembly (the motile cilia of airway epithelial cells), *Staphylococcus aureus* infection (a leading cause of bacterial pneumonia), antimicrobial humoral response and antigen processing (mucosal immune defense in the respiratory tract), focal adhesion (structural integrity of lung parenchyma), and cytokine response pathways characteristic of the pulmonary immune environment. Gene

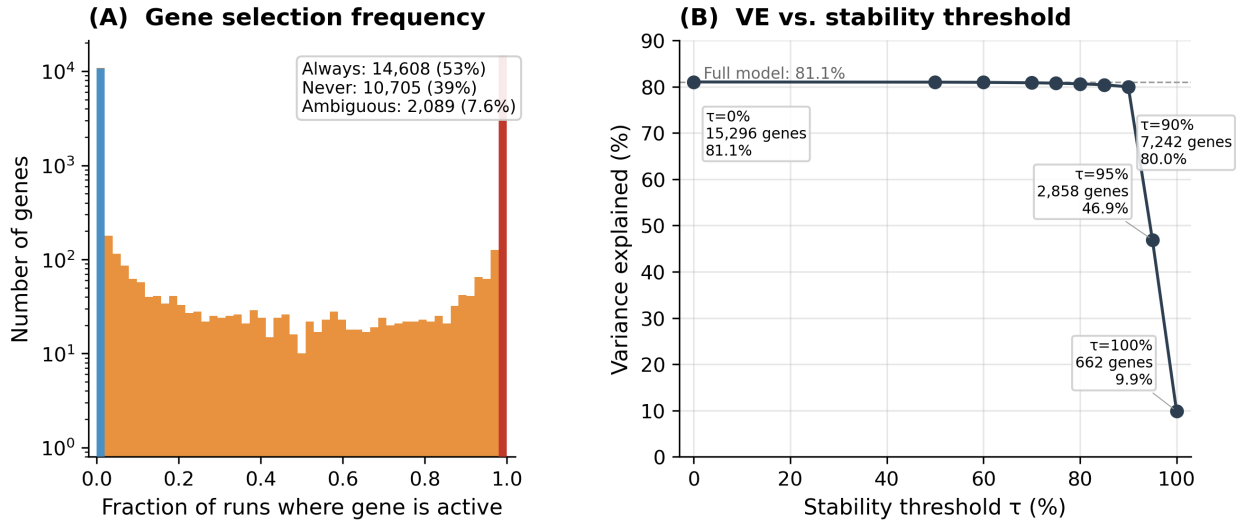

Figure 6: HLCA stability across 100 random initializations at $k = 5{,}000$. (A) Gene selection frequency. The distribution is strongly bimodal: 53% of genes are active in every run (red), 39% are never active (blue), and only 8% are ambiguous (orange). (B) Variance explained vs. stability threshold τ. The stable support (support-90, 7,242 genes, 47% of active genes) retains 98.7% of the full model's VE, with a sharp cliff between $\tau = 90\%$ and $\tau = 95\%$. Core size distribution in Appendix Figure 19.

Table 1: Functional enrichment of selected consensus cores (core-90). Over-representation analysis (Fisher's exact test) against GO Biological Process, KEGG, and Reactome (background: 14,990 active genes). All 50 top components are significantly enriched. Selected examples span distinct lung-relevant biological programs.

Core size	Top GO BP	Top KEGG / Reactome	Adj. p
1,869	Cytoplasmic Translation	Ribosome (KEGG)	8.3×10^{-40}
1,727	Cilium Assembly	Cilium Assembly (Reactome)	9.7×10^{-35}
634	Antimicrobial Humoral Response	Immune System (Reactome)	4.1×10^{-25}
551	Defense Response to Bacterium	Antigen processing (KEGG)	4.3×10^{-11}
492	Granulocyte Chemotaxis	*S. aureus* infection (KEGG)	2.5×10^{-12}
410	Regulation of Cell Migration	Focal adhesion (KEGG)	3.2×10^{-14}
376	Cellular Response to Cytokine Stimulus	Antigen processing (KEGG)	7.4×10^{-12}

modules identified from reproducibility across runs, without any biological input, recover lung-specific pathways, providing independent validation that the consensus cores capture tissue-relevant structure rather than generic transcriptional noise.

4 Discussion

ROSS-PCA demonstrates that sparse PCA does not require sparsity during training. A smooth objective produces loadings in which signal and noise entries separate by magnitude, and post-hoc pruning imposes zeros at any desired sparsity level from a single trained model. This decoupling enables GPU-accelerated minibatch optimization, scaling sparse PCA to dataset sizes ($584{,}944 \times 27{,}402$) inaccessible to all existing methods.

Addressing known failure modes. Camacho et al. (2020; 2021; 2025) showed that ℓ_1 methods fail when components share features, because thresholding cannot represent graded contributions to multiple components and deflation propagates errors across components. ROSS-PCA avoids both failure modes. The smooth penalty retains graded loadings throughout training, and all k components are optimized simulta-

neously. CPP is applied independently per component after convergence, so pruning one component cannot corrupt another.

Stability as a diagnostic. The multi-initialization analysis reframes the stability question from "does the method converge to the same solution?" to "what does the data determine?" On synthetics, feature assignments are 95% consistent when the generating structure is identifiable, and the ∼24 percentage-point cost of overlapping support is attributable to the support structure itself, not the method. On the HLCA, consensus gene modules identified from reproducibility across runs are enriched for lung-specific biological pathways, providing independent validation that the reproducible structure is biologically meaningful.

Relationship to closest prior work. The Fantope relaxation (Vu et al., 2013) and its stochastic variant (Qiu et al., 2023) are methodologically closest, as both use minibatch gradient descent for sparse PCA targeting genomics. Three differences are substantive. First, the Fantope framework retains ℓ_1 constraints, so binary feature selection occurs during optimization. Second, it optimizes a $p \times p$ projection matrix ($O(p^3)$ per step for eigendecomposition), while ROSS-PCA optimizes the $p \times k$ loading matrix directly ($O(bpk)$ per step). Third, the Fantope's convexity precludes using multiple initializations to probe identifiability. Additional comparisons to robust sparse PCA methods and the combinatorial literature are in Appendix F.

Limitations and future work. ROSS-PCA requires GPU for competitive wall-clock times, though a CPU fallback remains faster than most baselines at large p. The objective is non-convex, so there are no global optimality guarantees, though the multi-initialization analysis provides empirical evidence that solution quality is data-determined. λ_s and λ_{orth} are dimension-independent (Section 2.1.5), but optimal values still depend on the desired sparsity level and signal structure.

References

Boyle, E. A., Li, Y. I., and Pritchard, J. K. (2017). An expanded view of complex traits: from polygenic to omnigenic. *Cell*, 169(7):1177–1186.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. (2018). JAX: composable transformations of Python+NumPy programs. `http://github.com/google/jax`.

Berk, L. and Bertsimas, D. (2019). Certifiably optimal sparse principal component analysis. *Mathematical Programming Computation*, 11:381–420.

Bertsimas, D., Cory-Wright, R., and Pauphilet, J. (2022). Solving large-scale sparse PCA to certifiable (near) optimality. *Journal of Machine Learning Research*, 23(13):1–35.

Cory-Wright, R. and Pauphilet, J. (2022). Sparse PCA with multiple components. *arXiv preprint arXiv:2209.14790*.

Camacho, J., Smilde, A. K., Saccenti, E., and Westerhuis, J. A. (2020). All sparse PCA models are wrong, but some are useful. Part I: Computation of scores, residuals and explained variance. *Chemometrics and Intelligent Laboratory Systems*, 196:103907.

Camacho, J., Smilde, A. K., Saccenti, E., Westerhuis, J. A., and Bro, R. (2021). All sparse PCA models are wrong, but some are useful. Part II: Limitations and problems of deflation. *Chemometrics and Intelligent Laboratory Systems*, 208:104212.

Camacho, J., Smilde, A. K., Saccenti, E., Westerhuis, J. A., and Bro, R. (2025). All sparse PCA models are wrong, but some are useful. Part III: Model interpretation. *Chemometrics and Intelligent Laboratory Systems*, 266:105498.

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

Candès, E. J., Wakin, M. B., and Boyd, S. P. (2008). Enhancing sparsity by reweighted ℓ_1 minimization. *Journal of Fourier Analysis and Applications*, 14(5–6):877–905.

Otsu, N. (1979). A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66.

Croux, C., Filzmoser, P., and Fritz, H. (2013). Robust sparse principal component analysis. *Technometrics*, 55(2):202–214.

d'Aspremont, A., El Ghaoui, L., Jordan, M. I., and Lanckriet, G. R. (2007). A direct formulation for sparse PCA using semidefinite programming. *SIAM Review*, 49(3):434–448.

Del Pia, A., Zhou, D., and Zhu, Y. (2025). Efficient sparse PCA via block diagonalization. In *International Conference on Learning Representations (ICLR)*.

Fan, J. and Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association*, 96(456):1348–1360.

Frost, H. R., Li, Z., and Moore, J. H. (2015). Principal component gene set enrichment (PCGSE). *BioData Mining*, 8:25.

Ho, K. P. and Masuda, B. (2025). Variable selection via thresholding. *Scandinavian Journal of Statistics*, to appear.

Halko, N., Martinsson, P. G., and Tropp, J. A. (2011). Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2):217–288.

Hubert, M., Reynkens, T., Schmitt, E., and Verdonck, T. (2016). Sparse PCA for high-dimensional data with outliers. *Technometrics*, 58(4):424–434.

Holland, P. W. and Welsch, R. E. (1977). Robust regression using iteratively reweighted least-squares. *Communications in Statistics — Theory and Methods*, 6(9):813–827.

Johnstone, I. M. and Lu, A. Y. (2009). On consistency and sparsity for principal components analysis in high dimensions. *Journal of the American Statistical Association*, 104(486):682–693.

Lei, J. and Vu, V. Q. (2015). Sparsistency and agnostic inference in sparse PCA. *Annals of Statistics*, 43(1):299–322.

Loshchilov, I. and Hutter, F. (2019). Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Ma, Z. (2013). Sparse principal component analysis and iterative thresholding. *Annals of Statistics*, 41(2):772–801.

Luecken, M. D., Büttner, M., Chaichoompu, K., Danese, A., Interlandi, M., Müller, M. F., Strobl, D. C., Zappia, L., Duber, M., Kuemmerle, L. B., et al. (2022). Benchmarking atlas-level data integration in single-cell genomics. *Nature Methods*, 19(1):41–50.

Meinshausen, N. and Bühlmann, P. (2010). Stability selection. *Journal of the Royal Statistical Society: Series B*, 72(4):417–473.

Parikh, N. and Boyd, S. (2014). Proximal algorithms. *Foundations and Trends in Optimization*, 1(3):127–239.

