FAST LOCAL MEAN CURVATURE ESTIMATION FOR HIGH-DIMENSIONAL DATA MANIFOLDS

Anonymous authors
Paper under double-blind review

ABSTRACT

Local mean curvature is a powerful geometric descriptor of data manifolds. Nonetheless, its computational cost has largely prevented its adoption in modern machine learning(ML) methods. In the present work, it is introduced a scalable curvature estimation framework for high-dimensional data (MeCuCo). The new method combines an exact algebraic reformulation of a classical curvature estimator with a low-rank approximation that exploits the structure of local neighborhoods. This innovative combination yields a reduced computational complexity – from quartic to nearly linear in the ambient dimension for typical settings, while preserving the quality of the estimated curvature values. Theoretical analysis of the proposed formulation and approximation are provided, resulting in outstanding speed-up improvements between 50 and 300 times on real-world datasets. Importantly, through rendering curvature estimation computationally tractable at scale, MeCuCo facilitates the integration of curvature-aware representations and geometric inductive biases into manifold learning, representation learning, and related ML paradigms.

1 INTRODUCTION

Recent advances in ML have increasingly emphasized the importance of geometric structure in data representations. Methods such as graph neural networks (GNN), manifold learning algorithms, geometric deep learning (GDL) architectures, and representation learning models rely on the premise that useful information is encoded not only in the observed features but also in the geometry of the underlying data domain (Weber, 2025; Papillon et al., 2025). This perspective has led to a growing interest in learning algorithms that exploit intrinsic geometric properties of data manifolds rather than treating observations as isolated points in a high-dimensional Euclidean space.

The theoretical foundation of this paradigm is the *manifold hypothesis* (Fefferman et al., 2016), which states that high-dimensional datasets often concentrate near a smooth low-dimensional manifold embedded in the ambient feature space. This assumption underlies many successful ML techniques, including ISOMAP (Tenenbaum et al., 2000), locally linear embedding or LLE (Roweis & Saul, 2000), t-SNE (van der Maaten & Hinton, 2008), and UMAP (McInnes et al., 2018a;b). While these methods focus primarily on recovering low-dimensional representations that preserve neighborhood structure, a broader goal of geometric ML (GML) is to estimate differential-geometric quantities directly from data, enabling models to reason about the shape of the underlying manifold.

Among such quantities, local mean curvature is particularly attractive. This is due to the fact that it measures how the manifold bends in the ambient space, providing information that is complementary to both density- and distance-based statistics. Curvature can reveal transition regions between clusters, identify geometric boundaries, characterize local structural complexity, and provide informative signals for anomaly detection, active learning, graph learning, and representation learning (Asao & Ike, 2022; Cheng & Wu, 2021; Levada, 2025). Consequently, local curvature can potentially serve as a general-purpose geometric feature as well as a source of inductive bias for ML algorithms.

Despite its promise, local mean curvature remains largely absent from modern ML pipelines. The main obstacle is computational rather than conceptual. Existing estimators either assume low-dimensional ambient spaces or require operations whose computational complexity grows prohibitively with the number of features. In particular, the estimator recently introduced in Levada

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

(2025) provides accurate curvature estimates directly from local neighborhoods, although its computational cost scales as $O(m^4)$ per point. Thus, its adoption is impractical for many contemporary datasets, where the ambient dimension may range from hundreds to thousands.

In the present paper, we introduce the mean curvature computation (**MeCuCo**) method. This is a scalable framework for local mean curvature estimation in high-dimensional data manifolds. Our approach combines an exact algebraic reformulation of the original estimator with a low-rank approximation tailored to the neighborhood structure typically encountered in ML datasets. The first contribution is a closed-form identity that eliminates the explicit construction of the local shape operator while reducing the dominant computational cost from $O(m^4)$ to $O(m^3)$, without introducing any approximation error. The second contribution exploits the fact that local covariance matrices are inherently low-rank when the neighborhood size is much smaller than the ambient dimension. This replaces full eigendecompositions with a truncated singular value decomposition and an analytical treatment of the null-space contribution, yielding a practical complexity of $O(k^2m + kmp^2)$, where $p = k - 1$.

