

LoomSC: Scalable Deep Subspace Clustering with Projector Factorization and Exact Spectral Reduction

Nairouz Mrabah, Youssef Melki, Mohamed Bouguessa, Riadh Ksantini, Shakeeb Murtaza, and Tehseen Zia

Abstract—Dense self-expression matrices and full-affinity spectral clustering limit the scalability of subspace clustering. We introduce the Latent Orthogonal Optimization Model for Subspace Clustering (LoomSC), a framework that addresses both bottlenecks through projector factorization and exact spectral reduction. Motivated by the spectral structure of least-squares regression, LoomSC jointly learns latent features and a projector self-representation through two thin factors. Alternating Procrustes and least-squares updates preserve the sample factor’s orthogonality while keeping the coefficient matrix implicit. We construct a nonnegative quadratic affinity that preserves the projector’s support. An exact feature map then reduces its normalized spectral problem to an eigenproblem whose dimension depends only on the factor width. Neither the full affinity nor the sample Laplacian needs to be formed. Our analysis quantifies the projector approximation and identifies conditions for subspace preservation and within-subspace connectivity. For fixed dimensions and iteration budgets, the complete pipeline has linear time and memory complexity in the number of samples. Across five image-clustering benchmarks, LoomSC ranks first or second in all 15 dataset–metric comparisons against 9 state-of-the-art baselines. Its mean accuracy exceeds the highest baseline mean by 6.66 percentage points. Synthetic experiments scale to 500,000 samples while maintaining at least 99.8% accuracy.

I. INTRODUCTION

In several cases, high-dimensional data can be modeled by a generation process from the union of multiple low-dimensional linear subspaces. In this context, subspace clustering is a prominent unsupervised strategy that aims to partition the data into clusters by identifying these subspaces, where each cluster is defined by exactly one low-dimensional subspace. The subspace assumption is widely used in visual analysis and has been successfully exploited in several applications, such as image segmentation [1], motion segmentation [2], [3], and face clustering [3], [4]. For example, face images of a person can be represented by a low-dimensional subspace for a fixed pose and under Lambertian reflectance [5].

The most recent subspace clustering methods [6]–[8] follow a two-step strategy. First, these methods learn an affinity matrix that captures the similarity between points within the same subspace. Second, a graph clustering approach, such as spectral clustering, is used to identify the clusters. The affinity matrix is constructed according to the self-expression property [8], [9] of the subspaces. More precisely, each data point can be expressed as a linear combination of points in the same subspace. Given

a dataset $X \in \mathbb{R}^{d_x \times n}$ with n data points of dimensionality d_x , the optimization problem associated with this strategy is:

$$\min_C \frac{1}{2} \|X - XC\|_F^2 + \lambda \theta(C), \quad (1)$$

where $C = (c_{ij}) \in \mathbb{R}^{n \times n}$ is a matrix that captures the self-expression property based on the first term; $\|\cdot\|_F$ denotes the Frobenius norm, which is defined for any matrix $M = (m_{ij})$ by $\|M\|_F = (\sum_{i=1}^n \sum_{j=1}^n m_{ij}^2)^{\frac{1}{2}}$; the second term, controlled by the function θ , serves as a regularization to avoid trivial solutions (e.g., $C = I$) and to adjust the structure of the self-representation matrix C ; $\lambda \in \mathbb{R}_+^*$ is a balancing hyperparameter between the two terms. Based on the self-representation matrix, the affinity graph is constructed by computing its adjacency matrix $A_{\text{abs}} = (a_{ij}^{\text{abs}}) \in \mathbb{R}^{n \times n}$, such that $A_{\text{abs}} = \frac{|C| + |C^T|}{2}$.

Different regularizers encode complementary structural preferences. Sparse Subspace Clustering (SSC) [8] promotes sparse coefficients through an ℓ_1 penalty. Low-Rank Representation (LRR) [10] favors a low-rank coefficient matrix through nuclear-norm regularization. Multi-Subspace Representation (MSR) [11] combines these penalties, whereas Block Diagonal Representation (BDR) [6] directly encourages a block structure. Least Squares Regression (LSR) [9] uses squared Frobenius regularization and admits a closed-form solution. Its grouping effect encourages correlated samples to receive similar coefficients. These formulations seek to suppress cross-subspace connections while retaining informative relations within each subspace. Both requirements matter because a sparse affinity can separate samples from different subspaces yet fragment samples from the same subspace [12].

The self-expression framework nevertheless introduces two distinct computational bottlenecks. Explicitly storing a dense coefficient matrix requires $\mathcal{O}(n^2)$ memory, even when its estimation admits an efficient solver. Constructing a full affinity retains this quadratic storage cost. The spectral stage adds a second bottleneck. A dense eigendecomposition requires $\mathcal{O}(n^3)$ operations, while partial eigensolvers still perform repeated products with the sample-level affinity. Consequently, accelerating coefficient estimation alone does not ensure a scalable clustering pipeline. Both self-expression and spectral assignment must avoid dense sample-by-sample computations.

Several strategies reduce these costs. Sampling methods replace the full dictionary with a smaller set of representatives [13]–[16]. Subspace-preserving selection seeks representatives that span the underlying subspaces [17]. Sparse solvers accelerate coefficient estimation through greedy pursuit [18], active-set selection [19], or stochastic dictionaries [20].

N. Mrabah and S. Murtaza are with the École de technologie supérieure (ÉTS), Montréal, QC, Canada. Y. Melki and R. Ksantini are with the University of Bahrain (UOB), Kingdom of Bahrain. M. Bouguessa is with the Université du Québec à Montréal (UQAM), Montréal, QC, Canada. T. Zia is with COMSATS University Islamabad, Islamabad, Pakistan.

Structure-aware methods exploit compact graph models [21] or structured affinity estimation [22]. Neural coefficient predictors [23] provide another route by learning a mapping that replaces repeated regression.

Representative-based reduction is particularly attractive because it replaces an all-sample dictionary with $m \ll n$ representatives. Each sample is then described by m coefficients instead of n . Anchor-graph methods can also obtain the spectral embedding from a compact factor [15], [16]. Their effectiveness depends on how well the representatives capture the subspaces in the supplied feature space. Learning the coefficients improves the graph, but does not by itself adapt the image representation. This motivates a compact model in which both the representation and the directions used for self-expression are learned from the data.

The representation itself is important because complex appearance variation and data corruption can weaken the linear subspace structure of raw observations. Kernel methods address nonlinear structure through an implicit feature mapping [24]. Deep subspace clustering [25]–[27] instead couples an auto-encoder with self-expression in the latent space. The reconstruction objective preserves information about the inputs. The self-expression objective encourages latent features that conform to a subspace model. Joint optimization therefore adapts the features to clustering rather than imposing self-expression on a fixed representation.

Joint learning also makes computational compactness essential. A dense self-expression layer introduces n^2 sample-dependent coefficients that must be optimized together with the encoder [25], [26]. Neural parameterizations and mini-batch training can reduce training costs [28], but do not by themselves eliminate full-affinity spectral inference. Our goal is to retain both pairwise self-expression and normalized spectral clustering while learning nonlinear representations, without constructing dense sample-level matrices at either stage.

To this end, we introduce the Latent Orthogonal Optimization Model for Subspace Clustering (LoomSC). Motivated by the spectral structure of LSR, LoomSC represents self-expression through a projector $C = PP^\top$. The sample factor $P \in \mathbb{R}^{n \times m}$ has orthonormal columns and a prescribed width m . An equivalent two-factor objective separates latent directions from sample coefficients and couples both to the auto-encoder. Alternating Procrustes and least-squares updates refine these factors while preserving orthogonality. Neither the coefficient matrix nor a sample Gram matrix needs to be formed. The latent directions are updated using all samples rather than remaining fixed to selected representatives. Our analysis quantifies the projector approximation to LSR and identifies conditions under which the full row-space projector preserves subspace memberships and within-subspace connectivity.

LoomSC extends this compact representation to spectral assignment through a nonnegative quadratic affinity. Squaring the projector coefficients preserves their zero pattern and thus retains the connections encoded by self-expression. An exact feature map represents this affinity using $q = m(m+1)/2$ features per sample. Its normalized spectral problem then reduces to a q -dimensional eigenproblem. This reduction avoids forming either the full affinity or the sample Laplacian. For

fixed dimensions and iteration budgets, the complete pipeline has linear time and memory complexity in n . The same compact representation therefore supports feature learning, self-expression, and global spectral assignment.

Contributions. (1) We introduce a two-factor projector model for joint feature and self-expression learning with orthogonality-preserving updates. (2) We construct a support-preserving nonnegative affinity and derive an exact reduction of its normalized spectral problem without forming full pairwise matrices. (3) We quantify the projector approximation to LSR and establish conditions for subspace preservation and connectivity. We also prove linear time and memory bounds for the complete pipeline under fixed dimensions and iteration budgets. (4) Across five image-clustering benchmarks, LoomSC ranks first or second in all 15 dataset–metric comparisons against nine baselines. Its mean accuracy exceeds the highest baseline mean by 6.66 percentage points. Experiments on synthetic data reach 500,000 samples while maintaining at least 99.8% accuracy throughout the tested range.

II. RELATED WORK

In line with the focus of this work, we discuss the advantages and limitations of two existing subspace clustering categories: the scalable subspace clustering category and the deep subspace clustering category. In that respect, we highlight the pertinence of introducing a subspace clustering approach that combines the advantages of both categories.

A. Scalable subspace clustering:

Least-squares regression (LSR) [9] provides an efficient alternative to sparse coefficient estimation. Its squared Frobenius regularization encourages a grouping effect among correlated samples and admits a closed-form solution. However, solving the regression analytically does not eliminate the dense $n \times n$ coefficient matrix, where n is the number of samples. Its explicit construction and storage remain quadratic in n . Thus, improving the regression solver alone does not resolve the full-affinity bottleneck. LoomSC instead adopts a prescribed-width projector model and learns its thin factors directly. This changes the representation of self-expression rather than merely accelerating a dense coefficient solve.

Sparse methods reduce storage by retaining a small number of coefficients per sample. SSC-OMP [18] replaces ℓ_1 optimization with orthogonal matching pursuit and establishes subspace-preserving guarantees under suitable conditions. EnSC [19] combines ℓ_1 and squared ℓ_2 regularization to balance subspace preservation and graph connectivity. Its oracle-guided active-set solver restricts optimization to candidate dictionary elements. Both approaches substantially accelerate coefficient estimation. Nevertheless, their dictionary searches still compare target residuals with the full dataset. Repeating these searches across samples retains a quadratic sample-size dependence for fixed sparsity and search budgets. Sparse spectral solvers reduce the subsequent graph-processing cost, but do not remove this coefficient-estimation cost. LoomSC avoids these separate full-dictionary searches by optimizing a shared factorization through thin matrix products and an orthogonal Procrustes update.

Stochastic sparse subspace clustering (SSSC) [20] addresses weak within-subspace connectivity through dictionary dropout. For normalized samples, the expected dropout objective induces an additional squared ℓ_2 penalty, which encourages denser representations. Its scalable implementation combines coefficients from multiple reduced dictionaries through a consensus formulation. This improves connectivity while preserving a sparse affinity. However, with a fixed retained fraction, each dictionary still grows with n , and the reduced regression must be repeated for every target. Parallel execution accelerates these subproblems without changing their aggregate arithmetic complexity. LoomSC obtains compactness through a common learned latent factor rather than repeated stochastic dictionary reduction. The resulting sample relations are represented jointly without an ensemble of coefficient estimates.

Representative-based approaches reduce the dictionary size more directly. LMVSC [15] learns nonnegative sample-to-anchor coefficients and computes its spectral embedding from a compact normalized factor. SGL [16] further couples anchor-graph learning with a spectral constraint that targets k connected components. These methods achieve linear sample-size scaling for fixed anchor and optimization budgets. Their graph weights are learned, but their reconstruction is performed using supplied features and an anchor dictionary. Consequently, they do not jointly adapt the image representation to the self-expression objective. LoomSC preserves a compact representation while learning both the encoder and the latent factor. After initialization, the columns of the latent factor are updated from all samples rather than remaining fixed anchor representatives. Unlike the nonnegative anchor weights in LMVSC and SGL, its sample-factor coefficients can be signed. This permits linear combinations beyond convex anchor assignments.

Selective sampling-based scalable sparse subspace clustering (S^5C) [29] selects representatives using stochastic estimates of subgradient violations. It then solves restricted sparse regressions for all samples. Its spectral stage exploits a graph with only $O(nm)$ nonzero entries, where m is the representative budget. Thus, S^5C reduces both regression and spectral costs without requiring a dense affinity. However, its representation remains tied to selected samples in the feature space. LoomSC replaces this exemplar dictionary with adaptive latent directions and jointly refines the features used to construct them.

A-DSSC [22] addresses a different source of difficulty: the compatibility between self-expression coefficients and spectral normalization. It first estimates coefficients and then obtains a doubly stochastic affinity through quadratically regularized optimal transport. A support-restricted solver exploits sparsity to accelerate this second stage. This gives a principled alternative to heuristic coefficient normalization, but retains separate coefficient-estimation and transport problems. LoomSC instead derives the affinity and its normalization from the same learned projector. For its squared affinity, each degree equals the squared norm of the corresponding sample-factor row. Thus, degree computation requires only the thin factor and no additional transport optimization.

Compact spectral computation also appears in scalable attributed-graph subspace clustering (SAGSC) [30]. SAGSC constructs a projector from graph-smoothed features and uses

a quadratic kernel to obtain a nonnegative factorized affinity. This permits spectral clustering without explicitly assembling pairwise weights. LoomSC applies this spectral principle to jointly learned image representations rather than fixed graph-smoothed features. Its two-factor objective allows the projector to evolve with the encoder, and its analysis connects the learned factors to dominant latent directions and subspace structure. The distinction is therefore the integration of adaptive representation learning with a compact spectral construction, rather than factorization alone.

B. Deep subspace clustering:

DSCNet [25] integrates a differentiable self-expression layer into a convolutional auto-encoder. This couples feature learning with the reconstruction of each latent sample from the others. $S^2\text{ConvSCN}$ [26] additionally uses spectral assignments to supervise feature learning and coefficient estimation. These formulations make the representation responsive to clustering, but retain a sample-to-sample self-expression matrix. Additional supervision does not remove its quadratic parameter storage. LoomSC retains joint feature and self-expression learning while replacing this layer with two thin factors. Its factor updates admit closed-form solutions, and their dimensions are controlled by the prescribed factor width.