Patterson, N., Price, A. L., and Reich, D. (2006). Population structure and eigenanalysis. *PLoS Genetics*, 2(12):e190.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Price, A. L., Patterson, N. J., Plenge, R. M., Weinblatt, M. E., Shadick, N. A., and Reich, D. (2006). Principal components analysis corrects for stratification in genome-wide association studies. *Nature Genetics*, 38(8):904–909.

Qiu, Y., Lei, J., and Roeder, K. (2023). Gradient-based sparse principal component analysis with extensions to online learning. *Biometrika*, 110(2):339–360.

Sikkema, L., Ramírez-Suástegui, C., Strunz, D., Marchetti, A., Weersma, R. K., et al. (2023). An integrated cell atlas of the lung in health and disease. *Nature Medicine*, 29:1563–1577.

Stuart, T., Butler, A., Hoffman, P., Hafemeister, C., Papalexi, E., Mauck, W. M., Hao, Y., Stoeckius, M., Smibert, P., and Satija, R. (2019). Comprehensive integration of single-cell data. *Cell*, 177(7):1888–1902.

Townes, F. W., Hicks, S. C., Aryee, M. J., and Irizarry, R. A. (2019). Feature selection and dimension reduction for single-cell RNA-Seq based on a multinomial model. *Genome Biology*, 20:295.

Wagner, F. (2015). GO-PCA: An unsupervised method to explore gene expression data using prior knowledge. *PLoS One*, 10(11):e0143196.

Witten, D. M., Tibshirani, R., and Hastie, T. (2009). A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis. *Biostatistics*, 10(3):515–534.

Vu, V. Q. and Lei, J. (2013). Minimax sparse principal subspace estimation in high dimensions. *Annals of Statistics*, 41(6):2905–2947.

Vu, V. Q., Cho, J., Lei, J., and Rohe, K. (2013). Fantope projection and selection: A near-optimal convex relaxation of sparse PCA. In *Advances in Neural Information Processing Systems*, 26:2670–2678.

Wolf, F. A., Angerer, P., and Theis, F. J. (2018). SCANPY: large-scale single-cell gene expression data analysis. *Genome Biology*, 19:15.

Zhang, C.-H. (2010). Nearly unbiased variable selection under minimax concave penalty. *Annals of Statistics*, 38(2):894–942.

Zhao, P. and Yu, B. (2006). On model selection consistency of Lasso. *Journal of Machine Learning Research*, 7:2541–2563.

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

Zou, H. (2006). The adaptive lasso and its oracle properties. *Journal of the American Statistical Association*, 101(476):1418–1429.

Zou, H. and Xue, L. (2018). A selective overview of sparse principal component analysis. *Proceedings of the IEEE*, 106(8):1311–1320.

Erichson, N. B., Zheng, P., Manohar, K., Brunton, S. L., Kutz, J. N., and Aravkin, A. Y. (2020). Sparse principal component analysis via variable projection. *SIAM Journal on Applied Mathematics*, 80(2):977–1002.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B*, 58(1):267–288.

Friedman, J., Hastie, T., and Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B*, 67(2):301–320.

Freijeiro-González, L., Febrero-Bande, M., and González-Manteiga, W. (2022). A critical review of LASSO and its derivatives for variable selection under dependence among covariates. *International Statistical Review*, 90(1):118–145.

Natarajan, B. K. (1995). Sparse approximate solutions to linear systems. *SIAM Journal on Computing*, 24(2):227–234.

Chen, S. S., Donoho, D. L., and Saunders, M. A. (1998). Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20(1):33–61.

Pfeiffer, P., Vana-Gür, L., and Filzmoser, P. (2025). Cellwise robust and sparse principal component analysis. *Advances in Data Analysis and Classification*. doi:10.1007/s11634-025-00656-3.

Puchhammer, P., Wilms, I., and Filzmoser, P. (2026). Sparse outlier-robust PCA for multi-source data. *Statistics and Computing*, 36(1):22.

Savitzky, A. and Golay, M. J. E. (1964). Smoothing and differentiation of data by simplified least squares procedures. *Analytical Chemistry*, 36(8):1627–1639.

Rousseeuw, P. J. and Croux, C. (1993). Alternatives to the median absolute deviation. *Journal of the American Statistical Association*, 88(424):1273–1283.

Fang, Z., Liu, X., and Peltz, G. GSEApy: a comprehensive package for performing gene set enrichment analysis in Python. *Bioinformatics*, 39(1):btac757, 2022.

Kuhn, H. W. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1–2):83–97, 1955.

A Method Details

This appendix provides the full method specification referenced from Section 2.

A.1 Scale Normalization and Dimensionless Scaling

Table 2: **Scale normalization and dimensionless scaling of objective terms.** Each term is written as an average over its natural index set. Additional normalizations make terms comparable across datasets and problem sizes.

Term	**What it measures**	**Scale normalization**	**Dimensionless / size-invariant scaling**
Reconstruction $\mathcal{L}_{\text{recon}}(\mathbf{X}^*, \mathbf{W})$	Residual energy on a mini-batch $\mathbf{X}^* \in \mathbb{R}^{b\times p}$, using raw $\mathbf{W}$	Divide by robust variance $\hat{\sigma}^2_{\text{rob}}$	Average per entry $(1/(bp))$
Sparsity $\mathcal{L}_{\text{sparse}}(\mathbf{W})$	Magnitude of individual loading entries, evaluated on a direction-only view of $\mathbf{W}$	Column-direction normalization with gradient-detached norms	Average over parameters $(1/(pk))$; ε set by Otsu's method
Orthogonality $\mathcal{L}_{\text{orth}}(\mathbf{W})$	Correlation between components via Gram matrix of column-normalized loadings $\hat{\mathbf{W}}$	Column normalization $\hat{\mathbf{W}}$ (gradients flow)	Average over Gram entries $(1/k^2)$

Table 3: Scaling mechanisms ensuring each loss term is $O(1)$.

Term	Mechanism	Result
$\mathcal{L}_{\text{recon}}$	Divide by $\hat{\sigma}^2_{\text{rob}}$	Scale-free; $O(1)$ for any data variance
$\mathcal{L}_{\text{sparse}}$	$\log(1 + \hat{W}^2_{j\ell}/\varepsilon^2)$, mean over pk	Bounded on $[0, 1]$ domain; ε set by Otsu
$\mathcal{L}_{\text{orth}}$	$\hat{\mathbf{W}}^\top\hat{\mathbf{W}}$ (unit columns), mean over k^2 entries	Gram entries in $[-1, 1]$; mean $O(1)$

In existing sparse PCA methods, the effective strength of the sparsity parameter changes with p, k, and feature variance, requiring re-tuning for each new dataset. The normalization strategy above eliminates this dependence: a setting of $\lambda_s = 1.0$ means sparsity pressure is calibrated to the same magnitude as reconstruction pressure at PCA-equivalent quality, regardless of whether $p = 10$ or $p = 30{,}000$.

PCA calibration computation. The calibration constant $\mathcal{L}^{\text{PCA}}_{\text{recon}}$ is computed before training begins by evaluating the exact training loss function (including robust sample weighting and $\hat{\sigma}^2_{\text{rob}}$ normalization) at the PCA loading matrix $\mathbf{W}_{\text{PCA}}$. Specifically, $\mathbf{W}_{\text{PCA}}$ is obtained via randomized SVD (Halko et al., 2011) of the centered data (the same routine used for PCA initialization). $\mathcal{L}^{\text{PCA}}_{\text{recon}}$ is then averaged over n_{cal} random minibatches (default $n_{\text{cal}} = 5$, each of size b) to reduce variance:

$$\mathcal{L}^{\text{PCA}}_{\text{recon}} = \frac{1}{n_{\text{cal}}} \sum_{m=1}^{n_{\text{cal}}} \mathcal{L}_{\text{recon}}(\mathbf{X}^{(m)}, \mathbf{W}_{\text{PCA}}), \tag{11}$$

where $\mathbf{X}^{(m)}$ denotes the m-th random minibatch. This is a property of the data and the number of components, not of the initialization strategy. When PCA components are provided externally, they are used directly. When they are not provided, the method computes them internally via randomized SVD. The cost is negligible relative to training (one SVD plus n_{cal} forward passes).

A.2 Robust Reconstruction Details

The robust variance estimate is $\hat{\sigma}^2_{\text{rob}} = \left(\text{median}_j^+(\text{MAD}_j) \times 1.4826\right)^2$, where $\text{MAD}_j = \text{median}_i |x_{ij} - \bar{x}_j|$ is centered around the column mean (not the feature median), and median_j^+ denotes the median over features with $\text{MAD}_j > 0$. The 1.4826 factor is the Fisher consistency correction for Gaussian data.

Two design choices distinguish this estimator from the standard MAD. First, centering uses the column mean $\bar{x}_j$ rather than the per-feature median. This is necessary for conceptual consistency: the loss operates on centered data $\tilde{x}_{ij} = x_{ij} - \bar{x}_j$, so the scale estimate should reflect the spread of $\tilde{x}_{ij}$. For sparse count data (such as scRNA-seq) the feature median is typically zero while the feature mean is not, making the mean-centered MAD strictly more informative. Second, the outer median is restricted to features with $\text{MAD}_j > 0$. In high-dropout data, a majority of features may have $\text{MAD}_j = 0$, so the unrestricted median would also be zero. Restricting to nonzero MADs ensures $\hat{\sigma}^2_{\text{rob}} > 0$ whenever at least one feature is expressed.

Remark 1 (Relationship to M-estimators)**.** *The weighting scheme corresponds to the log-cosh M-estimator ($\rho(t) = \log\cosh t$, $\psi(t) = \tanh t$), which occupies a middle ground between Huber loss (non-redescending; influence plateaus) and Tukey's biweight (hard cutoff; non-differentiable at threshold). The* tanh *influence function is bounded, monotone, and infinitely differentiable, making it uniquely suited to gradient-based optimization.*

A.3 Log Penalty Properties

The log penalty $\phi(x;\varepsilon) = \log(1 + x^2/\varepsilon^2)$ is a fully smooth (C^∞) version of the log penalty $\log(|x| + \varepsilon)$ introduced by Candès et al. (2008). Its gradient structure provides intrinsic adaptive lasso behavior, as described in Section 2.1.3.

Near the origin, the Taylor expansion is $\phi(x) = x^2/\varepsilon^2 - x^4/(2\varepsilon^4) + O(x^6)$, showing that the penalty behaves as a scaled ℓ_2 penalty for small entries. The second derivative is $\phi''(x) = 2(\varepsilon^2 - x^2)/(\varepsilon^2 + x^2)^2$, which is positive for $|x| < \varepsilon$ (locally convex) and negative for $|x| > \varepsilon$ (locally concave). The penalty is therefore convex near the origin and concave in the tails — sharing the concave-penalty structure of SCAD (Fan and Li, 2001) and MCP (Zhang, 2010), but without their non-differentiable points.