Beyond computational efficiency, our work aims to enable the adoption of curvature as a practical tool for ML. While reducing the cost of curvature estimation by several orders of magnitude, MeCuCo enables the integration of differential-geometric information into large-scale learning pipelines, including manifold learning, clustering, anomaly detection, active learning, graph learning, and representation learning. In this sense, the proposed method transforms local curvature estimation from a computational bottleneck into a scalable geometric primitive that can be routinely employed in modern ML systems.

Our contributions are summarized as follows: (1) we derive an exact algebraic reformulation of the local mean curvature estimator of Levada (2025), reducing its computational complexity from $O(m^4)$ to $O(m^3)$, while preserving exactness; (2) we develop a scalable low-rank approximation based on truncated singular value decomposition (SVD) and a principled analytical treatment of the covariance null space, yielding complexity $O(k^2m + kmp^2)$; (3) we provide theoretical analysis supporting both the exact reformulation and the proposed approximation; (4) we demonstrate experimentally that MeCuCo achieves speedups ranging from tens to hundreds of times on real-world datasets while preserving the geometric information captured by the original estimator; and (5) we establish local mean curvature as a practical geometric descriptor for geometry-aware ML and representation learning in high-dimensional settings.

The remainder of the paper is organized as follows. Section 2 revisits the original curvature estimator, and analyzes its computational limitations. Section 3 presents the proposed MeCuCo framework. Section 4 reports experimental results, and Section 5 presents the conclusions and final remarks.

2 Fast Mean Curvature Estimation

The mathematical backbone of the proposed method is drawn from classical differential geometry, the branch of mathematics concerned with the infinitesimal structure of smooth curves, surfaces, and manifolds (Spivak, 1999; Oprea, 2007; do Carmo, 2016; O'Neill, 2006; Tu, 2017; Needham, 2021). Let $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\} \subset \mathbb{R}^D$ be a finite dataset sampled from an unknown smooth d-dimensional manifold $\mathcal{M} \hookrightarrow \mathbb{R}^D$, with $d \ll D$. Our goal is to estimate, at each point $\mathbf{x}_i \in \mathcal{X}$, a discrete approximation of the shape operator $\mathcal{S}_i$ and, from it, the local mean curvature $\mathcal{K}_i$, using only the geometry of a small k-nearest-neighbor patch around $\mathbf{x}_i$. The procedure consists of five steps, described below.

Step 1: Local Neighborhood Construction. For each point $\mathbf{x}_i$, we identify its k-nearest neighbors under the Euclidean metric: $\mathcal{N}_i = \mathrm{kNN}(\mathbf{x}_i, k) = \{\mathbf{x}_{i_1}, \ldots, \mathbf{x}_{i_k}\}$. Under the manifold hypothesis, the set $\mathcal{N}_i$ provides a discrete approximation of a local coordinate chart around $\mathbf{x}_i$: for k sufficiently large relative to the local reach of $\mathcal{M}$, the convex hull of $\mathcal{N}_i$ is contained in a thin tubular neighborhood of $\mathcal{M}$, and the geodesic distances within $\mathcal{N}_i$ are well approximated by Euclidean distances (Fefferman et al., 2016; Tenenbaum et al., 2000). The parameter k governs a fundamental bias–variance trade-off: small k yields sharper local estimates but increases sensitivity to noise, while large k improves statistical stability at the cost of smoothing out fine geometric features.

Step 2: Local Covariance and Metric Approximation. Let $\bar{\mathbf{x}}_i = k^{-1} \sum_{\mathbf{x}_j \in \mathcal{N}_i} \mathbf{x}_j$ be the local centroid. We compute the sample covariance matrix of the neighborhood:

$$\mathbf{C}_i \;=\; \frac{1}{k-1} \sum_{\mathbf{x}_j \in \mathcal{N}_i} \big(\mathbf{x}_j - \bar{\mathbf{x}}_i\big)\big(\mathbf{x}_j - \bar{\mathbf{x}}_i\big)^\top \;\in\; \mathbb{R}^{D\times D}. \tag{1}$$