Neural coefficient prediction offers another route to scalability. SENet [23] parameterizes pairwise coefficients with a neural network that can generalize to unseen samples. Its objective learns a coefficient-prediction rule for the supplied representation rather than jointly learning a new feature space for self-expression. This separates the number of network parameters from the number of coefficient entries. However, a compact predictor does not itself provide a compact spectral representation of the resulting affinity. LoomSC instead learns a sample factor whose algebraic structure supports both self-expression and the subsequent spectral reduction.

PRO-DSC [28] jointly learns latent representations and self-expression coefficients. A log-determinant regularizer controls representation collapse under the conditions established in its analysis. Its implementation predicts coefficients through neural mappings and Sinkhorn normalization, which supports mini-batch training. The distinction from LoomSC arises at full-set clustering. PRO-DSC constructs the coefficient matrix over the clustering set and applies spectral clustering to the resulting affinity. For n samples, this inference route retains quadratic matrix storage despite efficient training. LoomSC does not reconstruct a full coefficient matrix after learning. The final sample factor directly defines a nonnegative affinity whose normalized spectral problem is solved in a feature space determined by the factor width. Thus, the computational reduction extends through spectral assignment rather than ending with network training.

Other deep methods reduce cost by changing how the final partition is obtained. The large-scale extension of pseudo-supervised deep subspace clustering [27] trains on a subset and assigns the remaining latent samples through nearest-neighbor classification. LoomSC instead retains an explicit model of sample relations while avoiding their dense storage. All samples

participate in factor learning, and the final partition is obtained by spectral clustering of the learned affinity rather than by extending labels from a training subset. Its efficiency follows from representing that affinity compactly while preserving the self-expression formulation.

III. LOOMSC

The proposed method develops a compact self-representation by exploiting the spectral structure of least-squares regression (LSR). We first analyze the LSR solution and motivate a projector approximation. We then express this approximation through two thin factors that are learned jointly with a deep auto-encoder. Finally, we construct a nonnegative affinity from the learned factors and obtain the spectral embedding through a reduced eigenproblem. Both phases operate directly on the factors, which makes their computational cost linear.

Let the matrix $X = [x_1, \dots, x_n] \in \mathbb{R}^{d_x \times n}$ contain the input samples $x_i \in \mathbb{R}^{d_x}$, and let k denote the number of clusters.

A. A Projector Approximation Motivated by LSR

LSR [9] models each sample through a linear combination of the data collection and penalizes the energy of the coefficients. For the input X , its diagonal-unconstrained formulation is

$$\min_{C \in \mathbb{R}^{n \times n}} \frac{1}{2} \|X - XC\|_F^2 + \lambda \|C\|_F^2, \quad \lambda > 0. \quad (2)$$

The reconstruction term encourages self-expression. The Frobenius penalty controls coefficient magnitudes while allowing correlated samples to contribute jointly. Differentiation with respect to C gives the closed-form solution of this problem

$$C_{\text{LSR}}^* = (X^\top X + 2\lambda I_n)^{-1} X^\top X, \quad (3)$$

where I_n is the $n \times n$ identity matrix. A direct evaluation of Eq. (3) requires forming the Gram matrix $X^\top X$, at a cost of $O(d_x n^2)$, followed by the solution of an $n \times n$ linear system, which requires $O(n^3)$ operations using a dense factorization. Moreover, the resulting coefficient matrix contains n^2 entries and therefore requires $O(n^2)$ memory. Although the inversion cost can be reduced in particular dimensional regimes, explicitly constructing the dense self-representation remains quadratic in the number of samples. Thereofre, we examine the spectral structure of C_{LSR}^* to derive a compact representation.

Let $r = \text{rank}(X)$. Since $X^\top X$ is symmetric positive semidefinite with rank r , its compact eigendecomposition is

$$X^\top X = V_x \text{diag}(\beta_1, \dots, \beta_r) V_x^\top, \quad (4)$$

where $V_x \in \mathbb{R}^{n \times r}$ has orthonormal columns given by the eigenvectors of $X^\top X$ associated with its positive eigenvalues, and $\beta_1 \geq \dots \geq \beta_r > 0$ are the corresponding eigenvalues.

Proposition 1 (Spectral structure of LSR). *For $\lambda > 0$, the LSR solution is symmetric and positive semidefinite, with rank r . Its nonzero eigenvalues are*

$$\tau_j = \frac{\beta_j}{\beta_j + 2\lambda}, \quad j = \{1, \dots, r\}, \quad (5)$$

and $C_{\text{LSR}}^* = V_x \text{diag}(\tau_1, \dots, \tau_r) V_x^\top$.

The eigenvectors identify the sample relationships represented by LSR. Eq. (5) also shows that directions with β_j large relative to 2λ have eigenvalues close to one. This observation motivates replacing their spectral weights by one and retaining the corresponding subspace. For the full row space, define $C_{\text{row}} = V_x V_x^\top \in \mathbb{R}^{n \times n}$. The approximation error is

$$\|C_{\text{LSR}}^* - C_{\text{row}}\|_F^2 = \sum_{j=1}^r \left(\frac{2\lambda}{\beta_j + 2\lambda} \right)^2. \quad (6)$$

Thus, the projector approaches the LSR representation as the regularization becomes small relative to the positive data spectrum. This gives a direct spectral justification for an idempotent self-representation.

To obtain a compact model, choose a factor width m with $k \leq m < n$. When $m \leq r$, let $V_{x,m} \in \mathbb{R}^{n \times m}$ contain the first m columns of V_x . The corresponding approximation satisfies

$$\|C_{\text{LSR}}^* - V_{x,m} V_{x,m}^\top\|_F^2 = \sum_{j=1}^m (1 - \tau_j)^2 + \sum_{j=m+1}^r \tau_j^2. \quad (7)$$

For the retained directions $j \leq m$, the approximation replaces the LSR eigenvalue τ_j by the projector eigenvalue 1. The first term in Eq. (7) measures the resulting error. The remaining directions $j > m$ are discarded, and the second term measures the corresponding loss of the LSR spectral components. Since τ_j is monotonically increasing with β_j , the ordering of the τ_j 's is identical to that of the positive eigenvalues of $X^\top X$. Hence, for a fixed width m , retaining the first m eigenvectors gives the closest projector among the rank- m projectors formed from the eigenvectors of $X^\top X$.

Setting $m = r$ recovers the full row-space projector $C_{\text{row}} = V_x V_x^\top$. However, constructing this projector requires determining r and computing all r nonzero right singular vectors of X . For a dense $X \in \mathbb{R}^{d_x \times n}$, computing the full singular subspace costs $O(d_x n \min\{d_x, n\})$, while storing V_x requires $O(nr)$ memory. These costs become prohibitive when r grows with the data size. Thus, LoomSC does not attempt to recover the complete rank- r projector. Instead, m is treated as a prescribed model width, and the corresponding thin factors are learned directly in the subsequent formulation.

Accordingly, we parameterize the self-representation as

$$C = PP^\top, \quad P \in \mathbb{R}^{n \times m}, \quad P^\top P = I_m. \quad (8)$$

The factor P stores the relationships between all samples through nm coefficients. Its orthogonality makes C symmetric, positive semidefinite, and idempotent, with rank m . It also fixes the coefficient energy:

$$\|PP^\top\|_F^2 = \text{tr}(P^\top P) = m. \quad (9)$$

Substituting Eq. (8) into the LSR objective in Eq. (2) yields the following optimization over fixed-width projectors:

$$\min_{P^\top P = I_m} \|X - XPP^\top\|_F^2. \quad (10)$$

Since $PP^\top$ is an orthogonal projector, Eq. (10) is equivalent to maximizing $\text{tr}(P^\top X^\top X P)$. Hence, for $m \leq r$, an optimal solution is $P^* = V_{x,m} R$, where $R \in \mathbb{R}^{m \times m}$ is any

orthogonal matrix. All such solutions span the same dominant m -dimensional subspace and yield

$$P^* P^{*\top} = V_{x,m} V_{x,m}^\top. \quad (11)$$

The minimum reconstruction error is $\sum_{j=m+1}^r \beta_j$. Thus, the exact solution requires computing the dominant right singular subspace of the current representation. This is practical when the representation is fixed, but it becomes less attractive when the features are learned jointly, since the dominant subspace must be recomputed as the representation evolves. To address this problem, LoomSC replaces the repeated spectral solution with a two-factor formulation. The resulting updates involve only thin matrix products and an $n \times m$ Procrustes problem, while the full coefficient matrix C remains implicit.

B. Learning a Latent Representation

We apply the compact projector model to a learned latent space. This lets the self-expression structure guide feature extraction. We use an encoder $\mathcal{E}$ and a decoder $\mathcal{D}$ with parameter collections W and $\widehat{W}$, respectively. They define

$$\begin{aligned} Z &= \mathcal{E}(X; W) \in \mathbb{R}^{d_z \times n}, \\ \widehat{X} &= \mathcal{D}(Z; \widehat{W}) \in \mathbb{R}^{d_x \times n}, \end{aligned} \quad (12)$$

where d_z is the latent dimension and $z_i \in \mathbb{R}^{d_z}$ is the representation of x_i . The vector $\widehat{x}_i \in \mathbb{R}^{d_x}$ is the i^{th} column of the matrix $\widehat{X}$. For image inputs, convolutional layers extract spatial features, which are flattened to form the columns of the latent representation matrix Z . The decoder reconstructs the input from these features. Its reconstruction objective is

$$J_{\text{rec}}(W, \widehat{W}) = \frac{1}{n} \|X - \widehat{X}\|_F^2. \quad (13)$$

This objective trains the representation to retain information about the observed samples. The projector objective in Eq. (10), applied to Z , imposes a complementary subspace structure.

C. Self-Expression Through Two Thin Factors

To optimize projector self-expression in the latent space, we introduce a latent factor $L \in \mathbb{R}^{d_z \times m}$. Its columns represent learned directions in the latent space. The following decomposition connects the resulting two-factor model to projector self-expression:

$$\|Z - LP^\top\|_F^2 = \|Z - ZPP^\top\|_F^2 + \|L - ZP\|_F^2. \quad (14)$$

The identity in Eq. (14) holds for every P satisfying $P^\top P = I_m$. The proof is provided in Appendix B. Minimizing its left-hand side with respect to L gives $L = ZP$. Hence, we obtain:

$$\min_{L, P^\top P = I_m} \|Z - LP^\top\|_F^2 = \min_{P^\top P = I_m} \|Z - ZPP^\top\|_F^2. \quad (15)$$

The two-factor formulation separates the latent directions represented by L from the sample coefficients encoded by P . This separation gives simple block updates and requires only products involving thin matrices.

For fixed Z , let $r_z = \text{rank}(Z)$ and write $Z = U_z \Sigma_z V_z^\top$. The matrices $U_z \in \mathbb{R}^{d_z \times r_z}$ and $V_z \in \mathbb{R}^{n \times r_z}$ have orthonormal

columns. The diagonal matrix $\Sigma_z \in \mathbb{R}^{r_z \times r_z}$ contains the positive singular values $\sigma_1 \geq \dots \geq \sigma_{r_z} > 0$.

Proposition 2 (Reconstruction through dominant directions). *For a fixed latent matrix Z , the optimal factor residual is*

$$\min_{L, P^\top P = I_m} \|Z - LP^\top\|_F^2 = \sum_{j=m+1}^{r_z} \sigma_j^2, \quad (16)$$

where an empty sum is zero. For $m \leq r_z$, a minimizing P spans an m -dimensional dominant right singular subspace of Z . For $m \geq r_z$, an optimal projector contains the row space of Z and satisfies $ZPP^\top = Z$. In particular, $m = r_z$ gives $C = V_z V_z^\top$.

Proposition 2 connects the learned factors to the dominant spectral directions of Z . Thus, the spectral approximation developed above applies directly to the learned representation. It also quantifies how the width controls the amount of latent information represented by the model. The proofs of (14) and Proposition 2 are given in Appendix B.

We couple the factor objective with the auto-encoder to learn the latent features and their self-representation jointly:

$$\begin{aligned} \min_{W, \widehat{W}, L, P} J(W, \widehat{W}, L, P) &= J_{\text{rec}}(W, \widehat{W}) + \|Z - LP^\top\|_F^2, \\ \text{subject to } P^\top P &= I_m. \end{aligned} \quad (17)$$

The reconstruction term retains information about the input samples. The factor term encourages the latent codes to admit a common compact self-representation. Joint optimization allows the encoder to adjust this representation as the factors are refined. For sample i , the row coefficients $p_i = P_{i,:}^\top \in \mathbb{R}^m$ reconstruct its latent vector as Lp_i .

D. Alternating Optimization

We first pretrain the auto-encoder with (13). We then initialize the columns of L using m latent samples selected by k -means++ seeding [31]. This spreads the initial directions across the observed latent features. Appendix E specifies the sampling rule. The factor P is initialized by its update below. Each joint iteration updates the network parameters, refreshes Z , and then updates P and L in sequence.

a) *Network parameters:* With L and P fixed, an Adam step [32] updates the encoder and decoder:

$$(W, \widehat{W}) \leftarrow \text{Adam}_t((W, \widehat{W}), \nabla_{W, \widehat{W}} J), \quad (18)$$

where t is the joint iteration index. The factor term contributes the latent gradient $2(Z - LP^\top)$, which encourages the encoder to produce representations consistent with the current factors. To control activation memory, the samples can be processed in chunks $\{\mathcal{B}_b\}$ while keeping L and P fixed:

$$J = \sum_b \sum_{i \in \mathcal{B}_b} \left(\frac{1}{n} \|x_i - \widehat{x}_i\|_2^2 + \|z_i - Lp_i\|_2^2 \right). \quad (19)$$

The gradients are accumulated over all chunks before a single update of $(W, \widehat{W})$. Hence, chunking reduces memory usage without changing the full-data objective. After the update, the encoder recomputes Z before updating P and L .

Algorithm 1 LoomSC**Require:** $X \in \mathbb{R}^{d_x \times n}$, $k \leq m < n$, T_{pre}, T .**Ensure:** Cluster labels $\gamma_1, \dots, \gamma_n$.