A.4 Sparsity Warmup via Cosine Annealing

Applying full sparsity penalty from initialization can trap the optimizer in poor local minima. We gradually increase the sparsity weight using a cosine schedule:

$$\alpha(t) = \frac{1}{2}\left(1 - \cos\left(\frac{\pi \cdot t}{T_{\text{warm}}}\right)\right) \tag{12}$$

where t is the current training step and T_{warm} is the warmup period in steps.

A.5 Orthogonality Penalty Details

We differentiate through the column normalization in $\hat{\mathbf{W}}$ (unlike the sparsity term, we do not detach column norms here). The unnormalized penalty $\|\mathbf{W}^\top\mathbf{W} - \mathbf{I}_k\|_F^2$ would simultaneously penalize deviations in column norm from 1 *and* inter-component correlation, creating a counterproductive tug-of-war with the sparsity penalty.

A.6 Initialization

PCA initialization (default). The initial loading matrix consists of the top-k right singular vectors of the centered data, computed via randomized SVD (Halko et al., 2011). For sparse or large-p data, we use a `LinearOperator` wrapper that applies centering implicitly.

Random orthonormal initialization. When outlier contamination may have corrupted the covariance structure, we provide a covariance-independent alternative: (1) Draw $\mathbf{R} \in \mathbb{R}^{p\times k}$ with i.i.d. $\mathcal{N}(0,1)$ entries; (2) QR-decompose to get orthonormal $\mathbf{Q}$; (3) Scale: $\mathbf{W}^{(0)} = \hat{\sigma}_{\text{rob}} \cdot \mathbf{Q}$.

A.7 Optimization Details

Training is steps-based (not epoch-based). The learning rate follows a two-phase schedule: linear warmup from $\eta_{\max}/100$ to $\eta_{\max}$ over $T_{\text{lr-warm}} = T_{\text{warm}}/2$ steps, then cosine decay to $\eta_{\max}/10$. After the scheduled T steps, training optionally continues at $\eta_{\max}/10$ with patience-based stopping (terminate when neither Gini nor reconstruction improves for P consecutive checkpoints).

Algorithm 1 ROSS-PCA Training Procedure

Require: Data $\mathbf{X} \in \mathbb{R}^{n \times p}$, components k, sparsity weight λ_s, orthogonality weight λ_{orth}, total steps T, warmup steps T_{warm}, ε-update interval T_ε, batch size B, patience P

Preprocessing:
  Compute global mean $\bar{\mathbf{x}} = \frac{1}{n}\sum_i \mathbf{x}_i$
  Compute per-feature MAD: $\text{MAD}_j = \text{median}_i\, |x_{ij} - \bar{x}_j|$
  Compute robust scale $\hat{\sigma}^2_{\text{rob}} = (\text{median}_j^+ \text{MAD}_j \times 1.4826)^2$
  Compute $\mathcal{L}^{\text{PCA}}_{\text{recon}}$ from PCA components
Initialization:
  $\mathbf{W}^{(0)} \leftarrow$ PCA (top-k right singular vectors) *or* random orthonormal $\times\, \hat{\sigma}_{\text{rob}}$
  $\varepsilon \leftarrow 1/\sqrt{p}$
for $t = 1, 2, \ldots$ **do**
  Sample minibatch $\mathbf{X}_b$ of size B from infinite stream
  $\alpha \leftarrow \frac{1}{2}(1 - \cos(\pi t / T_{\text{warm}}))$
  Compute $\nabla_{\mathbf{W}} \mathcal{L}$ via autodifferentiation
  Update $\mathbf{W}$ using AdamW with scheduled learning rate
  if $t \bmod T_\varepsilon = 0$ **and** $t \geq T_{\text{warm}}$ **then**
    $\varepsilon \leftarrow \max\big(\text{Otsu}(|\hat{\mathbf{W}}|),\ 1/\sqrt{p}\big)$
  end if
  if $t = T$ **then**
    Switch to constant LR $= \eta_{\max}/10$; begin patience tracking
  end if
  if $t > T$ **and** no improvement for P checkpoints **then**
    break
  end if
end forreturn $\mathbf{W}^{(t)}$

A.8 Cosine-Preserving Pruning Details

CPP prunes entries in rank order of $|w_i|$ (smallest first). Because w_{cpp} is obtained by zeroing a subset of entries, $\cos(w, w_{\text{cpp}}) = \|w_{\text{cpp}}\|_2 / \|w\|_2$, so the cosine constraint is equivalent to preserving at least $\cos_t^2$ of the squared ℓ_2 energy. We define effective sparsity as:

$$\text{EffSparse}_{\text{cpp}}(W; \cos_t) = \frac{1}{pk} \sum_{j=1}^{p} \sum_{\ell=1}^{k} \mathbf{1}\Big\{ \big(W_{\text{cpp}}(\cos_t)\big)_{j\ell} = 0 \Big\}. \tag{13}$$

Algorithm 2 Cosine-Preserving Pruning (CPP)

Require: Trained loading matrix $\mathbf{W} \in \mathbb{R}^{k \times p}$, cosine threshold $\cos_t \in (0, 1]$

for each component $j = 1, \ldots, k$ **do**
  $E_j \leftarrow \|\mathbf{w}_j\|_2^2$ ▷ total energy
  $\pi_j \leftarrow \text{argsort}(|w_{j,1}|, \ldots, |w_{j,p}|)$ ▷ ascending
  $C_j[i] \leftarrow \sum_{m=1}^{i} w_{j,\pi_j[m]}^2$ for $i = 1, \ldots, p$ ▷ cumulative removed energy
end for

for each component $j = 1, \ldots, k$ **do**
  $E_{\min} \leftarrow \cos_t^2 \cdot E_j$ ▷ energy to preserve
  Find largest i^* s.t. $E_j - C_j[i^*] \geq E_{\min}$ ▷ binary search
  $w_{j,\pi_j[1:i^*]} \leftarrow 0$ ▷ zero the i^* smallest entries
end forreturn Pruned $\mathbf{W}$

Operating point selection. For each model, we select a single operating point from the CPP sparsity–VE frontier. Both axes are normalized to $[0, 1]$ per curve and the VE curve is smoothed via a Savitzky–Golay filter (Savitzky and Golay, 1964). We compute the rate of change of the tangent angle $\vartheta = \arctan(d\,\text{VE}/d\sigma)$ with respect to normalized sparsity σ:

$$\frac{d\vartheta}{d\sigma} = \frac{d^2\text{VE}/d\sigma^2}{1 + (d\,\text{VE}/d\sigma)^2}. \tag{14}$$

The operating point is the sparsity at which $d\vartheta/d\sigma$ is most negative — the sharpest transition from flat to steep on the frontier. For hyperparameter selection across configurations, we apply the same criterion to the frontier of best-per-configuration operating points.

B Simulation Details

B.1 Simulation Designs

Sparse PCA serves two distinct purposes: dimensionality reduction and feature selection. These are not the same objective. Moreover, individual sparse components may not be identifiable when leading eigenvalues are close (Ma, 2013), and whether the "correct" set of active features is well-defined depends on identifiability conditions (Lei and Vu, 2015). We organize the evaluation around three questions: (1) sparsity–reconstruction tradeoff, (2) feature identification, and (3) stability across initializations.

Spiked covariance model. The spiked covariance model and the generating-basis terminology are introduced in Section 3.1 (Eq. 10). We elaborate here on the relationship between the generating basis $\mathbf{V}$ and the eigenbasis of Σ.

When the columns of $\mathbf{V}$ have disjoint support (S1, S2), $\mathbf{V}^\top\mathbf{V} = \mathbf{I}_k$ and the generating basis coincides with an eigenbasis of Σ. When columns overlap (S3, S4), $\mathbf{V}^\top\mathbf{V} \neq \mathbf{I}_k$, so $\mathbf{V}\mathbf{D}\mathbf{V}^\top$ is not an eigendecomposition and the eigenvectors of Σ are rotations of the columns of $\mathbf{V}$. In either case, the column space of $\mathbf{V}$ equals the top-k eigenspace of Σ.

This distinction clarifies which comparisons are well-posed under each identifiability regime. The column space of $\mathbf{V}$ and the generating support are always meaningful targets. Comparisons against individual generating components are informative only when the generating basis is identifiable: under eigenvalue degeneracy (S2), any rotation within the eigenspace is equally optimal, so different bases may have different per-column support patterns; under overlap (S3, S4), the generating components are not eigenvectors of Σ.

Crossing support overlap (0% vs. 25%) with eigenvalue separation gives four configurations (S1–S4). Details on the overlap structure, outlier contamination, and replication strategy are as described in the main text.

B.2 Evaluation Metrics

We evaluate each method on both model-free metrics (sparsity, variance explained, orthogonality) and metrics that compare against the generating basis (global AUPRC, per-column AUPRC). The AUPRC metrics require precise specification because they involve choices about scoring functions and component matching.

Global AUPRC. Global AUPRC measures whether the method identifies the correct set of active features, without regard to which component each feature belongs to. Given estimated loadings $\hat{\mathbf{W}} \in \mathbb{R}^{p \times k}$ and a binary indicator $y_j \in \{0, 1\}$ for whether feature j is in the generating support:

1. Compute the feature importance score as the row norm of the estimated loading matrix: $s_j = \|\hat{\mathbf{w}}_{j\cdot}\|_2 = \sqrt{\sum_{\ell=1}^{k} \hat{W}_{j\ell}^2}$ for $j = 1, \ldots, p$.
2. Compute the area under the precision–recall curve (AUPRC) using s_j as the predicted score and y_j as the binary label, via `sklearn.metrics.average_precision_score`.

Global AUPRC = 1.0 means the row norms perfectly separate active from inactive features at some threshold. It is invariant to component permutation and does not require matching estimated components to generating components.

Per-column AUPRC. Per-column AUPRC measures whether each estimated component loads on the correct features, requiring a correspondence between estimated and generating components. Because the component ordering is arbitrary, we establish this correspondence via optimal assignment (Hungarian algorithm) on AUPRC scores.

Given estimated loadings $\hat{\mathbf{W}} \in \mathbb{R}^{p \times k}$ and per-column generating supports $S_1, \ldots, S_{k_{\text{gen}}}$ (each S_j is a set of feature indices):

1. Compute the $k \times k_{\text{gen}}$ AUPRC matrix A, where entry $A_{i,j}$ is the AUPRC obtained by using $|\hat{W}_{\cdot,i}|$ (absolute values of the i-th estimated column) as the predicted score and the binary indicator of S_j as the label.
2. Match estimated components to generating components via optimal assignment: solve for the permutation π that maximizes $\sum_i A_{i,\pi(i)}$ using the Hungarian algorithm (`scipy.optimize.linear_sum_assignment`).
3. Per-column AUPRC is the mean of the $\min(k, k_{\text{gen}})$ matched AUPRC values.