The matrix $\mathbf{C}_i$ captures the anisotropic dispersion of the data within $\mathcal{N}_i$, providing a first-order approximation of the local geometry of $\mathcal{M}$ near $\mathbf{x}_i$. Its leading eigenvectors span an estimate of the tangent space $T_{\mathbf{x}_i}\mathcal{M}$, while its eigenvalues reflect the local extent of the manifold along each direction (Singer & Wu, 2012). Following a Riemannian interpretation in which the metric tensor is adapted to the local data distribution, we associate the inverse covariance with the local metric $\mathbf{g}_i \;\approx\; \mathbf{C}_i^{-1}$. This corresponds to a Mahalanobis-type metric that contracts distances along high-variance directions while expanding them along low-variance ones, thereby de-emphasising ambient dimensions irrelevant to the local manifold structure (do Carmo, 2016). When $\mathbf{C}_i$ is rank-deficient, as is generically the case when $k \leq D$, the inverse is interpreted as a Moore-Penrose pseudoinverse restricted to the column space of $\mathbf{C}_i$.

Step 3: Second-Order Structure. Let $\mathbf{C}_i = \mathbf{W}_i \mathbf{\Lambda}_i \mathbf{W}_i^\top$ be the eigendecomposition of $\mathbf{C}_i$, where $\mathbf{\Lambda}_i = \operatorname{diag}(\lambda_1^{(i)}, \ldots, \lambda_D^{(i)})$ with $\lambda_1^{(i)} \geq \cdots \geq \lambda_D^{(i)} \geq 0$, and $\mathbf{W}_i = [\mathbf{w}_1^{(i)} \mid \cdots \mid \mathbf{w}_D^{(i)}]$ is the orthogonal matrix of corresponding eigenvectors. The leading d columns of $\mathbf{W}_i$, those associated with the d largest eigenvalues, span the estimated tangent space $\widehat{T}_{\mathbf{x}_i}\mathcal{M}$, while the remaining $D-d$ columns span the estimated normal space $\widehat{N}_{\mathbf{x}_i}\mathcal{M}$. To capture second-order geometric information, that is, the local curvature of $\mathcal{M}$, we construct a feature matrix $\mathbf{H}_i \in \mathbb{R}^{D\times p}$, with $p = D + D(D-1)/2$, whose columns are:

- the *quadratic terms* $\big(\mathbf{w}_j^{(i)}\big)^{\circ 2}$, for $j = 1, \ldots, D$, encoding the self-interaction of each eigenvector direction; and
- the *cross terms* $\mathbf{w}_j^{(i)} \circ \mathbf{w}_\ell^{(i)}$, for $1 \leq j < \ell \leq D$, encoding the interaction between pairs of eigenvector directions,

where $\circ$ denotes the Hadamard (element-wise) product. The matrix $\mathbf{H}_i$ constitutes a local quadratic expansion of the eigenvector frame, and its columns span a feature space that is sensitive to deviations of $\mathcal{M}$ from its tangent hyperplane. The second fundamental form is approximated by the Gram matrix of this feature space: $\widehat{\mathit{II}}_i = \mathbf{H}_i \mathbf{H}_i^\top \in \mathbb{R}^{D\times D}$. This construction provides a data-driven estimate of the local curvature tensor by measuring, in a least-squares sense, the degree to which the neighborhood $\mathcal{N}_i$ departs from the linear (flat) approximation provided by $\widehat{T}_{\mathbf{x}_i}\mathcal{M}$ (Levada, 2025; Cheng & Wu, 2021).

Step 4: Shape Operator Estimation. The shape operator is the linear map $\mathcal{S}_i : T_{\mathbf{x}_i}\mathcal{M} \to T_{\mathbf{x}_i}\mathcal{M}$ given by $\mathcal{S} = G^{-1}L$ in local coordinates, that is, the composition of the curvature matrix with the inverse metric. Substituting the approximations, we obtain the discrete estimator: $\widehat{\mathcal{S}}_i = -\widehat{\mathit{II}}_i \mathbf{g}_i^{-1} = -\mathbf{H}_i \mathbf{H}_i^\top \mathbf{C}_i$, where the negative sign follows a convention, ensuring that positive eigenvalues correspond to convex bending toward the normal. The estimator is computable from the eigendecomposition of $\mathbf{C}_i$, not requiring explicit knowledge of the embedding map or the unit normal field.