- 1: Pretrain $(W, \widehat{W})$ for T_{pre} updates using (13).
- 2: Compute Z using (12).
- 3: Initialize L with m sample columns selected by applying k -means++ to Z .
- 4: Initialize P using (21).
- 5: **for** $t = 1, \dots, T$ **do**
- 6: Fix L and P and accumulate the gradient using (19).
- 7: Update $(W, \widehat{W})$ using (18).
- 8: Recompute Z using (12).
- 9: Update P using the thin-SVD step in (21).
- 10: Update $L \leftarrow ZP$ using (22).
- 11: **end for**
- 12: Obtain the cluster labels from Algorithm 2 using the final P and Z .
- 13: **return** $\gamma_1, \dots, \gamma_n$.

b) *Sample factor:* For fixed Z and L , orthogonality gives

$$\|Z - LP^\top\|_F^2 = \|Z\|_F^2 + \|L\|_F^2 - 2\text{tr}(P^\top Z^\top L). \quad (20)$$

Thus, updating P amounts to maximizing the trace term under $P^\top P = I_m$. This is an orthogonal Procrustes problem. To solve it, we form the thin product $Z^\top L \in \mathbb{R}^{n \times m}$ and compute

$$Z^\top L = U_P \Sigma_P V_P^\top, \quad P \leftarrow U_P V_P^\top, \quad (21)$$

where $U_P \in \mathbb{R}^{n \times m}$ and $V_P \in \mathbb{R}^{m \times m}$ have orthonormal columns. The diagonal matrix $\Sigma_P \in \mathbb{R}^{m \times m}$ contains the singular values. This update attains the maximum trace and preserves the orthogonality constraint.

c) *Latent factor:* For fixed Z and P , differentiation gives $\nabla_L \|Z - LP^\top\|_F^2 = 2(L - ZP)$. The minimizing update is

$$L \leftarrow ZP. \quad (22)$$

Each column of L is refined using the current latent samples and their coefficients. Together, the two updates adapt the representation to the full dataset using an $n \times m$ decomposition and thin matrix products.

Proposition 3 (Descent of the factor updates). *For fixed Z , each update in (21) and (22) minimizes the factor objective with respect to its block. Repeating these updates yields a nonincreasing sequence of objective values that converges to a finite limit.*

Appendix B proves this result. Algorithm 1 combines the factor updates with representation learning for a prescribed number of joint iterations.

E. Subspace Structure of the Learned Projector

The projector interpretation also explains how self-expression can preserve subspace memberships. Consider latent samples spanning subspaces $S_1, \dots, S_k \subseteq \mathbb{R}^{d_z}$ with dimensions

$r_1, \dots, r_k$. For independent subspaces, their sum is direct and

$$\dim(S_1 + \dots + S_k) = \sum_{a=1}^k r_a = r_z. \quad (23)$$

Proposition 4 (Subspace preservation and connectivity). *Suppose all latent samples are nonzero and span the direct sum in (23). If $m = r_z < n$ and the fixed- Z factor objective attains its minimum, then $C = PP^\top$ is block diagonal after grouping samples by subspace. Within a subspace, its nonzero off-diagonal coefficients define a connected graph whenever that subspace's samples admit no partition into two nonempty sets whose spans form a direct sum.*

The result identifies conditions under which the projector both separates the generating subspaces and connects the samples within each subspace. Its proof uses the block structure of the row-space projector and appears in Appendix C. The next phase converts these coefficients into nonnegative graph weights while preserving their nonzero connections.

F. Scalable Spectral Clustering from the Learned Factors

The learned factor P provides the pairwise coefficients through $C = PP^\top$. We now use this structure to obtain the final partition directly from compact matrices. The construction has two steps. First, we convert the coefficients into nonnegative affinities while preserving their support. We then express the normalized spectral problem through a quadratic feature map whose dimension depends only on m .

a) *A nonnegative affinity with an explicit feature map:*

The magnitude of $C_{ij} = p_i^\top p_j$ measures the strength of the relation between samples i and j . Squaring this coefficient gives the symmetric affinity

$$A = C \circ C \in \mathbb{R}^{n \times n}, \quad A_{ij} = (p_i^\top p_j)^2, \quad (24)$$

where $\circ$ denotes entrywise multiplication. This construction assigns nonnegative weights to both positive and negative coefficients. It also preserves their zero pattern. Thus, any separation between subspaces encoded by C is retained in A . Moreover, the affinity is invariant to an orthogonal change of basis in P .

The quadratic form in (24) admits an explicit feature representation. Let $q = m(m+1)/2$. For a symmetric matrix $S \in \mathbb{R}^{m \times m}$, $\text{svec}(S) \in \mathbb{R}^q$ contains its m diagonal entries followed by its upper-triangular entries. Each off-diagonal entry is multiplied by $\sqrt{2}$. This scaling preserves the Frobenius inner product between symmetric matrices. We define

$$\begin{aligned} \phi(p) &= \text{svec}(pp^\top) \in \mathbb{R}^q, \\ F_{i:} &= \phi(p_i)^\top, \quad F \in \mathbb{R}^{n \times q}. \end{aligned} \quad (25)$$

Consequently, $\phi(p_i)^\top \phi(p_j) = (p_i^\top p_j)^2$, which gives

$$A = FF^\top. \quad (26)$$

Hence, the $n \times n$ affinity matrix is represented exactly through q features per sample and does not need to be formed explicitly.

We retain the diagonal entries of A as self-loop weights. The projector identity $C^2 = C$ then gives a particularly simple

degree formula,

$$\delta_i = \sum_{j=1}^n A_{ij} = (C^2)_{ii} = C_{ii} = \|p_i\|_2^2, \quad \sum_{i=1}^n \delta_i = m. \quad (27)$$

Each degree therefore requires only a row norm of P . Computing all degrees takes $O(nm)$ time. Let $\mathcal{I}_+ = \{i : \delta_i > 0\}$ denote the indices with positive degrees and let $n_+ = |\mathcal{I}_+|$. Spectral normalization is performed on these samples. Zero-degree samples receive labels through their latent features after clustering.

b) *An exact reduced spectral problem:* Let $F_+ \in \mathbb{R}^{n_+ \times q}$ contain the rows indexed by $\mathcal{I}_+$ in increasing order. Define

$$\begin{aligned} \Delta &= \text{diag}(\delta_i : i \in \mathcal{I}_+) \in \mathbb{R}^{n_+ \times n_+}, \\ B &= \Delta^{-1/2} F_+ \in \mathbb{R}^{n_+ \times q}, \\ G &= B^\top B \in \mathbb{R}^{q \times q}. \end{aligned} \quad (28)$$

The normalized affinity is $BB^\top$, and its normalized Laplacian is $I_{n_+} - BB^\top$. Thus, the smallest Laplacian eigenvalues correspond to the largest eigenvalues of $BB^\top$. The following result transfers their computation to G .

Proposition 5 (Exact spectral reduction). *For $P^\top P = I_m$, the normalized feature matrix satisfies*

$$m \leq \text{rank}(B) \leq \min\{n_+, q\}. \quad (29)$$

The positive eigenvalues of G and $BB^\top$ coincide, including multiplicities. If $Gv_j = \nu_j v_j$ with $\nu_j > 0$ and orthonormal $v_j \in \mathbb{R}^q$, then $Bv_j/\sqrt{\nu_j}$ are orthonormal eigenvectors of $BB^\top$ with eigenvalues ν_j .

The proof is provided in Appendix D. Since $k \leq m$, the reduced problem supplies at least k positive eigenvalues. The embedding dimension is k , whereas m controls the factor width and q determines the reduced spectral dimension.

The degree identity also gives one dominant eigenpair analytically,

$$v_1 = \frac{\text{svec}(I_m)}{\sqrt{m}}, \quad \nu_1 = 1, \quad Bv_1 = \frac{(\sqrt{\delta_i})_{i \in \mathcal{I}_+}}{\sqrt{m}}. \quad (30)$$

We retain this vector and compute the largest $k-1$ eigenpairs on $v_1^\perp$. This gives k dominant modes while providing a strictly positive coordinate for every active sample. The sample embedding and its normalized rows are

$$\begin{aligned} H &= B[v_1, \dots, v_k] \text{diag}(\nu_1, \dots, \nu_k)^{-1/2} \in \mathbb{R}^{n_+ \times k}, \\ \hat{H}_i &= H_i / \|H_i\|_2, \quad i = 1, \dots, n_+. \end{aligned} \quad (31)$$

Equation (30) ensures that every row norm is positive. Retaining the eigenvalue-one modes also preserves the component indicators when the affinity graph has k connected components. In that case, row normalization maps each component to a distinct embedding point, as established in Appendix D.

Following normalized spectral clustering [33], we apply k -means to the rows of $\hat{H}$. Let $\gamma_i \in \{1, \dots, k\}$ denote the resulting label of active sample i . For a zero-degree sample,

Algorithm 2 Spectral clustering through the quadratic feature map

Require: $P \in \mathbb{R}^{n \times m}$ with $P^\top P = I_m$, $Z \in \mathbb{R}^{d_z \times n}$, $k \leq m$, $R_{\text{km}}, T_{\text{km}}$.

Ensure: Cluster labels $\gamma_1, \dots, \gamma_n$.

- 1: Compute δ_i by (27) and retain the indices $\mathcal{I}_+$ with $\delta_i > 0$.
- 2: Set $q = m(m+1)/2$ and the feature ordering in (25).
- 3: Set v_1, ν_1 using (30).
- 4: **if** $k > 1$ **then**
- 5: **if** direct eigensolver **then**
- 6: Accumulate $G = \sum_{i \in \mathcal{I}_+} \phi(p_i) \phi(p_i)^\top / \delta_i$ using (28).
- 7: Compute its largest $k-1$ eigenpairs on $v_1^\perp$.
- 8: **else**
- 9: Compute the largest $k-1$ eigenpairs on $v_1^\perp$ with a symmetric partial eigensolver using (33).
- 10: **end if**
- 11: **end if**
- 12: Recover H and normalize its rows by (31).
- 13: Run R_{km} seeded k -means restarts using (54), with at most T_{km} iterations each.
- 14: Retain the restart with minimum distortion and restore the original sample indices.
- 15: Assign zero-degree samples by (32).
- 16: **return** $\gamma_1, \dots, \gamma_n$.

we use its distance to the latent centers of these clusters,

$$\begin{aligned} \mathcal{J}_a &= \{i \in \mathcal{I}_+ : \gamma_i = a\}, \\ \zeta_a &= \frac{1}{|\mathcal{J}_a|} \sum_{i \in \mathcal{J}_a} z_i \in \mathbb{R}^{d_z}, \\ \gamma_i &\in \arg \min_{a \in \{1, \dots, k\}} \|z_i - \zeta_a\|_2^2, \quad i \notin \mathcal{I}_+. \end{aligned} \quad (32)$$

The clustering conventions in Appendix E ensure nonempty active clusters and specify initialization and tie-breaking. Together, these steps assign every sample to a cluster. The corresponding membership matrix is $\Gamma \in \{0, 1\}^{n \times k}$ with $\Gamma_{ia} = \mathbf{1}\{\gamma_i = a\}$.

c) *Direct and implicit implementations:* When q is modest, we accumulate G from normalized feature rows and solve its symmetric eigenproblem directly. For larger q , the same operator can be applied by streaming the rows of P . Let $\text{smat} : \mathbb{R}^q \rightarrow \mathbb{R}^{m \times m}$ be the inverse of svec on symmetric matrices. For any $v \in \mathbb{R}^q$,

$$Gv = \text{svec} \left(\sum_{i \in \mathcal{I}_+} \frac{p_i^\top \text{smat}(v) p_i}{\delta_i} p_i p_i^\top \right). \quad (33)$$

Each operator evaluation takes $O(nm^2)$ time. A symmetric partial eigensolver uses this action on $v_1^\perp$ to recover the remaining dominant modes. Both implementations solve the spectral problem defined by (24). They use compact factors throughout, with C , A , and the sample Laplacian represented implicitly.

G. Complete Algorithm and Computational Complexity

Algorithm 1 combines representation learning with spectral clustering. Let T_{pre} and T be the numbers of pretraining updates and joint outer iterations, respectively. The spectral routine is given in Algorithm 2. It uses R_{km} clustering restarts with at most T_{km} Lloyd iterations per restart.

a) *Representation learning*: Let c_{net} bound the per-sample cost of network evaluation and backpropagation in one update, including the refreshed encoder evaluation. Pretraining takes $O(T_{\text{pre}}nc_{\text{net}})$ time. Seeding requires $O(nd_zm)$ time, and the initial Procrustes update requires $O(nd_zm + nm^2 + m^3)$. Each joint iteration takes

$$O(nc_{\text{net}} + nd_zm + nm^2 + m^3). \quad (34)$$

These terms account for network training, the products $Z^\top L$ and ZP , and the thin SVD. The representation cost is the sum of pretraining and initialization costs plus T times (34). Gradient accumulation allows the activation workspace to be controlled through the chunk size.

b) *Spectral computation*: Computing the degrees requires $O(nm)$ time. In the direct implementation, constructing the reduced matrix $G \in \mathbb{R}^{q \times q}$ costs $O(nq^2)$, and its eigendecomposition costs $O(q^3)$. Recovering the k -dimensional sample embedding requires $O(nqk)$ operations. The total complexity is therefore

$$O(nm + nq^2 + q^3 + nqk), \quad q = m(m+1)/2. \quad (35)$$

The dependence on n appears only in the construction of the reduced matrix and the recovery of the sample embedding. The eigendecomposition is performed in the q -dimensional space.

For the implicit implementation, the matrix G is not formed explicitly. Let T_{eig} denote the total number of matrix-vector products required by the iterative eigensolver, s_{eig} the maximum Krylov basis size, and R_{eig} the number of restarts. Including operator evaluations, orthogonalization, and the projected eigenproblems, the total complexity is

$$O(nm + T_{\text{eig}}(nq + qs_{\text{eig}}) + R_{\text{eig}}(qs_{\text{eig}}^2 + s_{\text{eig}}^3) + nqk). \quad (36)$$

Each vector processed by a block eigensolver contributes one matrix-vector product to T_{eig} . The quantities T_{eig} , s_{eig} , and R_{eig} depend on the convergence of the eigensolver and the required numerical accuracy. Since the implicit implementation evaluates products with G only when needed, it avoids storing the dense $q \times q$ matrix and is preferable when q is large.

c) *Clustering and memory*: The k -dimensional embedding gives a clustering cost of $O(R_{\text{km}}(T_{\text{km}}+1)nk^2)$. Computing latent centers and assigning zero-degree samples adds at most $O(nd_zk)$. Let M_{net} include network parameters, optimizer state and the activation workspace. A resident-data memory bound is

$$O(n(d_x + d_z + m + k) + d_zm + M_{\text{net}}), \quad (37)$$

plus $O(q^2 + qk)$ for direct reduction or $O(qs_{\text{eig}} + qk)$ for the implicit eigensolver. Quadratic feature rows can be evaluated as needed, so storing F is optional. The output labels require $O(n)$ memory.

Together, these bounds establish linear time and memory in n for fixed dimensions, architecture, factor width, cluster count and iteration budgets. The factor width m is a chosen model parameter, so both phases operate without estimating the data rank. Their shared use of compact factors makes scalability extend from representation learning to the final partition.