Per-column AUPRC = 1.0 means each estimated component perfectly separates the features in its matched generating support from all other features. Unlike global AUPRC, it is sensitive to whether the method assigns features to the *correct* generating component, not just whether it identifies them as active. This metric is most informative when the generating basis is identifiable (S1, S1c). Under eigenvalue degeneracy (S2) or overlapping support (S3, S4), per-column AUPRC conflates identifiability limitations with estimation quality; see Section 3.1 for discussion.

Subspace angles. Principal angles between the column spaces of the estimated loadings and the generating basis measure subspace recovery independently of basis choice. Given $\hat{\mathbf{W}} \in \mathbb{R}^{p \times k}$ and $\mathbf{V} \in \mathbb{R}^{p \times k}$, the k principal angles $\theta_1 \leq \cdots \leq \theta_k$ are computed via `scipy.linalg.subspace_angles`. We report $\overline{\sin\theta} = \frac{1}{k}\sum_{\ell=1}^{k} \sin\theta_\ell$, which equals 0 when the subspaces are identical and approaches 1 for maximally misaligned subspaces. Because subspace angles depend only on the column spans of $\hat{\mathbf{W}}$ and $\mathbf{V}$, they are invariant to rotation within each subspace. This makes them the appropriate recovery metric under eigenvalue degeneracy (S2), where the generating basis is non-identifiable but its column space is.

Orthogonality. We measure inter-component correlation via the mean absolute off-diagonal entry of the direction-normalized Gram matrix. Given $\hat{\mathbf{W}} \in \mathbb{R}^{p \times k}$:

1. Normalize each column to unit norm: $\tilde{\mathbf{w}}_\ell = \hat{\mathbf{w}}_\ell / \|\hat{\mathbf{w}}_\ell\|_2$.
2. Compute the Gram matrix $G = \tilde{\mathbf{W}}^\top \tilde{\mathbf{W}}$.
3. Orthogonality $= \frac{1}{k(k-1)} \sum_{i \neq j} |G_{ij}|$.

Orthogonality $= 0$ for perfectly orthogonal components; it approaches 1 for identical components. This metric is computed on the direction-normalized loadings, so it measures angular relationships between components independent of column magnitudes. It is consistent across methods regardless of how zeros were produced (ℓ_1 thresholding, CPP pruning, or native).

Additional metrics. *Variance explained:* $\mathrm{VE} = 1 - \|\tilde{\mathbf{X}} - \tilde{\mathbf{X}}\hat{\mathbf{W}}\hat{\mathbf{W}}^\top\|_F^2 / \|\tilde{\mathbf{X}}\|_F^2$, where $\tilde{\mathbf{X}}$ is centered data. *Sparsity:* fraction of exactly zero entries in $\hat{\mathbf{W}}$. *Relative reconstruction error:* $\|\tilde{\mathbf{X}} - \tilde{\mathbf{X}}\hat{\mathbf{W}}\hat{\mathbf{W}}^\top\|_F / \|\tilde{\mathbf{X}}\|_F$.

B.3 Baselines and Configuration

We compare against four sparse PCA methods plus standard PCA, all with PCA initialization.

Standard PCA. Randomized SVD (Halko et al., 2011). 1 configuration.

Zou–Hastie SparsePCA (Zou and Hastie, 2005). $\alpha \in \{0.01, 0.05, 0.1, 0.2, 0.5, 1.0, 2.0, 5.0\}$, ridge $= 0.01$. 8 configurations.

MiniBatch SparsePCA (Pedregosa et al., 2011). Same α grid. 8 configurations.

VarPro SPCA (Erichson et al., 2020). $\alpha \in \{10^{-4}, \ldots, 25\}$, $\beta = 0.001$. 13 configurations.

PMD/SPC (Witten et al., 2009). $\|v\|_1 \leq c$, $c = 1 + f(\sqrt{p} - 1)$ with $f \in \{0.05, \ldots, 0.8\}$. 8 configurations.

ROSS-PCA. $\lambda_s \in \{0, 0.001, 0.01, 0.1, 0.5, 1.0\} \times \lambda_{\text{orth}} \in \{0, 0.001, 0.01, 0.1, 0.5, 1.0\}$. 36 configurations. Fixed parameters in Table 4.

Two-level operating point selection. Each method sweeps a grid of hyperparameter configurations, and each configuration produces a model at a particular point on the sparsity–reconstruction frontier. Selecting a single representative result per method and replicate requires two levels of selection for ROSS-PCA and one level for baselines.

Level 1 (within-model, ROSS-PCA only). Each trained ROSS-PCA model is followed by a 100-point CPP sweep over $\theta \in [0°, 10°]$, tracing a sparsity–VE frontier. The operating point is selected via the $d\vartheta/d\sigma$ criterion (Section 2.2), which identifies the knee of the (sparsity, variance explained) curve. This produces one (sparsity, reconstruction error) pair per hyperparameter configuration. For baselines, each configuration directly produces native sparsity with no sweep.

Level 2 (across configurations, all methods). Given the set of (sparsity, relative reconstruction error) pairs across all hyperparameter configurations for a method, we select the best configuration via the chord-elbow method on the (sparsity, relative reconstruction error) frontier. Both axes are normalized to $[0, 1]$, points are sorted by sparsity, and the selected configuration is the one with maximum perpendicular distance from the chord connecting the least-sparse and most-sparse points. This selects the configuration that achieves the best tradeoff between sparsity and reconstruction quality.

C Per-Configuration Results

This appendix provides the full VE–sparsity and recovery figures for each factorial configuration, supplementing the synthesized results in Section 3.1.

Table 4: Factorial simulation design. The spiked covariance model (Johnstone and Lu, 2009) with $p = 500$, $k = 5$, $n = 1{,}000$, $s = 30$ nonzero entries per component. Support overlap and eigenvalue separation are crossed to produce four structural configurations (S1–S4), each evaluated at three SNR levels with 100 replicates per cell.

	Disjoint support (0%)		Overlapping support (25%)	
	Separated eig.	Similar eig.	Separated eig.	Similar eig.
Configuration	S1	S2	S3	S4
Spike eigenvalues	$10 \to 5$	all $= 7.5$	$10 \to 5$	all $= 7.5$
Identifiability	Full	Eig. degeneracy	Support ambiguity	Max. ambiguous
SNR levels (crossed with each configuration)				
Low SNR	$\sigma^2 = 0.675$ (VE $\approx 10\%$)			
Medium SNR	$\sigma^2 = 0.075$ (VE $\approx 50\%$)			
High SNR	$\sigma^2 = 0.0083$ (VE $\approx 90\%$)			
Methods and hyperparameter grids (each run on all 1,200 datasets)				
Standard PCA	Randomized SVD, 1 configuration			
Zou–Hastie	$\alpha \in \{0.01, 0.05, 0.1, 0.2, 0.5, 1.0, 2.0, 5.0\}$, ridge $= 0.01$ (8 configs)			
MiniBatch	$\alpha \in \{0.01, 0.05, 0.1, 0.2, 0.5, 1.0, 2.0, 5.0\}$, ridge $= 0.01$ (8 configs)			
VarPro	$\alpha \in \{10^{-4}, \ldots, 25\}$, $\beta = 0.001$ (13 configs)			
PMD/SPC	$\|v\|_1 \leq 1 + f(\sqrt{p} - 1)$, $f \in \{0.05, \ldots, 0.8\}$ (8 configs)			
ROSS-PCA	$\lambda_s \in \{0, 0.001, 0.01, 0.1, 0.5, 1\} \times \lambda_{\text{orth}} \in \{0, 0.001, 0.01, 0.1, 0.5, 1\}$ (36 configs)			
Datasets	4 configs $\times$ 3 SNR $\times$ 100 replicates $= 1{,}200$			
Total fits	$1{,}200 \times 74$ method configs $= 88{,}800$			

ROSS-PCA fixed parameters: PCA initialization, $T = 3{,}000$ steps, $T_{\text{warm}} = 1{,}000$, $\eta_{\max} = 0.01$, $b = 1{,}000$, patience $= 5$. Each ROSS-PCA model is evaluated on a 100-point CPP sweep ($\theta \in [0^\circ, 10^\circ]$).

Table 5: Outlier contamination experiment. S1c crosses contamination fraction $\{2\%, 5\%\}$ with outlier magnitude $\{4\times, 8\times\}$ the data's standard deviation, applied to S1 (disjoint support, separated eigenvalues). This isolates the contribution of robust sample weighting from overlap and degeneracy effects.

Parameter	Value
Base structure	S1: disjoint support, separated eigenvalues
Dimensions	$p = 500$, $k = 5$, $n = 1{,}000$, $s = 30$
Contamination fraction	$\{2\%, 5\%\}$ of samples replaced
Outlier magnitude	$\{4\times, 8\times\}$ the data's std
SNR levels	Low, Medium, High (same σ^2 as factorial)
Replicates	100 per configuration
Methods	Same 6 methods and grids as factorial (74 configs)
Datasets	4 configs $\times$ 3 SNR $\times$ 100 replicates $= 1{,}200$
Total fits	$1{,}200 \times 74 = 88{,}800$

S1: Disjoint support, separated eigenvalues (Figure 2, 7). With disjoint support and separated eigenvalues, all methods perform well. At medium and high SNR, Zou–Hastie, VarPro, and ROSS-PCA all achieve $> 90\%$ sparsity with variance explained within 1% of PCA. ROSS-PCA achieves per-column AUPRC 0.927 at high SNR with orthogonality 0.003. MiniBatch SparsePCA achieves comparable sparsity–VE tradeoffs but with elevated orthogonality (0.075, versus ≤ 0.01 for all other methods), indicating that its components are poorly separated despite adequate reconstruction and sparsity.

S2: Eigenvalue degeneracy (Figures 8, 9). With degenerate eigenvalues, the component basis is non-identifiable: any rotation within the eigenspace is equally optimal. Subspace angles confirm that ROSS-PCA recovers the subspace as well as Zou–Hastie (mean $\sin\theta = 0.018$ vs. 0.017 at high SNR). The per-column AUPRC gap (0.883 vs. 0.978) reflects basis rotation, not estimation failure. ROSS-PCA maintains the lowest orthogonality among high-sparsity methods (0.002).

Table 6: Identifiability analysis across random initializations. ROSS-PCA is run with 100 independent random orthonormal initializations per replicate at fixed hyperparameters, across all four factorial configurations to provide identifiable controls (S1, S2) and overlap conditions (S3, S4).