Step 5: Mean Curvature Estimation. The shape operator $\widehat{\mathcal{S}}_i$ encodes the full local curvature structure of $\mathcal{M}$ near $\mathbf{x}_i$: its eigenvalues $\widehat{\kappa}_1^{(i)} \geq \cdots \geq \widehat{\kappa}_d^{(i)}$ approximate the principal curvatures, and its eigenvectors approximate the principal curvature directions. The trace of $\widehat{\mathcal{S}}_i$, which is invariant under change of basis and equals the sum of principal curvatures, yields the mean curvature estimator: $\mathcal{K}_i = \big|\operatorname{tr}\big(\widehat{\mathcal{S}}_i\big)\big| = \big|\operatorname{tr}\big(-\mathbf{H}_i \mathbf{H}_i^\top \mathbf{C}_i\big)\big|$.

The absolute value accommodates the sign ambiguity of the unit normal in high co-dimension, ensuring that $\mathcal{K}_i \geq 0$ regardless of the orientation of $\widehat{N}_{\mathbf{x}_i}\mathcal{M}$. This scalar quantity provides a coordinate-free measure of local geometric complexity. High values of $\mathcal{K}_i$ identify regions where the manifold bends sharply and local linear approximations are least reliable, while low values correspond to near-flat regions where the tangent space provides an accurate description of the local structure.

As established in Theorem 1 (see Appendix for details), the naive method, which requires forming the $D \times p$ matrix $\mathbf{H}_i$ with $p = O(D^2)$ columns and computing the product $\mathbf{H}_i\mathbf{H}_i^\top$ at cost $O(D^4)$, can be replaced by the closed-form expression

$$\mathcal{K}_i \;=\; \left| \frac{1}{2}\,\boldsymbol{\lambda}_i^\top \left(C_i^{\odot 2}\,\mathbf{1}_D\right) + \frac{1}{2}\,\mathbf{1}_D^\top \boldsymbol{\lambda}_i \right|, \tag{2}$$

at a total cost of $O(D^2)$ after the eigendecomposition, where $\mathbf{W}_i^{(2)}$ denotes the element-wise square of $\mathbf{W}_i$, $C_i^{\odot 2}$ the element-wise square of C_i, and $\boldsymbol{\lambda}_i = (\lambda_1^{(i)}, \ldots, \lambda_D^{(i)})^\top$ the eigenvalues of $\mathbf{C}_i$.

Algorithm 1 summarizes the proposed MeCuCo framework. The method estimates a scalar mean curvature score at each sample point using the closed-form identity of Theorem 1, replacing the $O(D^4)$ computation of the original formulation with an $O(D^2)$ matrix operation after the local eigendecomposition. The algorithm operates in two regimes, selected automatically by the `mode` parameter: EXACT mode applies the full eigendecomposition of the local covariance matrix via a LAPACK divide-and-conquer solver (driver `evr`/`evd`), yielding exact curvature estimates at cost $O(D^3)$ per point; FAST mode replaces the eigendecomposition with a truncated SVD of the $k \times D$ centered neighborhood matrix, exploiting the rank-$(k{-}1)$ structure of the local covariance and an analytical approximation of the null-space contribution, reducing the per-point cost to $O(k^2D + kDp^2)$ with $p = k - 1$. In both modes, the algorithm is fully data-driven and requires neither an explicit manifold parametrisation nor prior knowledge of the intrinsic dimension d, making it directly applicable to high-dimensional datasets. In AUTO mode (default), the algorithm selects EXACT when $D < 50$ and FAST otherwise.