IV. RESULTS AND DISCUSSION

a) *Protocol*: We evaluate LoomSC on five image-clustering benchmarks. The face datasets are Extended YaleB (EYaleB), ORL, and UMIST, with $(n, k) = (2,432, 38)$, $(400, 40)$, and $(575, 20)$, respectively. We also use COIL-100 with 7,200 images from 100 object classes and a balanced subset of Fashion-MNIST with 5,000 images from 10 classes. Each sample is normalized to unit ℓ_2 norm. Tables I and IV specify the sample counts used in the comparisons.

We compare with nine baselines that distinguish deep representation learning from computational scalability. The *scalable non-deep methods* comprise two groups. LMVSC [15], SGL [16], and S⁵C [29] use sampling or anchor-based reduction. EnSC [19], SSC-OMP [18], A-DSSC [22], and SSSC [20] accelerate self-expression estimation without learning deep representations. These methods target scalability, but do not all achieve linear complexity for the complete clustering pipeline.

The *non-linear-time full-affinity references* are LSR [9] and PRO-DSC [28]. LSR is non-deep and computes a dense self-expression matrix. PRO-DSC is deep and supports mini-batch training. Its published spectral inference nevertheless constructs pairwise coefficients over the clustering set and processes the resulting affinity. Thus, scalable training does not eliminate its full-affinity spectral bottleneck. LoomSC combines deep representation learning with factorized self-expression and spectral assignment. Its complete pipeline has linear time and memory complexity in n for fixed architecture, dimensions, factor width, and iteration budgets.

A detailed comparison of computational and matrix-storage complexities is provided in Appendix F.

For LoomSC, we use the same architecture and optimization hyperparameters across all datasets. The encoder contains $L = 3$ convolutional layers with 10, 20, and 30 output channels and kernel sizes 5×5 , 3×3 , and 3×3 , respectively. Each layer uses stride 2 and a ReLU activation, with no batch normalization. The decoder mirrors the encoder using transposed convolutions. The resulting convolutional representation has size $30 \times 6 \times 6$, which gives a flattened latent dimension of $d_z = 1080$. We set the factor width to $m = 380$ for all datasets. The auto-encoder is first pretrained for 5,000 epochs using full-batch Adam with a learning rate of 10^{-3} . No data augmentation, weight decay, or learning-rate scheduling is used.

After pretraining, k -means with k -means++ initialization is applied to the latent representations, and its m centroids initialize the columns of L . The factor P is then initialized by the orthogonal Procrustes solution. Joint optimization is performed for another 5,000 epochs using full-batch Adam with a learning rate of 10^{-3} . The reconstruction and self-expression terms have equal weights. At each joint-training epoch, the network parameters are first updated by Adam.

TABLE I
CLUSTERING QUALITY — ACC (%) ON THE FIVE BENCHMARKS. HIGHER IS BETTER. MEAN $\pm$ STANDARD DEVIATION OVER THREE SEEDS WHERE AVAILABLE. BOLD INDICATES THE LARGEST MEAN IN EACH ROW.

Dataset	n	k	Non-deep methods								Deep methods	
			EnSC	SSC-OMP	LMVSC	SGL	S ⁵ C	LSR	A-DSSC	SSSC	PRO-DSC	LoomSC
Fashion-MNIST	5,000	10	61.35 $\pm$ 0.00	56.19 $\pm$ 1.36	59.18	54.47 $\pm$ 0.36	54.62 $\pm$ 1.99	59.52 $\pm$ 0.00	39.01 $\pm$ 2.20	42.01 $\pm$ 1.70	52.12 $\pm$ 2.92	65.33 $\pm$ 1.79
ORL	400	40	78.00 $\pm$ 1.59	73.75 $\pm$ 0.82	55.25 $\pm$ 0.00	62.25 $\pm$ 0.00	42.50 $\pm$ 0.00	82.50 $\pm$ 0.00	50.75 $\pm$ 0.00	46.50 $\pm$ 1.54	52.00 $\pm$ 7.91	86.17 $\pm$ 2.01
UMIST	575	20	72.52 $\pm$ 0.00	67.70 $\pm$ 0.00	60.62 $\pm$ 0.00	65.00 $\pm$ 0.00	40.87 $\pm$ 0.00	58.26 $\pm$ 0.00	52.52 $\pm$ 0.00	21.16 $\pm$ 1.31	81.39 $\pm$ 1.58	80.00 $\pm$ 2.19
EYaleB	2,432	38	88.19 $\pm$ 0.00	80.84 $\pm$ 1.34	21.50 $\pm$ 0.00	21.50 $\pm$ 0.00	22.41 $\pm$ 0.00	96.26 $\pm$ 0.00	61.88 $\pm$ 0.00	25.11 $\pm$ 2.27	77.63 $\pm$ 0.73	94.73 $\pm$ 1.26
COIL-100	7,200	100	67.50 $\pm$ 0.69	59.46 $\pm$ 0.00	49.50 $\pm$ 0.00	49.26 $\pm$ 0.00	42.04 $\pm$ 0.00	61.10 $\pm$ 0.00	48.57 $\pm$ 0.00	13.81 $\pm$ 1.31	82.31 $\pm$ 0.30	74.64 $\pm$ 1.34

TABLE II
CLUSTERING QUALITY — NMI (%) ON THE FIVE BENCHMARKS. HIGHER IS BETTER. MEAN $\pm$ STANDARD DEVIATION OVER THREE SEEDS. BOLD INDICATES THE LARGEST MEAN IN EACH ROW.

Dataset	Non-deep methods								Deep methods	
	EnSC	SSC-OMP	LMVSC	SGL	S ⁵ C	LSR	A-DSSC	SSSC	PRO-DSC	LoomSC
Fashion-MNIST	60.29 $\pm$ 0.88	57.93 $\pm$ 0.41	64.37 $\pm$ 0.63	62.51 $\pm$ 0.42	54.82 $\pm$ 2.33	63.33 $\pm$ 0.00	53.83 $\pm$ 1.00	46.88 $\pm$ 0.45	56.45 $\pm$ 3.08	63.91 $\pm$ 0.82
ORL	89.25 $\pm$ 0.07	84.54 $\pm$ 0.37	74.53 $\pm$ 0.00	78.24 $\pm$ 0.00	64.76 $\pm$ 0.00	90.07 $\pm$ 0.00	77.93 $\pm$ 0.00	66.84 $\pm$ 1.34	71.71 $\pm$ 5.39	91.76 $\pm$ 0.88
UMIST	87.06 $\pm$ 0.00	63.61 $\pm$ 0.53	62.49 $\pm$ 0.00	56.84 $\pm$ 0.00	56.72 $\pm$ 0.00	75.40 $\pm$ 0.00	66.01 $\pm$ 0.00	32.89 $\pm$ 1.40	90.02 $\pm$ 0.20	87.83 $\pm$ 1.44
EYaleB	74.31 $\pm$ 0.03	83.17 $\pm$ 0.55	13.37 $\pm$ 0.00	6.44 $\pm$ 0.00	37.92 $\pm$ 0.00	95.44 $\pm$ 0.00	75.61 $\pm$ 0.00	34.24 $\pm$ 2.42	82.38 $\pm$ 0.91	95.47 $\pm$ 0.43
COIL-100	90.95 $\pm$ 0.02	57.11 $\pm$ 0.16	74.69 $\pm$ 0.00	74.54 $\pm$ 0.00	68.03 $\pm$ 0.00	85.21 $\pm$ 0.00	74.31 $\pm$ 0.00	37.80 $\pm$ 1.08	95.75 $\pm$ 0.05	90.95 $\pm$ 0.41

TABLE III
CLUSTERING QUALITY — ARI (%) ON THE FIVE BENCHMARKS. HIGHER IS BETTER. MEAN $\pm$ STANDARD DEVIATION OVER THREE SEEDS. BOLD INDICATES THE LARGEST MEAN IN EACH ROW.

Dataset	Non-deep methods								Deep methods	
	EnSC	SSC-OMP	LMVSC	SGL	S ⁵ C	LSR	A-DSSC	SSSC	PRO-DSC	LoomSC
Fashion-MNIST	43.21 $\pm$ 1.04	42.00 $\pm$ 0.16	45.35 $\pm$ 0.87	42.75 $\pm$ 0.22	40.27 $\pm$ 1.50	46.53 $\pm$ 0.00	22.24 $\pm$ 1.29	19.41 $\pm$ 1.49	37.76 $\pm$ 4.07	51.89 $\pm$ 1.43
ORL	69.50 $\pm$ 0.55	60.22 $\pm$ 0.74	39.22 $\pm$ 0.00	47.60 $\pm$ 0.00	22.66 $\pm$ 0.00	74.10 $\pm$ 0.00	21.49 $\pm$ 0.00	15.32 $\pm$ 4.98	35.55 $\pm$ 8.75	78.05 $\pm$ 2.67
UMIST	66.91 $\pm$ 0.00	34.81 $\pm$ 0.47	32.12 $\pm$ 0.00	25.19 $\pm$ 0.00	26.63 $\pm$ 0.00	46.89 $\pm$ 0.00	22.99 $\pm$ 0.00	1.07 $\pm$ 0.31	79.36 $\pm$ 0.65	75.55 $\pm$ 2.69
EYaleB	19.67 $\pm$ 0.06	53.91 $\pm$ 5.11	1.87 $\pm$ 0.00	-0.40 $\pm$ 0.00	13.65 $\pm$ 0.00	92.19 $\pm$ 0.00	21.67 $\pm$ 0.00	2.16 $\pm$ 0.31	64.15 $\pm$ 1.28	91.76 $\pm$ 0.90
COIL-100	61.38 $\pm$ 1.84	21.98 $\pm$ 0.13	43.00 $\pm$ 0.00	43.93 $\pm$ 0.00	35.05 $\pm$ 0.00	54.03 $\pm$ 0.00	7.84 $\pm$ 0.00	1.59 $\pm$ 0.04	80.63 $\pm$ 1.28	66.60 $\pm$ 0.66

The factor P is then updated by the exact Procrustes solution, followed by the closed-form update $L = ZP$, as described in Algorithm 1. After joint optimization, spectral assignment is performed following Algorithm 2. The spectral embedding dimension is set to the target number of clusters k , after which k -means produces the final partition.

We report clustering accuracy (ACC), normalized mutual information (NMI), and adjusted Rand index (ARI) as percentages. Higher values indicate better clustering. Repeated-run results are summarized by their mean and standard deviation over three seeds. A scalar entry indicates that no standard deviation is reported. Differences between percentage scores are expressed in percentage points. The synthetic experiments additionally report the connectivity diagnostic (CONN) [34], for which higher values are preferred.

A. Clustering quality

Tables I–III show that LoomSC ranks first or second in all 15 dataset–metric comparisons, including a shared second place for COIL-100 NMI. No other evaluated method maintains this ranking across all five datasets and three metrics. Its unweighted means are 80.17% ACC, 85.98% NMI, and 72.77% ARI. These exceed the highest corresponding baseline means by 6.66, 4.09, and 10.02 points. Thus, the compact formulation provides consistently strong clustering quality while retaining linear-in- n computation under the conditions stated above.

LoomSC achieves higher ACC than every scalable non-deep baseline on all five datasets. The improvements over the strongest method in this group range from 3.98 to 8.17 points. The gains over sampling-based methods are larger. Relative to the best ACC among LMVSC, SGL, and S⁵C, LoomSC improves by 23.92 points on ORL and 25.14 points on COIL-100. The corresponding gains are 15.00 points on UMIST and 72.32 points on EYaleB. These comparisons show that a compact model need not be restricted to the clustering quality of fixed-representation sampling methods. LoomSC retains compactness while jointly refining the latent features and their self-expression factors.

On Fashion-MNIST, LoomSC obtains the highest ACC of 65.33 $\pm$ 1.79% and ARI of 51.89 $\pm$ 1.43%. These exceed the strongest competing results by 3.98 and 5.36 points, respectively. Its NMI of 63.91 $\pm$ 0.82% is within 0.46 points of the highest value. The agreement between ACC and ARI supports the quality of the partition beyond a single evaluation criterion.

On ORL, LoomSC leads all three metrics with 86.17 $\pm$ 2.01% ACC, 91.76 $\pm$ 0.88% NMI, and 78.05 $\pm$ 2.67% ARI. It improves upon LSR, the strongest alternative for each metric, by 3.67, 1.69, and 3.95 points. These results are obtained with only ten images per subject. On EYaleB, LoomSC reaches 94.73% ACC, 95.47% NMI, and 91.76% ARI. Its NMI closely matches LSR at 95.44%, and their ARI scores differ by only 0.43 points.

TABLE IV
MEASURED WALL-CLOCK TIME (S). LOWER IS BETTER. BOLD INDICATES THE SMALLEST MEAN IN EACH ROW.

Dataset	n	Non-deep methods								Deep methods	
		EnSC	SSC-OMP	LMVSC	SGL	S ² C	LSR	A-DSSC	SSSC	PRO-DSC	LoomSC
Fashion-MNIST	5,000	3219.752 ± 46.948	5.447 ± 0.681	16.965 ± 0.179	17.203 ± 0.219	8.280 ± 1.000	6.260 ± 0.157	35.612 ± 0.584	2231.314 ± 69.877	563.338 ± 1.392	9.008 ± 0.987
ORL	400	7.953 ± 0.514	1.856 ± 0.058	0.929 ± 0.031	0.958 ± 0.003	7.440 ± 0.419	0.049 ± 0.001	0.699 ± 0.005	7.084 ± 0.183	102.797 ± 2.671	0.239 ± 0.089
UMIST	575	24.366 ± 26.899	0.498 ± 0.012	1.086 ± 0.033	1.263 ± 0.043	10.249 ± 4.946	0.077 ± 0.002	1.026 ± 0.027	9.774 ± 1.423	78.689 ± 0.172	0.203 ± 0.003
EYaleB	2,432	43.843 ± 0.398	12.666 ± 0.975	9.124 ± 0.063	9.221 ± 0.083	471.768 ± 24.647	2.389 ± 0.542	13.918 ± 0.132	2286.421 ± 94.488	313.648 ± 2.670	2.240 ± 0.581
COIL-100	7,200	50.858 ± 1.523	1319.586 ± 164.863	21.421 ± 0.143	22.498 ± 0.390	4709.568 ± 169.023	49.888 ± 5.451	175.328 ± 1.551	2409.599 ± 245.492	7443.627 ± 4.992	25.311 ± 9.976

Thus, the thin-factor representation can match or improve upon dense least-squares self-expression on several clustering measures without storing all pairwise coefficients.

The comparison with PRO-DSC assesses whether this efficiency is compatible with deep clustering quality. LoomSC exceeds PRO-DSC on all three metrics for Fashion-MNIST, ORL, and EYaleB. The ACC gains are 13.21, 34.17, and 17.10 points, respectively. Across the five datasets, its average improvements over PRO-DSC are 11.08 ACC points, 6.72 NMI points, and 13.28 ARI points. Therefore, the advantage over the non-deep baselines is accompanied by a strong comparison with a deep full-affinity method.