Parameter	Value
Configurations	S1 (disjoint, separated), S2 (disjoint, similar), S3 (overlap, separated), S4 (overlap, similar)
SNR levels	Low, Medium, High
Replicates	100 per configuration × SNR
Initializations	100 random orthonormal seeds per replicate
Fixed hyperparameters	$\lambda_s = 1.0$, $\lambda_{\text{orth}} = 1.0$
Training	$T = 3{,}000$ steps, $T_{\text{warm}} = 1{,}000$, $\eta_{\max} = 0.01$, $b = 1{,}000$, patience $= 5$
Datasets	4 configs × 3 SNR × 100 replicates = 1,200
Total fits	1,200 × 100 seeds = 120,000

Each of the 120,000 models is evaluated on a 100-point CPP sweep. Identifiability metrics: Q1–Q4 (Section 3.2).

Table 7: Feature identifiability across SNR levels (mean ± std over 100 replicates, 100 random initializations each). Global AUPRC measures generating support recovery. Q2 and Q3 measure per-feature assignment consistency. Pairwise false co-assignment (Q4 not) increases with SNR under overlap, as stronger signal makes shared structure more visible.

Config	SNR	Global AUPRC	Q1 (active)	Q2 (private)	Q3 (shared)	Q4 co (same)	Q4 not (diff)
S1	Low	$.849 \pm .052$	$.784 \pm .074$	$.780 \pm .073$	—	$.620 \pm .099$	$.002 \pm .004$
	Med	$.927 \pm .021$	$.896 \pm .030$	$.892 \pm .029$	—	$.801 \pm .053$	$.002 \pm .003$
	High	$.966 \pm .014$	$.952 \pm .020$	$.948 \pm .020$	—	$.906 \pm .037$	$.002 \pm .003$
S2	Low	$.868 \pm .046$	$.812 \pm .066$	$.803 \pm .061$	—	$.661 \pm .096$	$.005 \pm .008$
	Med	$.955 \pm .019$	$.935 \pm .027$	$.930 \pm .025$	—	$.874 \pm .050$	$.003 \pm .003$
	High	$.979 \pm .015$	$.969 \pm .021$	$.963 \pm .020$	—	$.940 \pm .041$	$.003 \pm .006$
S3	Low	$.872 \pm .057$	$.833 \pm .075$	$.741 \pm .082$	$.609 \pm .131$	$.628 \pm .103$	$.022 \pm .021$
	Med	$.939 \pm .023$	$.921 \pm .030$	$.792 \pm .075$	$.803 \pm .080$	$.804 \pm .058$	$.059 \pm .055$
	High	$.966 \pm .019$	$.956 \pm .025$	$.710 \pm .115$	$.893 \pm .073$	$.890 \pm .054$	$.144 \pm .089$
S4	Low	$.882 \pm .042$	$.847 \pm .055$	$.739 \pm .069$	$.636 \pm .119$	$.650 \pm .095$	$.032 \pm .030$
	Med	$.951 \pm .021$	$.936 \pm .027$	$.776 \pm .082$	$.841 \pm .089$	$.841 \pm .059$	$.081 \pm .057$
	High	$.964 \pm .021$	$.953 \pm .027$	$.720 \pm .121$	$.892 \pm .066$	$.885 \pm .055$	$.138 \pm .107$

S3: Overlapping support (Figures 10, 11). ROSS-PCA achieves VE = 0.901 at high SNR (99.9% of PCA), compared to 0.888 for Zou–Hastie. Zou–Hastie's orthogonality is 0.033 (absent in S1/S2), while ROSS-PCA maintains 0.007–0.013. VarPro's per-column AUPRC collapses from 0.907 (medium SNR) to 0.750 (high SNR), likely due to its variable-projection formulation struggling with shared features at high signal strength.

S4: Maximally ambiguous (Figures 12, 13). ROSS-PCA achieves VE = 0.901 at high SNR with 91.6% sparsity, global AUPRC 0.978, and orthogonality 0.008. Zou–Hastie achieves higher per-column AUPRC (0.967) but with 4× worse orthogonality (0.033).

S1c: Outlier contamination (Figures 3, 14–17). The contamination study crosses fraction $\{2\%, 5\%\}$ with magnitude $\{4\times, 8\times\}$. At mild contamination (2% at 4×), all methods achieve global AUPRC > 0.93 at medium and high SNR. At the most severe condition (5% at 8×), ROSS-PCA achieves 0.71 at low SNR while baselines fall below 0.32; at medium SNR, ROSS-PCA achieves 0.89 versus the best baseline at 0.82 (PMD/SPC). The advantage emerges progressively as contamination severity increases.

Table 8: Computational scaling experiments. Problem size increases jointly in p, n, and k with $s = 30$ held fixed, so per-component sparsity structure is constant across the sweep. The ratio $n/p = 10$ is fixed throughout. CPU-bound methods (Zou–Hastie, MiniBatch, VarPro, PMD) are skipped where computationally infeasible. Structure: S1 (disjoint, separated), $\sigma^2 = 1.0$, 1 replicate per size.

p	n	k	s	Methods run	Configs
500	5,000	10	30	All 6	74
1,000	10,000	20	30	All 6	74
2,000	20,000	50	30	All 6	74
5,000	50,000	100	30	All 6	74
10,000	100,000	200	30	All 6	74
15,000	150,000	300	30	All 6	74
Total fits					444

Same hyperparameter grids as factorial. Wall-clock time and peak memory recorded for each run.

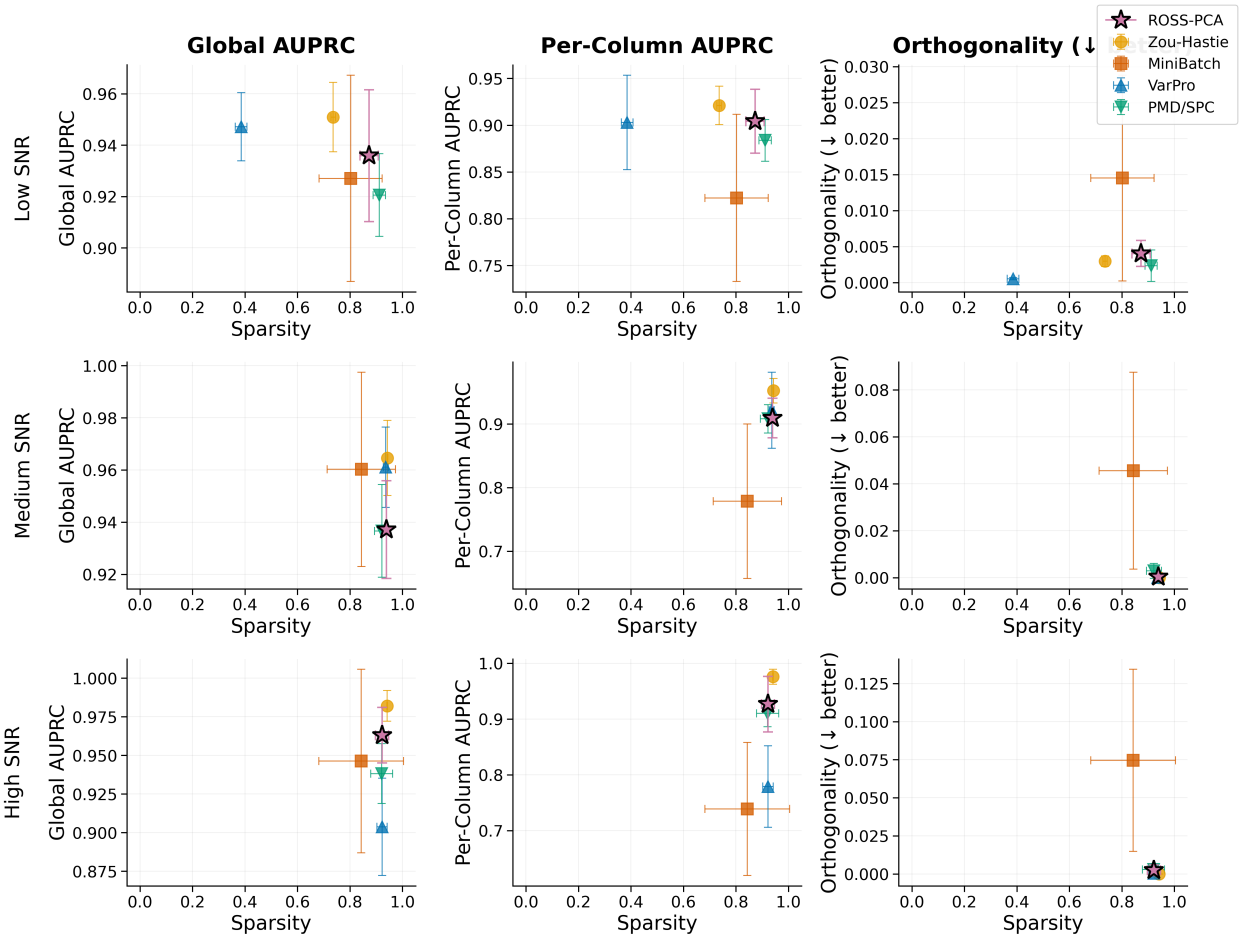

Figure 7: Feature recovery metrics for S1. Per-column AUPRC is the appropriate secondary metric (basis identifiable).

D Application Details: Human Lung Cell Atlas

Data. The HLCA core (Sikkema et al., 2023) contains 584,944 cells and 27,402 genes after standard preprocessing (log-normalization). The data matrix is stored as a sparse CSR matrix (∼3 GB; 1.14 billion nonzero entries). The dense representation would require ∼64 GB in float32.

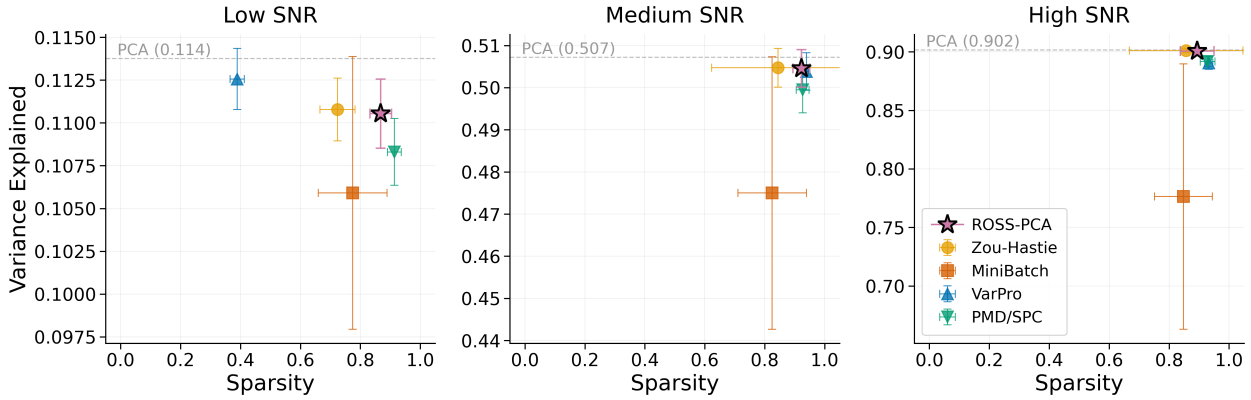

Figure 8: Sparsity vs. variance explained for S2.