Algorithm 1 Fast Mean Curvature Computation (MeCuCo)

function MECUCO(X, k, `mode`)
  Input: $X \in \mathbb{R}^{n \times D}$ (dataset), k (number of neighbors), `mode` $\in \{$AUTO, EXACT, FAST$\}$
  Output: $\mathcal{K} \in \mathbb{R}^n_{\geq 0}$ (mean curvature scores)
  `mode` $\leftarrow$ EXACT if $D < 50$, else FAST ▷ Auto-select regime
  $A \leftarrow$ Nearest_Neighbors(X, k) ▷ Find the k-nearest-neighbors for each sample
  for $i = 1$ **to** n **do**
    $\mathcal{N}_i \leftarrow$ neighbors of $\mathbf{x}_i$ in A ▷ Step 1: local neighborhood
    if `mode` $=$ EXACT **then**
      $\mathbf{C}_i \leftarrow \frac{1}{k-1} \sum_{\mathbf{x}_j \in \mathcal{N}_i} (\mathbf{x}_j - \bar{\mathbf{x}}_i)(\mathbf{x}_j - \bar{\mathbf{x}}_i)^\top$ ▷ Step 2: local covariance, $O(kD^2)$
      $\boldsymbol{\lambda}_i,\ \mathbf{W}_i \leftarrow \mathrm{eigh}(\mathbf{C}_i,\ \texttt{driver=evr/evd})$ ▷ Step 3: eigendecomposition, $O(D^3)$
      $C_i \leftarrow \mathbf{W}_i^\top\, \mathbf{W}_i^{\odot 2}$ ▷ $C_i[s,\ell] = \sum_a W_{as} W_{a\ell}^2$, $O(D^2)$
      $\mathcal{K}_i \leftarrow \left| \frac{1}{2}\,\boldsymbol{\lambda}_i^\top (C_i^{\odot 2}\,\mathbf{1}_D) + \frac{1}{2}\,\mathbf{1}_D^\top \boldsymbol{\lambda}_i \right|$ ▷ Theorem 1, $O(D^2)$
    else if `mode` $=$ FAST **then**
      $\bar{\mathbf{x}}_i \leftarrow \frac{1}{k} \sum_{\mathbf{x}_j \in \mathcal{N}_i} \mathbf{x}_j$; $X_c \leftarrow [\mathbf{x}_j - \bar{\mathbf{x}}_i]_{j=1}^k \in \mathbb{R}^{k \times D}$
      $\mathbf{s},\ V^\top \leftarrow \mathrm{SVD}_p(X_c)$, $p = k - 1$ ▷ Truncated SVD, $O(k^2D)$
      $V_r \leftarrow V_{[1:p,\,:]}^{\top\ \top} \in \mathbb{R}^{D \times p}$; $\boldsymbol{\lambda}_r \leftarrow \mathbf{s}_{[1:p]}^{\odot 2}/(k{-}1)$ ▷ Non-zero eigenvalues
      $\tau \leftarrow \boldsymbol{\lambda}_r^\top \mathbf{1}_p$ ▷ $\mathrm{tr}(\mathbf{C}_i) = \sum_{l=1}^p \lambda_l$
      $t_{\mathrm{range}} \leftarrow \left\| X_c\, V_r^{\odot 2} \right\|_F^2 / (k{-}1)$ ▷ Range contribution, $O(kDp)$
      $\mathbf{d}_P \leftarrow \mathbf{1}_D - (V_r^{\odot 2})\,\mathbf{1}_p$ ▷ Diagonal of $P_\perp = I - V_r V_r^\top$
      $t_A \leftarrow \left\| X_c\, \mathbf{d}_P \right\|^2 / (k{-}1)$ ▷ Null-space term A, $O(kD)$
      $t_{B_2} \leftarrow \left\| X_c\, V_r \right\|_F^2 / (k{-}1)$
      $G_{\mathrm{tens}} \leftarrow \mathrm{einsum}(X_c, V_r, V_r) \in \mathbb{R}^{k \times p \times p}$ ▷ $O(kDp^2)$
      $t_B \leftarrow \tau - 2\,t_{B_2} + \left\| G_{\mathrm{tens}} \right\|_F^2 / (k{-}1)$
      $t_{\mathrm{null}} \leftarrow (t_A + 2\,t_B)\,/\,(D - p)$ ▷ Null-space contribution
      $\mathcal{K}_i \leftarrow \left| \frac{1}{2}\,(t_{\mathrm{range}} + t_{\mathrm{null}}) + \frac{1}{2}\,\tau \right|$ ▷ Fast curvature estimate
    end if
  end for
  return $\mathcal{K}$
end function

2.1 EXACT MODE

The EXACT mode (lines 8–12 of Algorithm 1) evaluates the mean curvature estimator without approximation, using the closed-form identity of Theorem 1. Its four steps are as follows.