PRO-DSC attains the highest scores on UMIST and COIL-100, where LoomSC ranks second on all three metrics. On UMIST, LoomSC reaches 80.00% ACC compared with 81.39% for PRO-DSC. On COIL-100, it obtains 74.64% ACC, 90.95% NMI, and 66.60% ARI. Its COIL-100 ACC exceeds every non-deep baseline, with gains of 7.14 points over EnSC and 13.54 points over LSR. These results establish competitive deep clustering performance on both datasets. As quantified next, LoomSC attains this performance at a small fraction of PRO-DSC’s measured runtime and without its non-linear-time full-affinity spectral stage.

B. Computational efficiency

Table IV reports the measured wall-clock times. LoomSC requires 0.239 s on ORL and 0.203 s on UMIST. Its mean time is 2.240 s on EYaleB, 9.008 s on Fashion-MNIST, and 25.311 s on COIL-100. No competing method combines higher mean ACC with lower mean runtime on any of these datasets. Thus, LoomSC lies on the empirical ACC–runtime Pareto frontier for every benchmark among the evaluated implementations.

The distinction from PRO-DSC concerns the complete clustering pipeline rather than network training alone. PRO-DSC reduces training costs through mini-batches, but its published inference constructs a full pairwise affinity before spectral clustering [28]. Explicitly storing this affinity requires $O(n^2)$ memory for a clustering set of size n . LoomSC instead retains $P \in \mathbb{R}^{n \times m}$ and $L \in \mathbb{R}^{d_z \times m}$ during representation learning. Its quadratic feature map transfers spectral computation to dimension $q = m(m+1)/2$, which is independent of n for fixed m . Consequently, it removes the full-affinity bottleneck rather than only accelerating its construction. Section III-G establishes linear time and memory in n under fixed dimensions and iteration budgets.

The reported runtimes show substantial savings relative to the deep baseline. LoomSC is 62.5× faster than PRO-DSC on Fashion-MNIST while improving ACC by 13.21 points. On

ORL, it is 430.1× faster and improves ACC by 34.17 points. On EYaleB, the corresponding speedup is 140.0×, together with a 17.10-point ACC gain. Hence, on three datasets, the computational reduction accompanies higher clustering quality rather than a loss of accuracy.

The datasets where PRO-DSC achieves its strongest results further illustrate the scalability advantage. On UMIST, LoomSC attains 80.00% ACC in 0.203 s, compared with 81.39% in 78.689 s for PRO-DSC. This is a 387.6× speedup with an ACC difference of 1.39 points. On COIL-100, LoomSC obtains the second-highest ACC while reducing the measured time from 7,443.627 to 25.311 s, a 294.1× speedup. Across all five datasets, the speedups over PRO-DSC range from 62.5× to 430.1×. Therefore, competitive deep clustering performance is retained without the runtime of the full-affinity reference.

LoomSC also improves the quality–cost balance relative to scalable non-deep methods. On Fashion-MNIST, it exceeds EnSC by 3.98 ACC points with a 357.4× speedup. On COIL-100, it improves ACC over EnSC and LSR by 7.14 and 13.54 points while approximately halving their runtimes. Compared with LMVSC, the fastest method on COIL-100, LoomSC requires only 3.89 additional seconds and improves ACC by 25.14 points. These results show that LoomSC combines runtimes close to or below those of sampling methods with substantially stronger clustering. Its compact pipeline also remains far faster than the deep full-affinity reference.

C. Scalability Analysis

We first vary the sampling density in a controlled union-of-subspaces setting. Following the synthetic-data configuration in [20], we generate $k = 10$ subspaces with random dimensions between 6 and 12 in an ambient space of dimension $d_x = 784$. Each subspace contains $n_i \in \{30, 50, 100, 200, 500, 1000\}$ samples. The total sample count is therefore $n = 10n_i$, ranging from 300 to 10,000. Figure 1 compares ACC, connectivity, and CPU and GPU runtimes.

LoomSC maintains 99%–100% ACC across the entire density range. Its connectivity remains high, at 0.68–0.75. Several competing methods attain high ACC at sufficiently large sample counts but have much smaller connectivity scores. Thus, the two diagnostics provide complementary evidence: LoomSC produces accurate partitions while retaining strong graph connectivity under the reported measure. This behavior is consistent with the goal of preserving informative sample relations through the factorized representation.

The comparison with the SDSNet [35] reference is particularly informative at low sampling densities. With 30 samples per subspace, LoomSC reaches 99% ACC, compared with

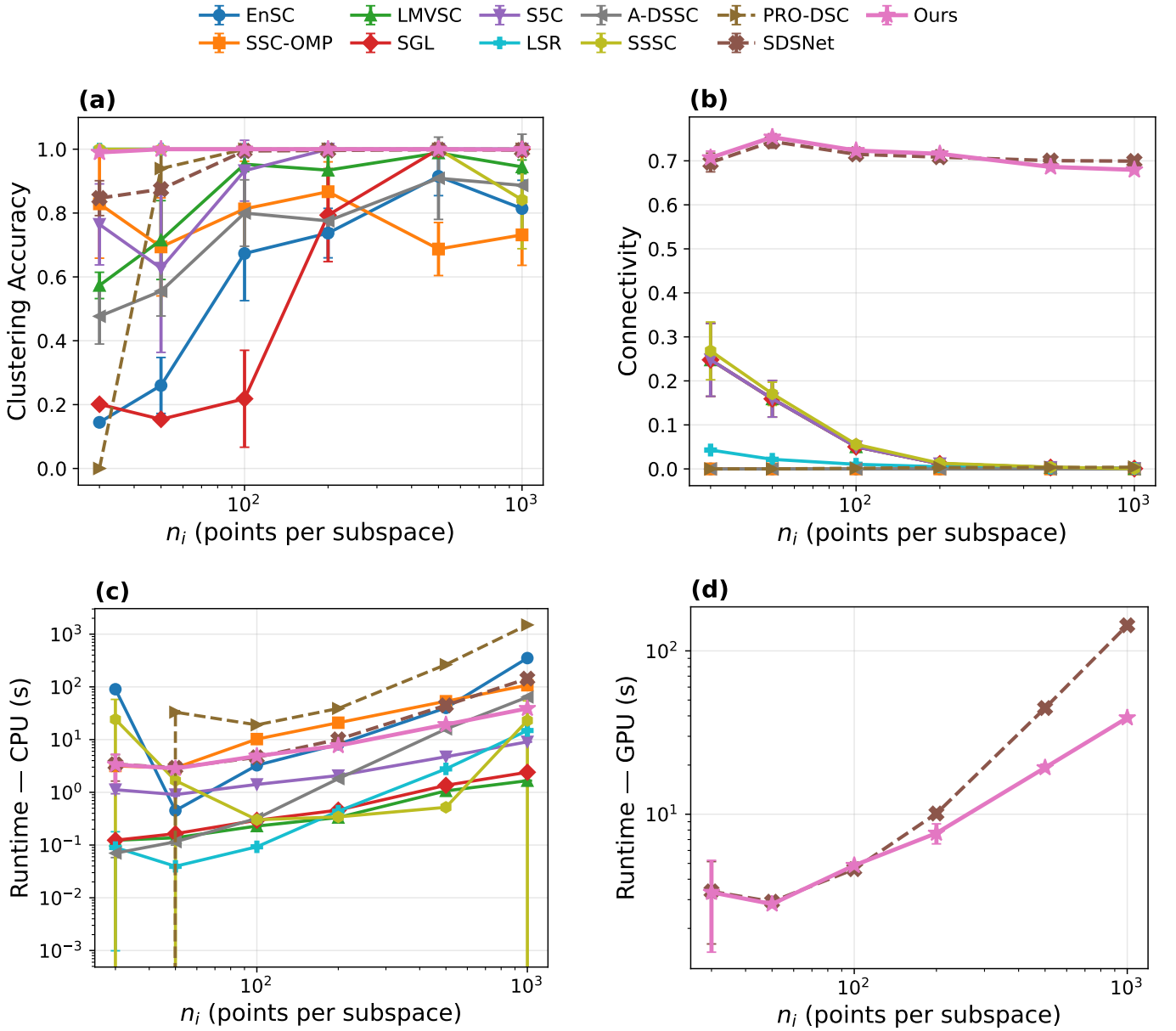


Fig. 1. Synthetic-data comparison with ten subspaces and n_i samples per subspace. Panels show (a) ACC, (b) connectivity, (c) CPU runtime, and (d) GPU runtime. “Ours” denotes LoomSC.

about 85% for SDSNet. With 50 samples per subspace, its ACC is 100%, compared with about 87%. At the largest size, the two approaches both retain near-perfect accuracy and high connectivity. LoomSC nevertheless reduces the measured runtime from 143 to 39 s in both the CPU and GPU panels. The improvement is visible within each hardware setting rather than arising only from a CPU-to-GPU comparison.

The same experiment contrasts LoomSC with the full-affinity deep baseline. PRO-DSC also reaches near-perfect ACC at higher sampling densities. However, its CPU runtime is substantially larger at the highest density. LoomSC combines high clustering accuracy across the full density range with lower computational cost at the largest size in this experiment.

Figure 2 extends the sample-size experiment from 1,000 to 500,000 points. LoomSC maintains 99.8% ACC or higher

throughout this 500-fold increase. Its runtime grows from about 57 s at 50,000 samples to 362 s at 500,000 samples. At the largest size, it retains near-perfect accuracy in approximately six minutes. S⁵C reaches comparable accuracy but requires about 1,054 s. LMVSC has a lower runtime of about 125 s, with ACC near 94%. SGL also reaches near-perfect ACC at large sizes, whereas LoomSC maintains this accuracy regime across the full tested range, including the smallest datasets.

The large-sample runtime trend is consistent with the linear-in- n bounds in Section III-G. For fixed factor width, the quadratic feature dimension $q = m(m+1)/2$ does not grow with the number of samples. Neither factor learning nor the reduced spectral stage requires storage of all pairwise affinities. With fixed dimensions and iteration budgets, the sample-dependent work remains linear. The experiment shows

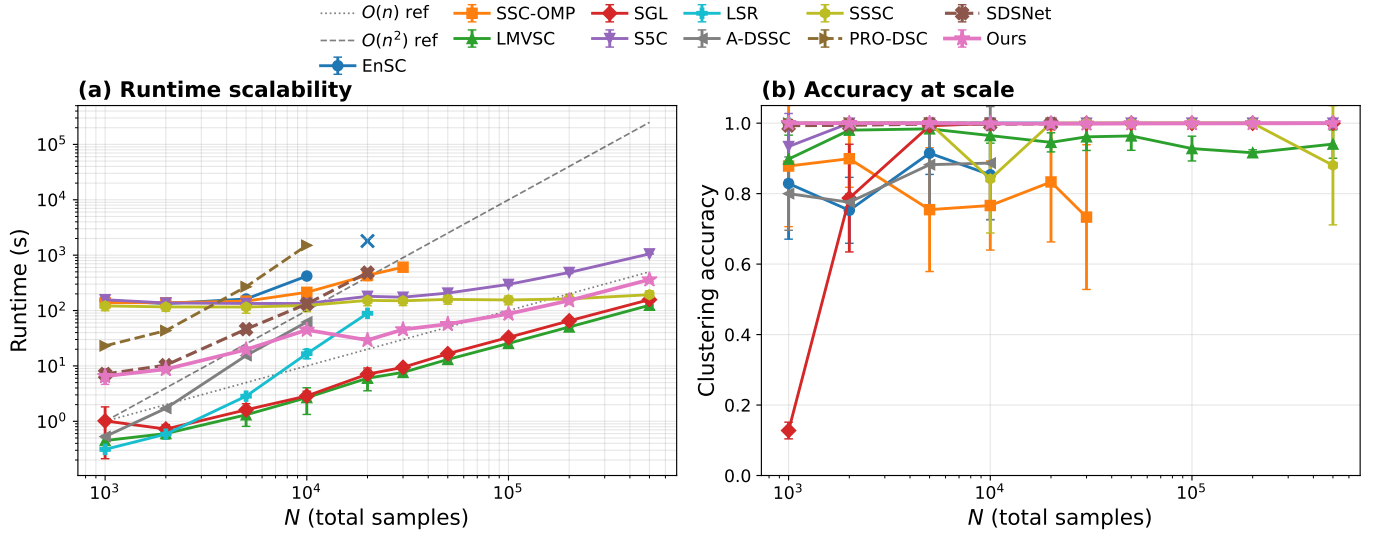


Fig. 2. Synthetic-data scalability from 1,000 to 500,000 samples: (a) runtime and (b) clustering accuracy. The horizontal-axis symbol N denotes the total sample count n . The reference curves indicate linear and quadratic growth. “Ours” denotes LoomSC.

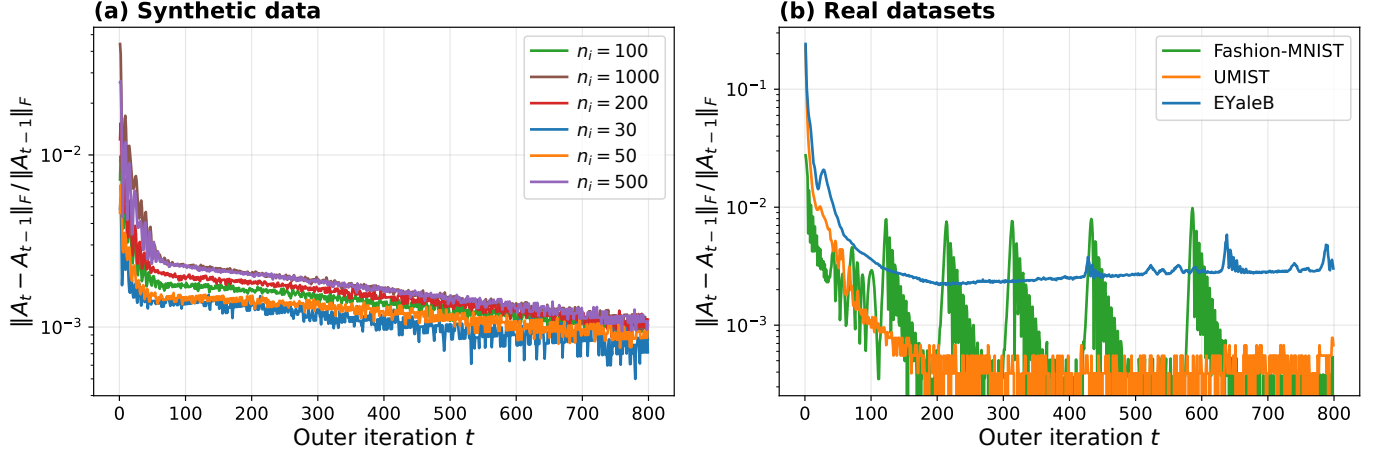


Fig. 3. Relative Frobenius change $\|A_t - A_{t-1}\|_F / \|A_{t-1}\|_F$ on (a) synthetic datasets with ten subspaces and (b) real datasets. Here, A_t denotes the affinity matrix in (24) at outer iteration t , and n_i is the number of samples in synthetic subspace i .

that this design supports increasing the dataset size without a corresponding loss of clustering accuracy.