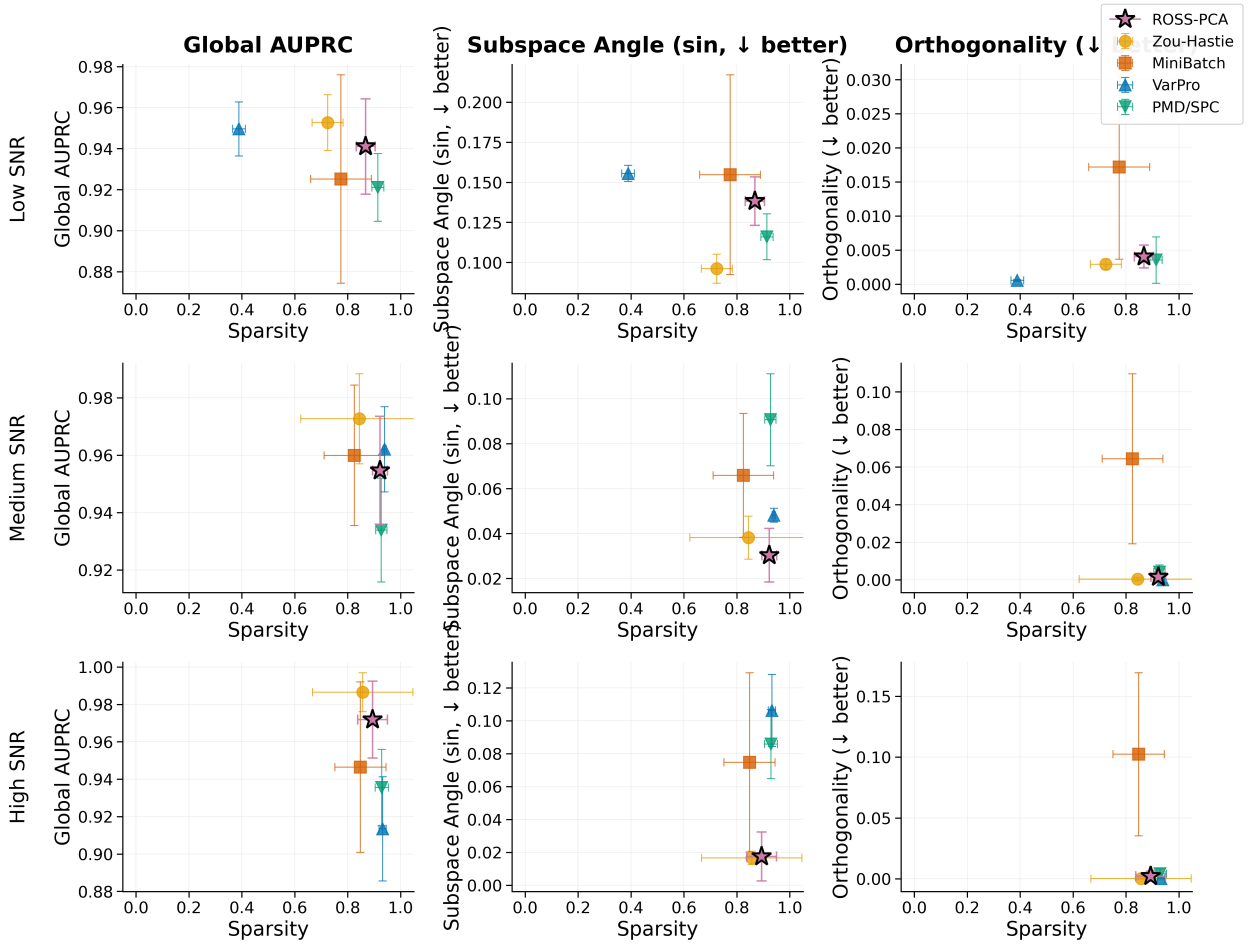

Figure 9: Feature recovery metrics for S2. Subspace angle (center) is the appropriate secondary metric (basis non-identifiable under eigenvalue degeneracy).

PCA. GPU-accelerated PCA ($k = 5{,}000$) is computed via rapids-singlecell on the sparse matrix. This provides the calibration constant $\mathcal{L}_{\text{recon}}^{\text{PCA}}$ and the variance explained reference (∼81% at $k = 5{,}000$).

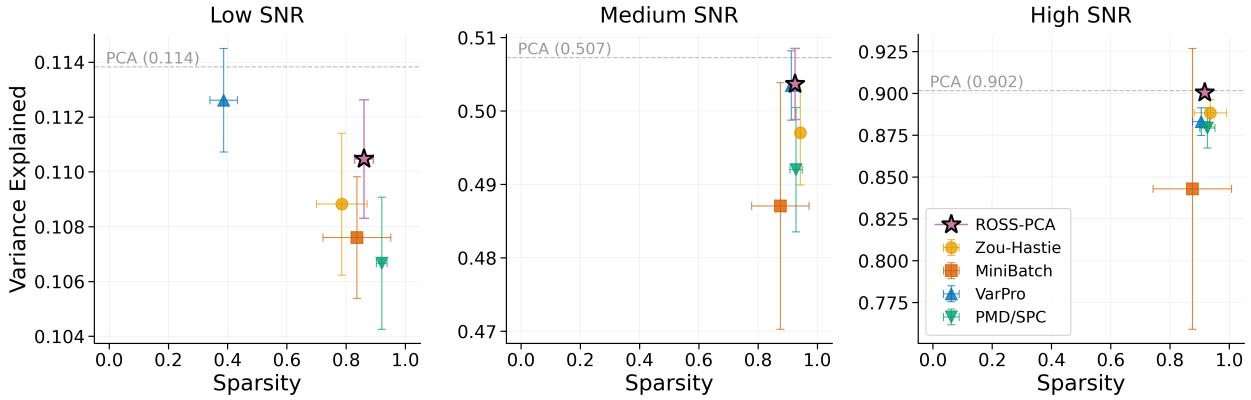

Figure 10: Sparsity vs. variance explained for S3.

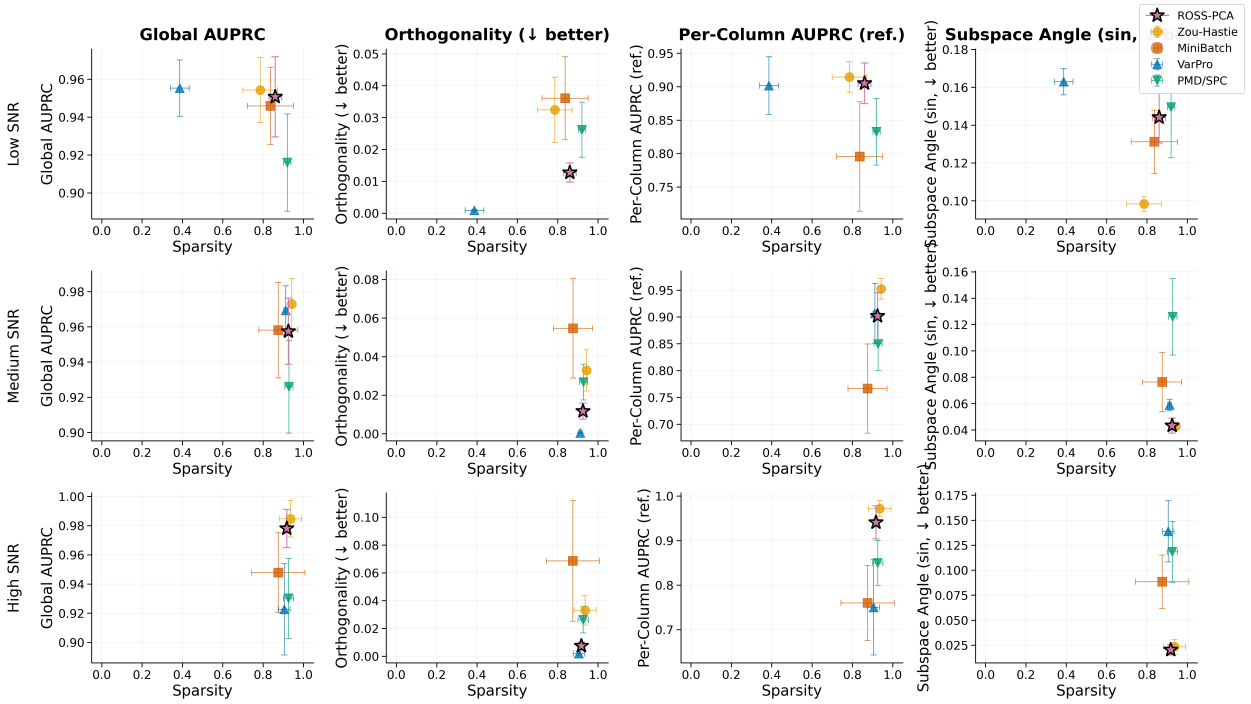

Figure 11: Feature recovery metrics for S3. Orthogonality (center) is the key diagnostic under overlapping support.

Baselines. All existing sparse PCA implementations (Zou–Hastie, MiniBatch, VarPro, PMD) require the dense data matrix in memory. At $584{,}944 \times 27{,}402$ in float64, this is $\sim$128 GB — exceeding typical single-machine RAM. ROSS-PCA operates directly on the sparse representation, densifying only one minibatch at a time ($10{,}000 \times 27{,}402 \times 8 \approx 2.2$ GB).

Training configuration. $\lambda_s = 0.5$, $\lambda_{\text{orth}} = 0.5$, $T = 3{,}000$ steps, $T_{\text{warm}} = 1{,}000$, $\eta_{\max} = 5 \times 10^{-4}$, batch size $= 10{,}000$, patience $= 5$. We run 100 independent random orthonormal initializations at $k = 5{,}000$.

Stability analysis pipeline. Each of the 100 models is followed by a 100-point CPP sweep ($\theta_{\max} = 10°$, float32 precision). The operating point is selected via the $d\theta/ds$ criterion (Section 2.2). Pruned loadings are saved as sparse CSR matrices.