D. Convergence of the Affinity Matrix

Figure 3 examines the relative Frobenius change between successive affinity iterates. The diagnostic divides the Frobenius norm of the difference by the norm of the previous iterate. This difference measures the stability of the sample relationships used for clustering, rather than the scale of the matrix alone.

On synthetic data, all six sampling densities exhibit a short initial transient followed by gradual stabilization. The relative change falls to a few 10^{-3} within roughly the first 100 outer iterations and approaches 10^{-3} by the end of the run. The curves reach similar terminal scales despite the increase from 30 to 1,000 samples per subspace. Thus, the larger synthetic datasets retain the same qualitative stabilization pattern.

The real-data curves show similarly small late-stage changes. UMIST falls below 10^{-3} after about 100 iterations. Fashion-

MNIST reaches a low baseline with temporary excursions, while EYaleB settles around a few 10^{-3} . The dominant pattern is a rapid reduction in the update magnitude followed by sustained small changes. This supports the practical stability of the learned affinities during joint training. Proposition 3 provides a complementary descent result for the exact factor updates with fixed latent features. The observed curves show that small affinity changes are also attained when these updates are coupled with representation learning.

E. Affinity-matrix structure

Figures 4–6 visualize the learned sample relations at epochs 10, 100, and 500 after pretraining. The face-data panels use ten-class subsets, and the synthetic panels contain ten subspaces. Samples are grouped by their classes, with class boundaries overlaid. These plots examine whether the representation retains class-aligned relations throughout optimization.

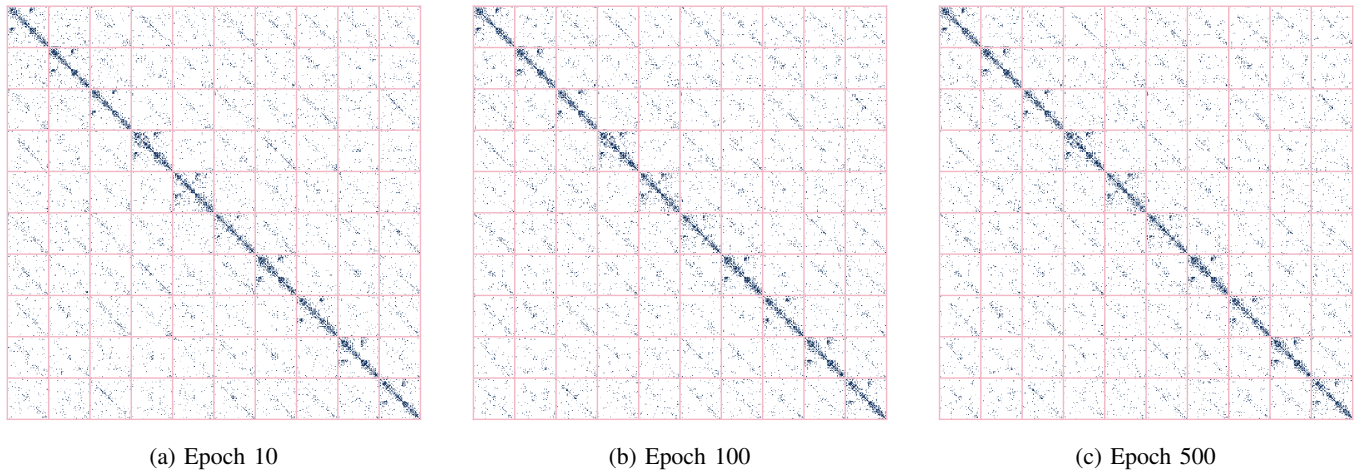


Fig. 4. Learned sample relations on a ten-class EYaleB subset at epochs 10, 100, and 500. Class boundaries are overlaid.

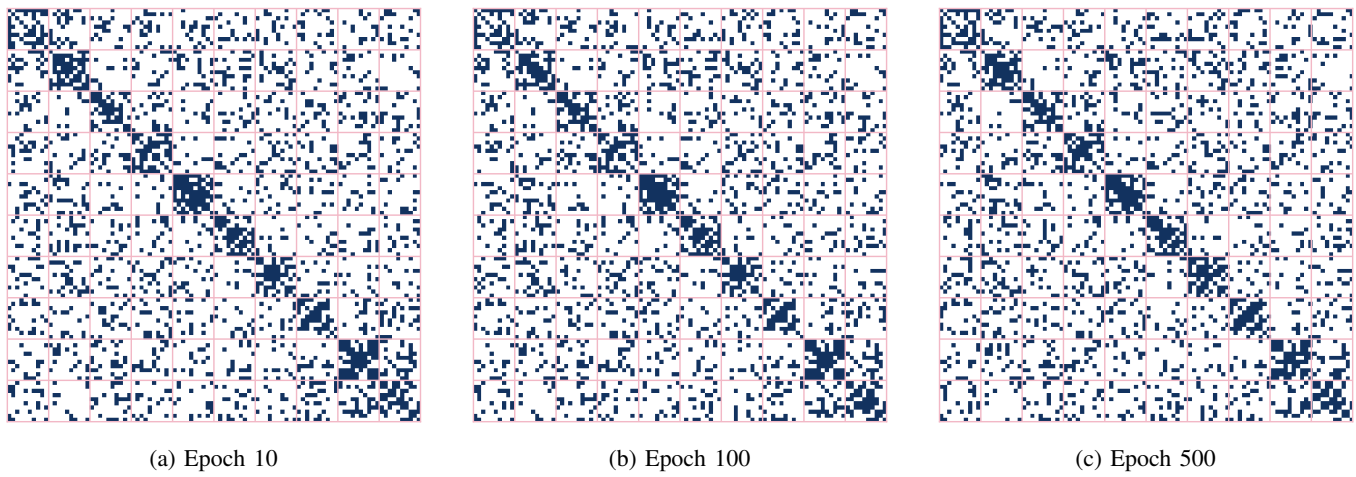


Fig. 5. Learned sample relations on a ten-class ORL subset at epochs 10, 100, and 500. Class boundaries are overlaid.

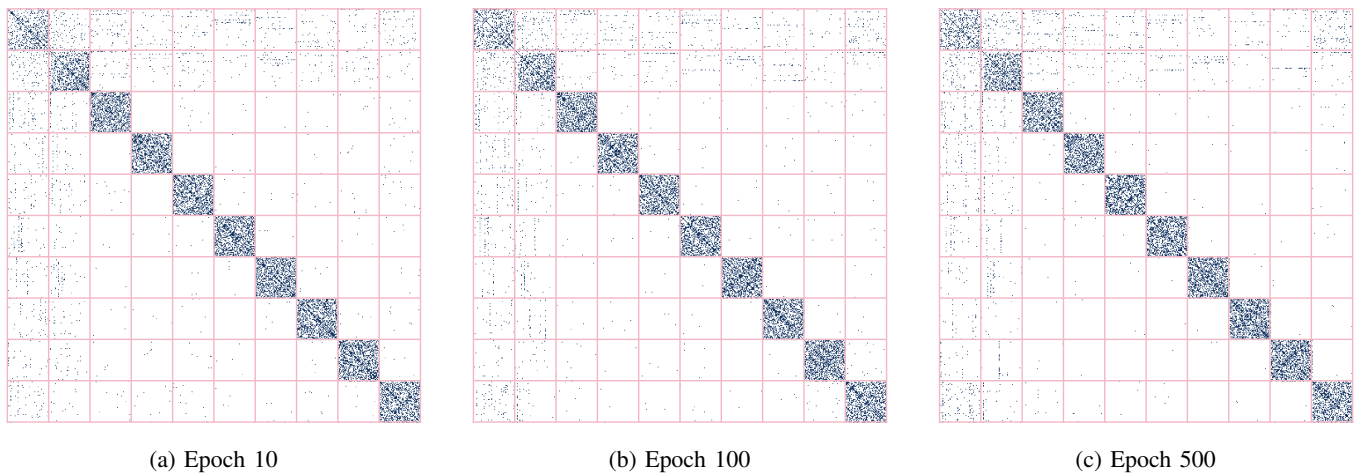


Fig. 6. Learned sample relations on synthetic data with ten subspaces at epochs 10, 100, and 500. Subspace boundaries are overlaid.

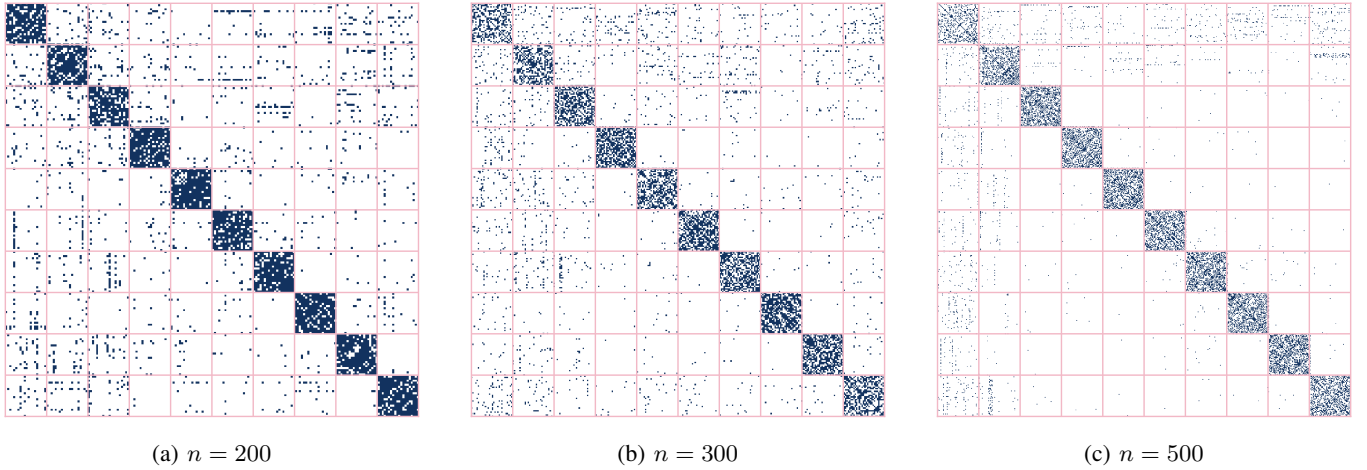


Fig. 7. Learned sample relations on synthetic data with ten subspaces and increasing total sample count at epoch 500. Subspace boundaries are overlaid.

On EYaleB, the diagonal blocks are already visible after 10 epochs and remain clearly identifiable at later epochs. The within-class regions contain more pronounced structure than most off-block regions. ORL also retains recognizable diagonal blocks despite having only ten images per subject. This visual organization is consistent with the strong face-clustering scores in Tables I–III. In particular, the displayed relations extend beyond individual self-connections and include associations between distinct samples from the same class.

The synthetic panels exhibit an even clearer separation between dense diagonal blocks and the surrounding regions. This organization persists across the training stages (i.e., 10, 100, and 500 epochs). Figure 7 compares $n = 200$, 300, and 500 at epoch 500. The 10 diagonal blocks remain identifiable as more samples are added, and the larger panels show a clearer visual contrast between within-subspace structure and off-block regions. Together with the quantitative connectivity results, these visualizations support the ability of the affinity matrix to retain structured sample relations as the dataset grows.

V. CONCLUSION

This work presents LoomSC, a scalable subspace clustering framework that combines projector-based self-expression with exact spectral reduction. Jointly learning latent representations and two thin factors avoids explicit sample-by-sample coefficient matrices. A nonnegative quadratic affinity preserves the projector’s support and enables normalized spectral clustering without forming the full affinity or sample Laplacian. For fixed dimensions and iteration budgets, the complete pipeline has linear time and memory complexity in the number of samples. Our analysis quantifies the projector approximation to regularized least squares. It also identifies conditions under which the full row-space projector preserves subspace memberships and ensures within-subspace connectivity.

Across five image-clustering benchmarks, LoomSC achieves the highest average ACC, NMI, and ARI among the evaluated methods. Synthetic experiments extend to 500,000 samples while maintaining at least 99.8% accuracy throughout the tested range. These findings demonstrate that compact self-expression

can combine strong clustering quality with scalability across both representation learning and spectral assignment.

REFERENCES

- [1] Y. Ma, H. Derksen, W. Hong, and J. Wright, “Segmentation of multivariate mixed data via lossy data coding and compression,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 9, pp. 1546–1562, 2007.
- [2] S. Rao, R. Tron, R. Vidal, and Y. Ma, “Motion segmentation in the presence of outlying, incomplete, or corrupted trajectories,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 10, pp. 1832–1845, 2010.
- [3] R. Vidal, “Subspace clustering,” *IEEE Signal Process. Mag.*, vol. 28, no. 2, pp. 52–68, 2011.
- [4] T. Ding, D. Lim, R. Vidal, and B. D. Haeffele, “Understanding doubly stochastic clustering,” in *Proc. Int. Conf. Mach. Learn. (ICML)*, ser. Proceedings of Machine Learning Research, vol. 162, 2022, pp. 5153–5165.
- [5] R. Basri and D. W. Jacobs, “Lambertian reflectance and linear subspaces,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 2, pp. 218–233, 2003.
- [6] C. Lu, J. Feng, Z. Lin, T. Mei, and S. Yan, “Subspace clustering by block diagonal representation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 2, pp. 487–501, 2019.
- [7] E. Elhamifar and R. Vidal, “Sparse subspace clustering,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2009, pp. 2790–2797.
- [8] —, “Sparse subspace clustering: Algorithm, theory, and applications,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 11, pp. 2765–2781, 2013.
- [9] C.-Y. Lu, H. Min, Z.-Q. Zhao, L. Zhu, D.-S. Huang, and S. Yan, “Robust and efficient subspace segmentation via least squares regression,” in *Eur. Conf. Comput. Vis. (ECCV)*, ser. Lecture Notes in Computer Science, vol. 7578, 2012, pp. 347–360.
- [10] G. Liu, Z. Lin, and Y. Yu, “Robust subspace segmentation by low-rank representation,” in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2010, pp. 663–670.
- [11] D. Luo, F. Nie, C. Ding, and H. Huang, “Multi-subspace representation and discovery,” in *Machine Learning and Knowledge Discovery in Databases*, ser. Lecture Notes in Computer Science, vol. 6912. Springer, 2011, pp. 405–420.
- [12] B. Nasihatkon and R. Hartley, “Graph connectivity in sparse subspace clustering,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2011, pp. 2137–2144.
- [13] X. Peng, L. Zhang, and Z. Yi, “Scalable sparse subspace clustering,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2013, pp. 430–437.
- [14] X. Peng, H. Tang, L. Zhang, Z. Yi, and S. Xiao, “A unified framework for representation-based subspace clustering of out-of-sample and large-scale data,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 12, pp. 2499–2512, 2016.
- [15] Z. Kang, W. Zhou, Z. Zhao, J. Shao, M. Han, and Z. Xu, “Large-scale multi-view subspace clustering in linear time,” in *Proc. AAAI Conf. Artif. Intell. (AAAI)*, vol. 34, no. 4, 2020, pp. 4412–4419.