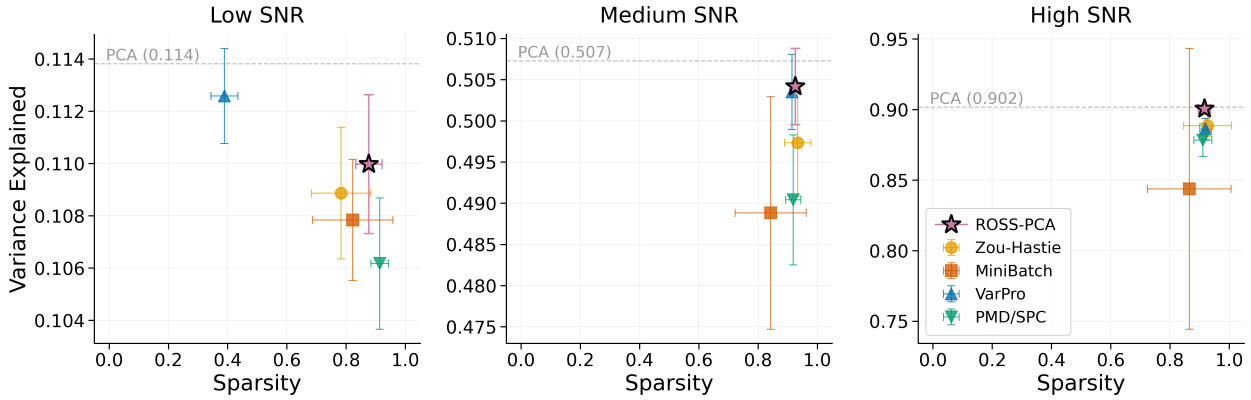

Figure 12: Sparsity vs. variance explained for S4.

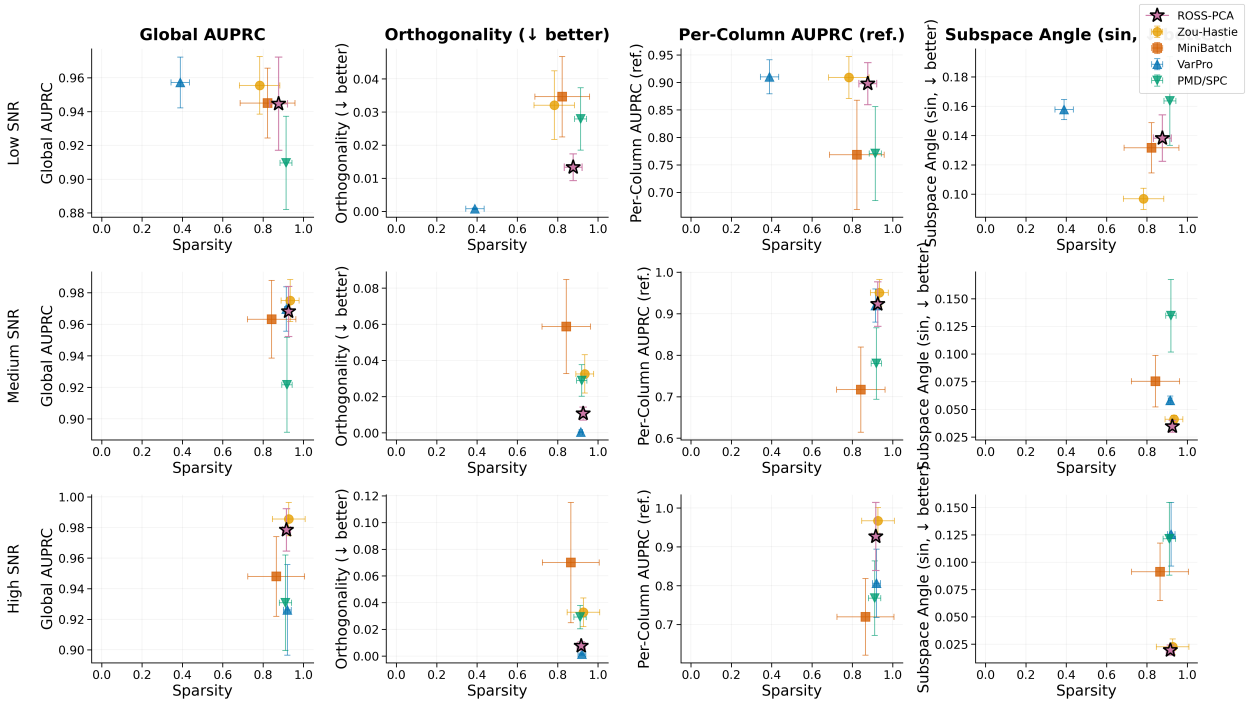

Figure 13: Feature recovery metrics for S4.

Consensus core computation. Given 100 pruned loading matrices $\mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(100)}$, each of shape (k, p) with $k = 5{,}000$ and $p = 27{,}402$, we compute consensus cores as follows.

Step 1: Binarize. For each run r, construct a binary matrix $B^{(r)} \in \{0,1\}^{k \times p}$ where $B^{(r)}_{jg} = 1$ if gene g has a nonzero loading in component j.

Step 2: Jaccard matching. Select one run as the reference (run $r = 0$). For each other run r', compute the $k \times k$ Jaccard similarity matrix between all pairs of components: $J_{ij} = |S^{(0)}_i \cap S^{(r')}_j| \,/\, |S^{(0)}_i \cup S^{(r')}_j|$, where $S^{(r)}_j$ is the support (nonzero gene set) of component j in run r. This is computed efficiently via GPU matrix multiplication: $B^{(0)} {B^{(r')}}^{\top}$ gives intersection sizes, from which Jaccard follows. Apply the Hungarian algorithm (Kuhn, 1955) to $-J$ to obtain an optimal one-to-one assignment $\sigma_{r'} : [k] \to [k]$ maximizing total Jaccard. This ensures no two reference components map to the same target, preventing inflated overlap scores.

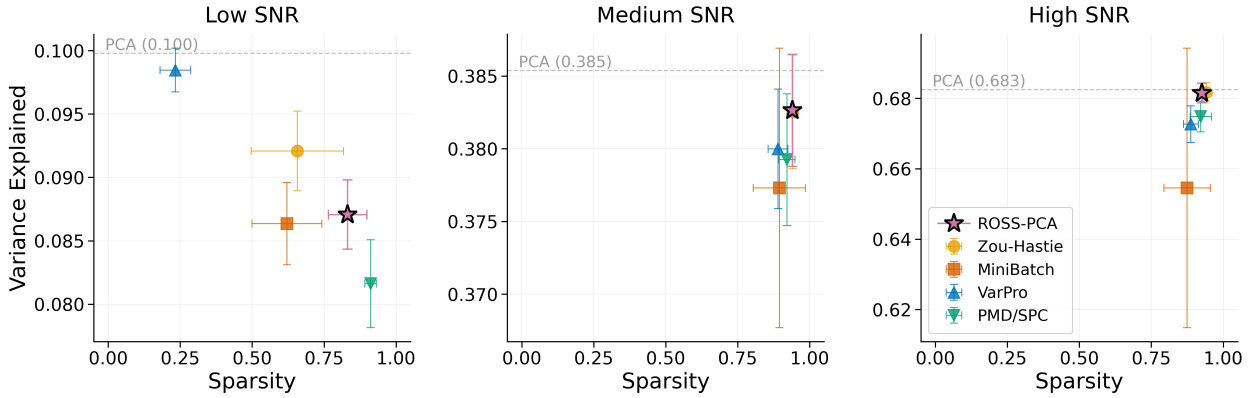

Figure 14: Sparsity vs. variance explained: S1c, 2% at 4×.

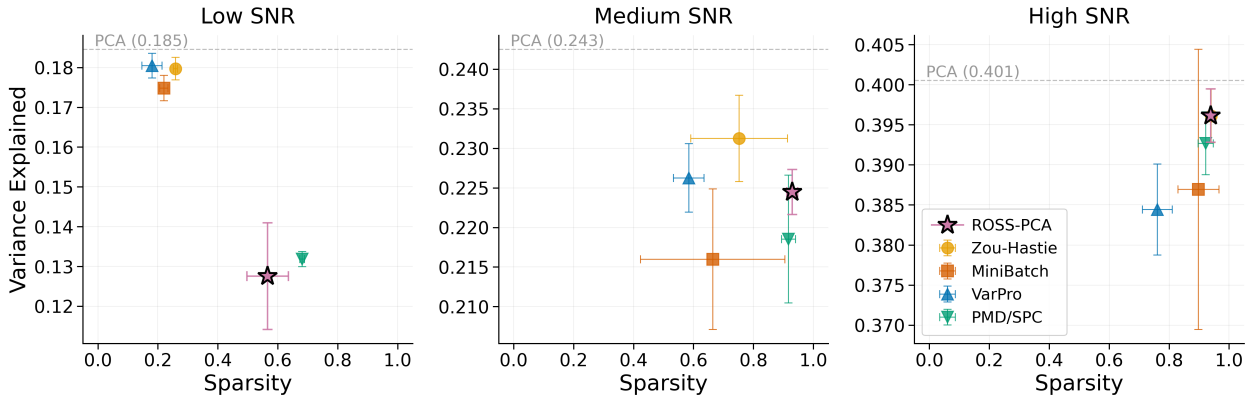

Figure 15: Sparsity vs. variance explained: S1c, 2% at 8×.

Step 3: Gene frequency. For each reference component j and each matched partner $\sigma_{r'}(j)$, record which genes are active. The *gene frequency* f_{jg} is the fraction of the 86 matched partners in which gene g is active: $f_{jg} = \frac{1}{86} \sum_{r' \neq 0} B^{(r')}_{\sigma_{r'}(j),\, g}$.

Step 4: Consensus core. The consensus core of component j at stability threshold τ is $\text{core}_\tau(j) = \{g \in S_j^{(0)} : f_{jg} \geq \tau\}$ — the genes in the reference component that appear in at least τ fraction of the matched partners. The *stable support* at threshold τ is the union $\text{support}_\tau = \bigcup_j \text{core}_\tau(j)$.

Step 5: Variance explained. To assess how much signal the stable support carries, we zero out all rows of $\mathbf{W}$ not in support_τ and recompute the reconstruction error $\|\mathbf{X} - \mathbf{X}\mathbf{W}^\top\mathbf{W}\|_F^2 / \|\mathbf{X} - \boldsymbol{\mu}\|_F^2$ via batched GPU computation on the sparse data matrix.

Functional enrichment. Gene identifiers (Ensembl IDs) are mapped to gene symbols via BioMart. Over-representation analysis uses Fisher's exact test (GSEApy; Fang et al. 2022) against GO Biological Process 2023, KEGG 2021 Human, and Reactome 2022. Background is the set of 14,990 active genes with valid symbol mappings.

E Stability Analysis Details

The identifiability analysis (Section 3.2) runs 100 random orthonormal initializations per replicate across all four factorial configurations at three SNR levels (120,000 total fits). Global AUPRC and Q2–Q4 metrics reported in the main text are aggregated over 100 replicates (mean ± std). The full SNR breakdown including global AUPRC is in Table 7.

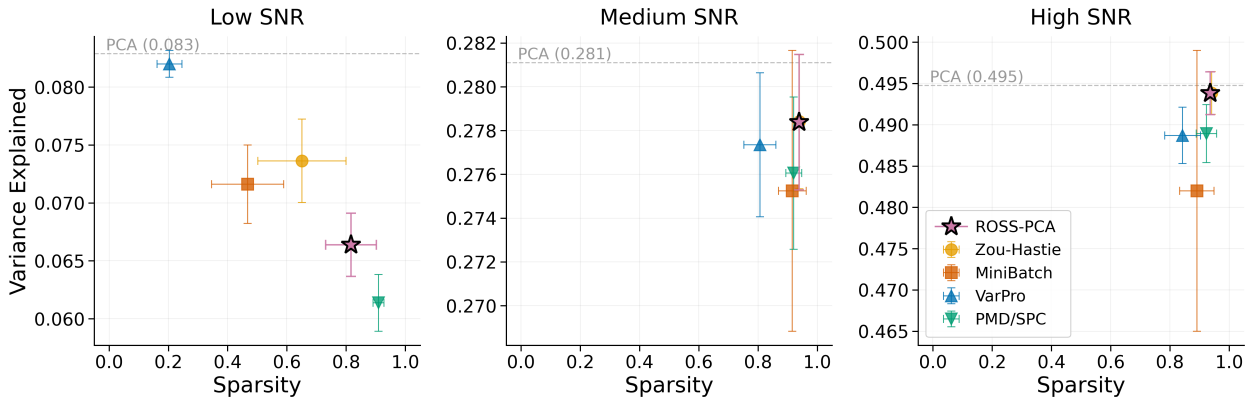

Figure 16: Sparsity vs. variance explained: S1c, 5% at 4×.

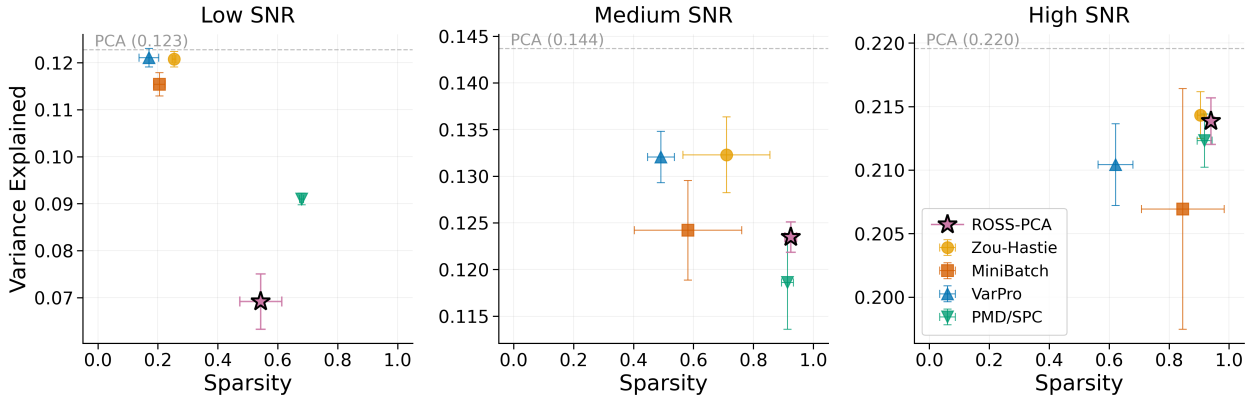

Figure 17: Sparsity vs. variance explained: S1c, 5% at 8×.

F Extended Discussion

Smooth penalties and correlated features. With a smooth penalty, no coefficient is set exactly to zero during optimization. All coefficients evolve simultaneously along continuous gradients, which may allow correlated features to settle at similar magnitudes rather than being arbitrarily partitioned. This hypothesis is consistent with the stability results (Section 3.2), but a direct test on block-correlated features is left for future work.

Relationship to robust sparse PCA methods. Recent methods combining robustness with sparsity (Croux et al., 2013; Hubert et al., 2016; Pfeiffer et al., 2025; Puchhammer et al., 2026) operate on the full data matrix without minibatch decomposition, restricting applicability to $p \leq 1{,}000$. ROSS-PCA integrates robustness directly into a formulation that scales to dimensions two orders of magnitude beyond existing robust methods.

Relationship to the combinatorial literature. A distinct tradition formulates sparse PCA as a cardinality-constrained optimization on the $p \times p$ covariance matrix (d'Aspremont et al., 2007; Berk and Bertsimas, 2019; Bertsimas et al., 2022; Del Pia et al., 2025). These methods require $O(p^2)$ storage and address a single sparse direction at a time. The Fantope relaxation (Vu et al., 2013) and its stochastic variant (Qiu et al., 2023) are closest: both use minibatch gradient descent for genomics, but optimize a $p \times p$ PSD matrix ($O(p^3)$ per step), while ROSS-PCA optimizes the $p \times k$ loading matrix ($O(bpk)$ per step). At $p = 27{,}000$, the $O(p^3)$ projection becomes prohibitive.

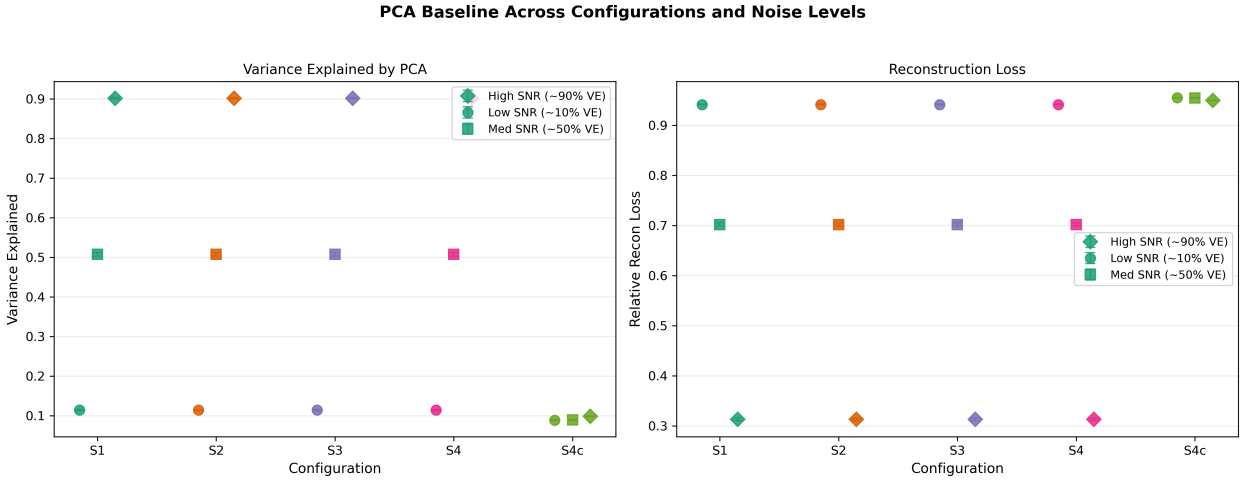

Figure 18: PCA baseline variance explained and reconstruction loss across all structural configurations and noise levels.

Table 9: Variance explained by stable gene supports at different stability thresholds. Support-τ: union of all genes appearing in $\geq \tau\%$ of matched components across 99 other runs. Full model: all 15,296 active genes.

Threshold (τ)	Support size	VE (%)	% of full VE
0% (full)	15,296	81.05	100.0
50%	11,670	81.01	100.0
70%	9,898	80.87	99.8
80%	8,716	80.65	99.5
90%	7,242	80.01	98.7
95%	2,858	46.91	57.9
100%	662	9.86	12.2

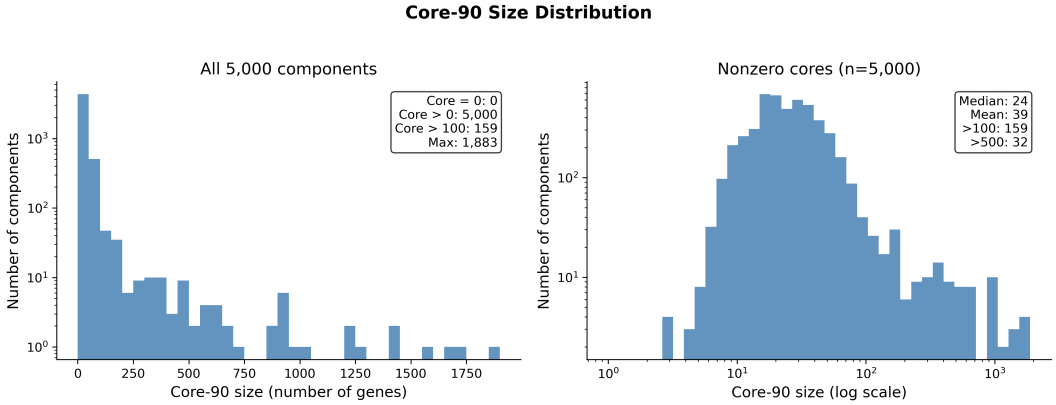

Figure 19: Distribution of consensus core sizes at the 90% stability threshold across 5,000 components. Left: all components (log y-axis). Right: nonzero cores only (log-log). Median core: 24 genes; 159 components have cores exceeding 100 genes; largest core: 1,883 genes.

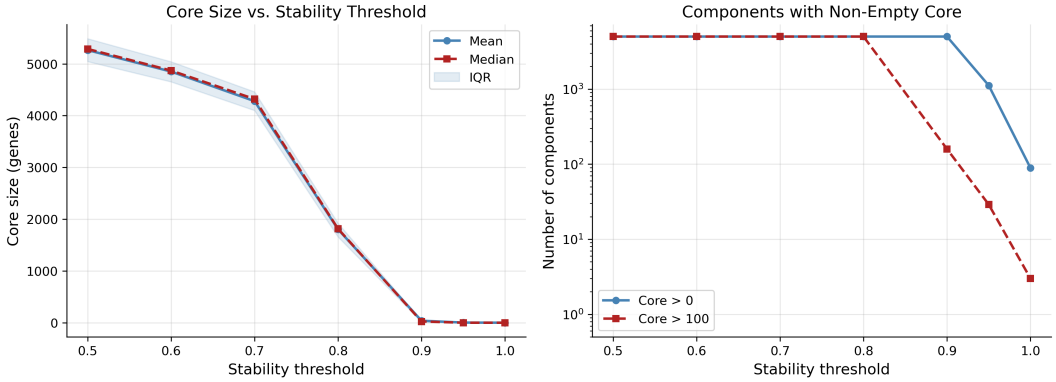

Figure 20: Left: mean and median consensus core size as a function of stability threshold. Core sizes decline gradually from 80% to 90% and drop sharply above 90%. Right: number of components with non-empty cores (blue) and cores exceeding 100 genes (red).