- [16] Z. Kang, Z. Lin, X. Zhu, and W. Xu, “Structured graph learning for scalable subspace clustering: From single view to multiview,” *IEEE Trans. Cybern.*, vol. 52, no. 9, pp. 8976–8986, 2022.
- [17] S. Wang, B. Tu, C. Xu, and Z. Zhang, “Exact subspace clustering in linear time,” in *Proc. AAAI Conf. Artif. Intell. (AAAI)*, vol. 28, no. 1, 2014, pp. 2113–2120.
- [18] C. You, D. P. Robinson, and R. Vidal, “Scalable sparse subspace clustering by orthogonal matching pursuit,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 3918–3927.
- [19] C. You, C.-G. Li, D. P. Robinson, and R. Vidal, “Oracle based active set algorithm for scalable elastic net subspace clustering,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 3928–3937.
- [20] Y. Chen, C.-G. Li, and C. You, “Stochastic sparse subspace clustering,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2020, pp. 4155–4164.
- [21] A. Adler, M. Elad, and Y. Hel-Or, “Linear-time subspace clustering via bipartite graph modeling,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 10, pp. 2234–2246, 2015.
- [22] D. Lim, R. Vidal, and B. D. Haeffele, “Doubly stochastic subspace clustering,” arXiv:2011.14859, 2020.
- [23] S. Zhang, C. You, R. Vidal, and C.-G. Li, “Learning a self-expressive network for subspace clustering,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2021, pp. 12 393–12 403.
- [24] Z. Kang, H. Xu, B. Wang, H. Zhu, and Z. Xu, “Clustering with similarity preserving,” *Neurocomputing*, vol. 365, pp. 211–218, 2019.
- [25] P. Ji, T. Zhang, H. Li, M. Salzmann, and I. Reid, “Deep subspace clustering networks,” in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 30, 2017, pp. 24–33.
- [26] J. Zhang, C.-G. Li, C. You, X. Qi, H. Zhang, J. Guo, and Z. Lin, “Self-supervised convolutional subspace clustering network,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2019, pp. 5473–5482.
- [27] J. Lv, Z. Kang, X. Lu, and Z. Xu, “Pseudo-supervised deep subspace clustering,” *IEEE Trans. Image Process.*, vol. 30, pp. 5252–5263, 2021.
- [28] X. Meng, Z. Huang, W. He, X. Qi, R. Xiao, and C.-G. Li, “Exploring a principled framework for deep subspace clustering,” in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2025.
- [29] S. Matsushima and M. Brbic, “Selective sampling-based scalable sparse subspace clustering,” in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 32, 2019.
- [30] C. Fattal, L. Labiod, and M. Nadif, “Scalable attributed-graph subspace clustering,” in *Proc. AAAI Conf. Artif. Intell. (AAAI)*, vol. 37, no. 6, 2023, pp. 7559–7567.
- [31] D. Arthur and S. Vassilvitskii, “k-means++: The advantages of careful seeding,” in *Proc. 18th Annu. ACM-SIAM Symp. Discrete Algorithms (SODA)*, 2007, pp. 1027–1035.
- [32] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2015.
- [33] A. Y. Ng, M. I. Jordan, and Y. Weiss, “On spectral clustering: Analysis and an algorithm,” in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 14, 2001, pp. 849–856.
- [34] Y. Chen, C.-G. Li, and C. You, “Stochastic sparse subspace clustering,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2020, pp. 4154–4163.
- [35] N. Mrabah, M. Bouguessa, and S. Sami, “Scalable deep subspace clustering network,” in *IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 2025.

APPENDIX A

SPECTRAL ANALYSIS OF THE LSR APPROXIMATION

The normal equation of (2) is

$$(X^\top X + 2\lambda I_n)C = X^\top X. \quad (38)$$

The coefficient matrix is positive definite for $\lambda > 0$, so the solution is unique. Since $X^\top X = V_x \text{diag}(\beta_1, \dots, \beta_r) V_x^\top$, the vectors in V_x diagonalize the solution on the row space of X . Along its j th eigenvector direction, the normal equation becomes $(\beta_j + 2\lambda)\tau_j = \beta_j$. In the orthogonal complement, the solution has an eigenvalue of zero. This proves Proposition 1.

The LSR solution and C_{row} have the same eigenvectors on the row space. The invariance of the Frobenius norm under orthonormal changes of coordinates therefore gives

$$\|C_{\text{LSR}}^* - C_{\text{row}}\|_F^2 = \sum_{j=1}^r (1 - \tau_j)^2, \quad (39)$$

which proves (6). For the rank- m projector $V_{x,m} V_{x,m}^\top$, the first m eigenvalues are one and the remaining eigenvalues are zero. Splitting the squared difference over these two sets proves (7). These identities quantify the spectral approximation used to motivate the factor model.

APPENDIX B

PROOFS FOR THE FACTOR MODEL

For fixed $Z \in \mathbb{R}^{d_z \times n}$ and feasible $P \in \mathbb{R}^{n \times m}$, decompose the residual as

$$Z - LP^\top = Z(I_n - PP^\top) + (ZP - L)P^\top. \quad (40)$$

The two terms are orthogonal in the Frobenius inner product because $(I_n - PP^\top)P = 0$. Moreover, $\|(ZP - L)P^\top\|_F^2 = \|ZP - L\|_F^2$ since $P^\top P = I_m$. Taking squared norms proves (14). The second term in that identity is strictly convex in L . Thus, $L = ZP$ is its unique minimizer. The equality $LP^\top = ZPP^\top$ holds after this update.

For fixed L , expansion gives (20). Let $Z^\top L = U_P \Sigma_P V_P^\top$ have the dimensions specified in (21). The trace inequality yields

$$\text{tr}(P^\top Z^\top L) \leq \text{tr}(\Sigma_P). \quad (41)$$

The feasible matrix $P = U_P V_P^\top$ attains equality. This proves the global optimality of the Procrustes update. When $Z^\top L$ is rank deficient, zero singular directions must be included so that both singular bases have m columns. Different orthonormal completions can yield different minimizers.

To prove Proposition 2, eliminate L and expand the remaining objective:

$$\|Z - ZPP^\top\|_F^2 = \|Z\|_F^2 - \text{tr}(P^\top Z^\top ZP). \quad (42)$$

The variational characterization of eigenvalues maximizes the trace over an m -dimensional dominant eigenspace of $Z^\top Z$. Its eigenvalues are $\sigma_1^2, \dots, \sigma_{r_z}^2$, followed by zeros. Subtracting their largest m values from $\|Z\|_F^2$ gives (16). When $m < r_z$, a spectral tie at the truncation boundary can make the optimal subspace nonunique.

When $m = r_z$, the minimum residual is zero. Hence, every global minimizer satisfies $ZPP^\top = Z$, which implies

$\text{range}(Z^\top) \subseteq \text{range}(P)$. Both spaces have dimension r_z , so they coincide. Consequently, $C = PP^\top = V_z V_z^\top = Z^\dagger Z$ is the unique row-space projector, where $\dagger$ denotes the Moore–Penrose pseudoinverse. Its orthonormal basis P remains nonunique. If $m > r_z$, the same inclusion holds, but $\text{range}(P)$ also contains $m - r_z$ directions orthogonal to the row space. This proves the remaining assertions of Proposition 2.

For Proposition 3, each exact block update minimizes the same fixed- Z objective with its other block held constant. Neither update can increase that objective. Its successive values are nonnegative and nonincreasing, so they converge.

APPENDIX C

SUBSPACE PRESERVATION AND CONNECTIVITY

Reorder the samples by subspace and write $Z = [Z_1, \dots, Z_k]$, where $Z_a \in \mathbb{R}^{d_z \times n_a}$, $\sum_{a=1}^k n_a = n$, and $\text{span}(Z_a) = S_a$. Under Proposition 4, S_a has dimension $r_a > 0$ and the subspaces form a direct sum. Choose a basis matrix $\Omega_a \in \mathbb{R}^{d_z \times r_a}$ and write $Z_a = \Omega_a \Xi_a$, where $\Xi_a \in \mathbb{R}^{r_a \times n_a}$ has full row rank. Define

$$\begin{aligned} \Omega &= [\Omega_1, \dots, \Omega_k] \in \mathbb{R}^{d_z \times r_z}, \\ \Xi &= \text{blkdiag}(\Xi_1, \dots, \Xi_k) \in \mathbb{R}^{r_z \times n}. \end{aligned} \quad (43)$$

The direct-sum condition makes Ω full column rank. Therefore, $Z = \Omega \Xi$ and $\Xi = \Omega^\dagger Z$, which imply $\text{row}(Z) = \text{row}(\Xi)$. Their orthogonal row-space projectors coincide:

$$\begin{aligned} Z^\dagger Z &= \Xi^\dagger \Xi \\ &= \text{blkdiag}(Z_1^\dagger Z_1, \dots, Z_k^\dagger Z_k). \end{aligned} \quad (44)$$

The second equality uses the block structure of Ξ and $\text{row}(\Xi_a) = \text{row}(Z_a)$. At a global fixed- Z optimum with $m = r_z$, Appendix B gives $C = Z^\dagger Z$. Equation (44) therefore proves subspace preservation.

For connectivity, the stated sample condition can be written

$$\text{rank}(Z_A) + \text{rank}(Z_B) > r_a \quad (45)$$

for every partition of a class's indices into two nonempty sets A and B . Here Z_A and Z_B contain the corresponding columns of Z_a . Indeed, their spans sum to S_a . Equality of their rank sum with r_a is equivalent to those spans forming a direct sum.

Set $C_a = Z_a^\dagger Z_a$. Suppose the graph defined by its nonzero off-diagonal entries is disconnected. There is a nontrivial partition A, B for which C_a is block diagonal. Since $\text{row}(C_a) = \text{row}(Z_a)$, this row space then splits into two spaces supported on the respective coordinate sets. Coordinate projection onto A has dimension $\text{rank}(Z_A)$, which also equals the rank of the corresponding diagonal block of C_a . The analogous equality holds for B . Adding the two block ranks gives $\text{rank}(Z_A) + \text{rank}(Z_B) = r_a$, contradicting (45). Conversely, equality in this rank sum gives a direct-sum partition. Applying (44) to its two groups makes C_a block diagonal. Thus, the sample condition is also necessary for connectivity of the full row-space projector graph. A singleton containing a nonzero sample has a connected one-vertex graph and satisfies the condition vacuously.

Every diagonal entry of C_a is positive when all samples are nonzero. A zero diagonal in an orthogonal projector forces

the corresponding row and column to vanish. The identity $Z_a = Z_a C_a$ would then make the corresponding sample zero, a contradiction. Positive row rescaling and entrywise squaring preserve the off-diagonal support. The affinity specified in the method therefore has exactly k connected components under these assumptions.

The eigenspace of eigenvalue one of its normalized affinity is spanned by the degree-weighted component indicators. Any orthonormal basis of this eigenspace differs from the normalized indicator basis by an orthogonal rotation. Row normalization maps every sample of one component to the same direction. Distinct components yield distinct directions. Consequently, the ideal embedding admits zero-distortion k -means assignments equal to the components, up to a label permutation.

APPENDIX D

PROPERTIES OF THE QUADRATIC AFFINITY AND SPECTRAL REDUCTION

a) Affinity and degrees: The definition of svec gives $\text{svec}(R)^\top \text{svec}(S) = \text{tr}(R^\top S)$ for symmetric $R, S \in \mathbb{R}^{m \times m}$. Applying this identity to the rank-one matrices $p_i p_i^\top$ and $p_j p_j^\top$ proves (26). Hence A is positive semidefinite. Its entrywise nonnegativity follows from (24). Let $e_i \in \mathbb{R}^n$ be the i th standard basis vector. Since $C^2 = C$,

$$\delta_i = e_i^\top C C^\top e_i = e_i^\top C e_i = \|p_i\|_2^2, \quad \sum_i \delta_i = \text{tr}(C) = m. \quad (46)$$

The diagonal entries of an orthogonal projector belong to $[0, 1]$, so $0 \leq \delta_i \leq 1$. Moreover, $\delta_i = 0$ if and only if $p_i = 0$, which makes the entire corresponding affinity row zero. Removing these rows, therefore, leaves the degrees of all active samples unchanged.

b) Proof of Proposition 5: Let $s = \text{rank}(F_+)$. Choose s rows of F_+ spanning its row space. The corresponding s rank-one matrices span all matrices $p_i p_i^\top$. In particular, their span contains

$$\sum_{i \in \mathcal{I}_+} p_i p_i^\top = P^\top P = I_m. \quad (47)$$

A linear combination of s rank-one matrices has rank at most s . Thus, $m \leq s$. The upper bound follows from the dimensions of F_+ . Since Δ is positive definite, B and F_+ have the same rank.

If $Gv = \nu v$ for $\nu > 0$, then $(BB^\top)(Bv) = B(Gv) = \nu Bv$ and $\|Bv\|_2^2 = \nu \|v\|_2^2$. Conversely, a positive eigenpair of $BB^\top$ maps to an eigenpair of G by multiplication by $B^\top$. For orthonormal eigenvectors,

$$\frac{v_a^\top B^\top B v_b}{\sqrt{\nu_a \nu_b}} = 1\{a = b\}. \quad (48)$$

This proves the correspondence, multiplicities, and orthonormality.

c) Leading vector and spectral convention: Let $\omega = \text{svec}(I_m)$. We have $F_+^\top 1_{n_+} = \omega$, $F_+ \omega = (\delta_i)_{i \in \mathcal{I}_+}$, and

$\|\omega\|_2^2 = m$. Therefore,

$$G\omega = F_+^\top \Delta^{-1} F_+ \omega = F_+^\top 1_{n_+} = \omega. \quad (49)$$

This proves (30). Let $A_{++} \in \mathbb{R}^{n_+ \times n_+}$ be the principal submatrix of A indexed by $\mathcal{I}_+$. The normalized affinity $BB^\top$ is similar to the row-stochastic matrix $\Delta^{-1} A_{++}$. Its eigenvalues have a magnitude of at most one. Positive semidefiniteness further restricts them to $[0, 1]$. Thus, the retained eigenvalue is dominant. Locking this vector remains valid when the eigenvalue one is repeated. Every active row then contains the strictly positive coordinate $\sqrt{\delta_i/m}$, which justifies the row normalization in (31).

Proposition 6 (Conditional recovery on the active graph). *Suppose A_{++} has exactly k connected components. The row-normalized embedding in (31), computed from exact eigenvectors, has one distinct row per component. An exact solution of k -means therefore recovers those components up to label permutation.*

Proof. Let $\Upsilon \in \{0, 1\}^{n_+ \times k}$ be the component membership matrix. The eigenvalue-one eigenspace of $BB^\top$ has dimension k and an orthonormal basis

$$\Delta^{1/2} \Upsilon (\Upsilon^\top \Delta \Upsilon)^{-1/2} O, \quad O^\top O = I_k. \quad (50)$$

The diagonal matrix $\Upsilon^\top \Delta \Upsilon$ contains the positive component volumes. Within component a , each row of this basis is a positive scalar multiple of the a th row of O . Row normalization removes that scalar. The k resulting rows are distinct because the rows of O are orthonormal. Clustering these rows into k nonempty groups has minimum within-cluster sum of squares zero, attained exactly by the component partition up to permutation. $\square$

d) Implicit reduced operator: Using (25) and the Frobenius inner-product identity, for each original index $i \in \mathcal{I}_+$, the corresponding row of $F_+ v$ equals $p_i^\top \text{smat}(v) p_i$. Multiplying by $F_+^\top \Delta^{-1}$ yields (33). This proves the implicit operator identity.

APPENDIX E

IMPLEMENTATION CONVENTIONS

The implementation specifies the network architecture, preprocessing, initialization, random seeds, Adam settings, and the iteration budgets in Algorithm 1. Each training update follows the stated order. When gradients are accumulated over sample chunks, their contributions in (19) are summed before the parameter update. Joint training starts with zero Adam moments.

a) Sample-based initialization: For k -means++ seeding of L , select the first index uniformly from $\{1, \dots, n\}$. Let $\mathcal{S}$ be the indices already selected. For each remaining index, compute its squared distance to the closest selected column,

$$d_i^{\text{seed}} = \min_{j \in \mathcal{S}} \|z_i - z_j\|_2^2, \quad \Pr(i \mid \mathcal{S}) = \frac{d_i^{\text{seed}}}{\sum_{j \notin \mathcal{S}} d_j^{\text{seed}}}. \quad (51)$$

Use this distribution until m distinct indices have been selected. When its denominator is zero, sample uniformly from the

remaining indices. Put the selected columns into L in selection order. This is the squared-distance seeding rule [31].

b) Adam update: For completeness, let w_t be the vector containing the current network parameters and let g_t be its gradient of the specified objective. These are vectorizations used only to define the optimizer. Let a_t and b_t be its first- and second-moment vectors. With $a_0 = b_0 = 0$, coefficient inputs $0 \leq \rho_1, \rho_2 < 1$, step size $\eta_t > 0$, and $\epsilon_{\text{Adam}} > 0$, one update is

$$\begin{aligned} a_t &= \rho_1 a_{t-1} + (1 - \rho_1) g_t, \\ b_t &= \rho_2 b_{t-1} + (1 - \rho_2) (g_t \odot g_t), \\ \hat{a}_t &= a_t / (1 - \rho_1^t), \quad \hat{b}_t = b_t / (1 - \rho_2^t), \\ w_{t+1} &= w_t - \eta_t \hat{a}_t \odot (\sqrt{\hat{b}_t} + \epsilon_{\text{Adam}}). \end{aligned} \quad (52)$$

Products $\odot$, divisions $\oslash$, and the square root are elementwise. These equations specify Adam [32], without an added weight-decay penalty. For pretraining, g_t differentiates (13); for joint training, it differentiates (17) with fixed factors. The moment vectors have the same dimension as w_t .

c) Factor accuracy: The thin decomposition in (21) uses m orthonormal columns, completing the singular bases when zero singular values occur. After each factor update, we check the residual $\|P^\top P - I_m\|_F$ at the chosen working precision.

d) Spectral accuracy and reproducibility: The eigensolver retains the analytical mode in (30) and computes the remaining dominant modes on its orthogonal complement. For prescribed residual, orthogonality, and positive-eigenvalue tolerances $\epsilon_{\text{eig}} > 0$, $\epsilon_{\text{orth}} > 0$, and $\epsilon_{\text{rank}} > 0$, respectively, we accept computed eigenpairs satisfying

$$\begin{aligned} \max_{1 \leq j \leq k} \|Gv_j - \nu_j v_j\|_2 &\leq \epsilon_{\text{eig}}, \\ \max_{1 \leq a, b \leq k} |v_a^\top v_b - \mathbf{1}\{a = b\}| &\leq \epsilon_{\text{orth}}, \\ \nu_k &> \epsilon_{\text{rank}}. \end{aligned} \quad (53)$$

The iteration budget or working precision is increased as needed to meet these criteria. Eigenvector ordering and random seeds are recorded for reproducibility. When an eigenvalue is repeated at the truncation boundary, the solver fixes a reproducible orthonormal basis of the selected modes. The active set contains every strictly positive degree in (27). Rows with zero degree are assigned by (32).

e) Final clustering: Let $\iota(i)$ denote the row of F_+ corresponding to the original sample index $i \in \mathcal{I}_+$. The same ordering is used in H and $\hat{H}$. With spectral centers $\mu_a \in \mathbb{R}^k$, the clustering objective and Lloyd updates are

$$\begin{aligned} \min_{\{\gamma_i\}, \{\mu_a\}} \sum_{i \in \mathcal{I}_+} \|\hat{H}_{\iota(i)}^\top - \mu_{\gamma_i}\|_2^2, \\ \gamma_i \leftarrow \arg \min_{1 \leq a \leq k} \|\hat{H}_{\iota(i)}^\top - \mu_a\|_2^2, \\ \mu_a \leftarrow \frac{1}{|\mathcal{J}_a|} \sum_{i \in \mathcal{J}_a} \hat{H}_{\iota(i)}^\top. \end{aligned} \quad (54)$$

Use $R_{\text{km}} \geq 1$ recorded k -means++ restarts on these rows. Apply the seeding rule (51) to the spectral rows, selecting k seeds among the n_+ active samples. Each restart takes at most T_{km} Lloyd steps. Ties between centers use the smallest

center index. Repair an empty cluster before its mean update by moving the sample with the largest current squared residual from a cluster containing at least two samples, with ties resolved by the original sample index. Then recompute affected centers. Stop early when assignments do not change. Keep the restart with minimum within-cluster sum of squares. This procedure returns k nonempty groups for $n_+ \geq k$, as ensured by (29).

APPENDIX F

COMPLEXITY COMPARISON WITH BASELINES

Table V separates representation learning from spectral computation. The bounds count sequential arithmetic operations. The numbers of training passes and optimization iterations are held fixed to isolate sample-size scaling. Eigensolver budgets are retained explicitly.

To compare dense, sparse, and implicit spectral operators on the same basis, define

$$C_{\text{sp}}(d_{\text{op}}, \omega) = T_{\text{eig}}(\omega + d_{\text{op}} s_{\text{eig}}) + R_{\text{eig}}(d_{\text{op}} s_{\text{eig}}^2 + s_{\text{eig}}^3). \quad (55)$$

Here, d_{op} is the operator dimension and ω bounds the cost of one matrix-vector product. The solver-specific budgets T_{eig} , s_{eig} , and R_{eig} have the meanings given in Section III-G. For an explicitly stored sparse graph, ω is its number of nonzero entries. A dense graph instead has $\omega = O(n^2)$. Thus, partial eigensolvers retain the benefit of sparse affinities. The common final k -means cost, $O(R_{\text{km}}(T_{\text{km}} + 1)nk^2)$, is added once to the sum of the two time columns for the complete pipeline.

PRO-DSC [28] trains on mini-batches and constructs its final affinity over the complete clustering set. Combining its regularizer costs with neural computation and matrix products gives the following per-batch computational-cost bound:

$$\begin{aligned} c_{\text{PRO}}(n_b) &= n_b c_{\text{net}} + n_b^2 d_z + \min\{n_b^3, d_z^3\} \\ &\quad + C_{\text{sp}}(n_b, n_b^2). \end{aligned} \quad (56)$$

For $1 \leq n_b \leq n$, the n/n_b multiplier counts batches in a full training pass. A fixed number of Sinkhorn iterations is absorbed into the matrix-product term. The separate $n^2 d_z$ term accounts for constructing the full-set affinity at inference. The per-sample network cost c_{net} is evaluated for each method's own architecture.

The regression-based methods differ in how they reduce optimization cost. S^5C [29] uses a selection budget of m and solves the final restricted regression for every sample. Its bound includes both selection and regression costs. For EnSC [19], the active-set solver is left explicit because the original algorithm permits different restricted solvers. A-DSSC [22] likewise permits different coefficient estimators. Each support-restricted transport evaluation is linear in e_{act} . A complete support check additionally costs $O(n^2)$ with stored coefficients, or $O(d_x n^2)$ when feature-space LSR entries are evaluated on demand. For LSR [22] with $d_x < n$, a feature-space solve reduces coefficient construction to $O(nd_x^2 + d_x^3 + n^2 d_x)$. Explicit assembly and storage of the full coefficient matrix remain quadratic in n .

The matrix-storage column isolates the representation responsible for affinity scaling. Total working memory also includes the input data, spectral embedding, eigensolver workspace, and regression-solver workspace. Deep methods additionally

TABLE V

TIME AND REPRESENTATION-STORAGE COMPLEXITY OF THE COMPARED METHODS. ALL ENTRIES ARE INSIDE $O(\cdot)$. THE TWO TIME COLUMNS ARE ADDITIVE, WITH THE COMMON FINAL k -MEANS COST ADDED ONCE. FIXED OPTIMIZATION BUDGETS ARE SUPPRESSED. SPECTRAL BUDGETS ARE INCLUDED THROUGH (55). THE LAST COLUMN COUNTS THE PRINCIPAL COEFFICIENT, FACTOR, AND AFFINITY MATRICES, NOT TOTAL WORKING MEMORY. BOUNDS ARE OBTAINED FROM THE ORIGINAL ALGORITHMS UNDER THE IMPLEMENTATION CONVENTIONS STATED BELOW.

Method	Deep	Representation / affinity learning	Spectral computation	Matrix storage
LMVSC [15]	No	$nd_xm + nm^3$	$nm^2 + m^3$	nm
SGL [16]	No	$nd_xm + nm^3$	$nm^2 + m^{3a}$	nm
S ⁵ C [29]	No	$n(d_xm + c_{\ell_1})$	$\mathcal{C}_{\text{sp}}(n, nm)$	nm
EnSC [19]	No	$n^2d_x + nc_{\text{EN}}$	$\mathcal{C}_{\text{sp}}(n, ns)$	ns
SSC-OMP [18]	No	n^2d_xs	$\mathcal{C}_{\text{sp}}(n, ns)$	ns
A-DSSC [22]	No	$\mathcal{C}_C + \mathcal{C}_{\text{chk}} + e_{\text{act}}$	$\mathcal{C}_{\text{sp}}(n, e_{\text{act}})$	$\mathcal{M}_C + e_{\text{act}}$
SSSC [20]	No	$R_{\text{drop}}\rho n^2d_xs$	$\mathcal{C}_{\text{sp}}(n, e_{\text{drop}})$	e_{drop}
LSR [9]	No	$n^2d_x + \min\{n^3, nd_x^2 + d_x^3\}^b$	$\mathcal{C}_{\text{sp}}(n, n^2)$	n^2
PRO-DSC [28]	Yes	$(n/n_b)c_{\text{PRO}}(n_b) + nc_{\text{net}} + n^2d_z$	$\mathcal{C}_{\text{sp}}(n, n^2)$	n^2
LoomSC	Yes	$nc_{\text{net}} + nd_zm + nm^2 + m^3$	$nm + \mathcal{C}_{\text{sp}}(q, nq) + nqk$	$nm + d_zm$

For LoomSC, $q = m(m+1)/2$. The table reports spectral computation using the operator in (33), without assembling G . s bounds the nonzeros per sample for EnSC and SSC-OMP, and per dropout subproblem for SSSC. For SSSC, ρ is the expected retained dictionary fraction and R_{drop} is the number of dropout dictionaries, giving $e_{\text{drop}} = \min\{n^2, nsR_{\text{drop}}\}$. The SSSC row gives the expected sequential computational cost at a fixed consensus budget. c_{ℓ_1} and c_{EN} are the costs of one restricted Lasso and elastic-net solve, respectively. The former uses at most m representatives; the latter depends on the active-set size. For A-DSSC, $\mathcal{C}_C$ and $\mathcal{M}_C$ denote the chosen coefficient solver's computational cost and matrix storage. $\mathcal{C}_{\text{chk}}$ includes support initialization and global checks, and e_{act} bounds the entries processed by its support-restricted transport solver. n_b is PRO-DSC's training batch size, and c_{PRO} is defined in (56). Anchor-method spectral costs use a direct thin SVD of the $n \times m$ factor. ^aSGL updates this representation within its graph-learning loop. ^bLSR uses the cheaper of the sample-space and feature-space linear solves. Both routes explicitly form the coefficient matrix.

retain latent features and network state. For LoomSC, these terms are specified in (37), with the corresponding eigensolver workspace. Quadratic feature rows can be generated as needed, so an $n \times q$ feature matrix need not be stored.

The comparison identifies the source of LoomSC's scalability advantage. LMVSC [15], SGL [16], and S⁵C [29] already have linear sample-size dependence for fixed representative and solver budgets. LoomSC retains this dependence while jointly learning nonlinear features and the two self-expression factors. EnSC [19] and SSC-OMP [18] can reduce coefficient storage through sparsity, but their dictionary scans still compare each target with the full dataset. SSSC reduces the dictionary size by dropout, giving the explicit $R_{\text{drop}}\rho n^2$ dependence in the table [20]. PRO-DSC [28] instead improves the training stage through mini-batches, while its full-affinity spectral inference retains quadratic storage. LoomSC avoids explicitly constructing either $n \times n$ matrix. Its q -dimensional reduction is exact for the defined squared affinity. For fixed m and solver budgets, both its learned factors and its reduced spectral computation scale linearly with n . Thus, LoomSC combines deep representation learning with scalability through the final clustering stage.