Local covariance (line 9). For each point $\mathbf{x}_i$, the centered neighborhood matrix $X_c \in \mathbb{R}^{k\times D}$, whose rows are the vectors $\mathbf{x}_j - \bar{\mathbf{x}}_i$ for $\mathbf{x}_j \in \mathcal{N}_i$, is used to form the sample covariance $\mathbf{C}_i = X_c^\top X_c/(k-1)$ at cost $O(kD^2)$.

Full eigendecomposition (line 10). The symmetric eigendecomposition $\mathbf{C}_i = \mathbf{W}_i \operatorname{diag}(\boldsymbol{\lambda}_i)\mathbf{W}_i^\top$ is computed via `scipy.linalg.eigh` with a LAPACK driver selected adaptively by dimension: `evr` (MRRR algorithm) for $D < 500$, and `evd` (divide-and-conquer) for $D \geq 500$, both of which are substantially faster than the default `ev` (QR iteration) for large matrices (Anderson et al., 1999). This step costs $O(D^3)$ while producing the orthogonal eigenvector matrix $\mathbf{W}_i$ and the eigenvalue vector $\boldsymbol{\lambda}_i \in \mathbb{R}^D$.

Construction of C_i (line 11). The matrix $C_i = \mathbf{W}_i^\top \mathbf{W}_i^{\odot 2} \in \mathbb{R}^{D\times D}$, where $\mathbf{W}_i^{\odot 2}$ denotes the element-wise square of $\mathbf{W}_i$, is computed by a single dense matrix multiplication at cost $O(D^3)$, although with a constant factor far smaller compared to the one forming $\mathbf{H}_i\mathbf{H}_i^\top$, since both operands are $D \times D$ rather than $D \times O(D^2)$. Recalling the definition $C_i[s,\ell] = \sum_{a=1}^{D} W_{as} W_{a\ell}^2$, each entry of C_i measures the inner product between the s-th eigenvector and the element-wise square of the ℓ-th eigenvector. This quantity encodes the geometric interaction between pairs of local curvature directions.

Curvature via the closed-form identity (line 12). Lastly, the mean curvature is evaluated as equation 2, which requires only element-wise squaring of C_i, row summation, and a dot product with $\boldsymbol{\lambda}_i$, all at cost $O(D^2)$. According Theorem 1, equation 2 is algebraically identical to the original $O(D^4)$ formulation; the only source of numerical discrepancy is floating-point rounding, which in practice produces relative errors of order 10^{-12}–10^{-14} (Remark 1). The total per-point cost of EXACT mode is dominated by the eigendecomposition and the matrix product C_i, both $O(D^3)$, resukting in an overall complexity of $\boldsymbol{O(kD^2 + D^3)}$ per sample.

2.2 FAST MODE: MOTIVATION AND CONSTRUCTION

For datasets with $D \gg k$, the EXACT mode still incurs an $O(D^3)$ eigendecomposition of the $D \times D$ covariance matrix, becoming the dominant bottleneck when D is in the hundreds or thousands. The FAST mode (lines 13–21 of Algorithm 1) addresses this bottleneck by exploiting a fundamental structural property of $\mathbf{C}_i$ that is present whenever $D > k$: the local covariance matrix is *rank-deficient*.

Low-rank structure of the local covariance. Since $\mathbf{C}_i = X_c^\top X_c/(k-1)$ is the outer product of the $k \times D$ matrix X_c with itself, its rank is at most $\min(k-1, D) = k-1 \ll D$. This means that $\mathbf{C}_i$ has at most $p = k-1$ non-zero eigenvalues, and the remaining $D-p$ eigenvalues are identically zero. Consequently, the full $D \times D$ eigendecomposition performed in EXACT mode computes $D-p$ eigenvectors, whose associated eigenvalues contribute nothing to $\boldsymbol{\lambda}_i^\top \boldsymbol{\lambda}_i$. However, their computation represents the bulk of the $O(D^3)$ cost. The key question is therefore: can the mean curvature estimator equation 2 be computed from the p non-zero eigenvectors alone, without forming the $D \times D$ matrix $\mathbf{C}_i$ or its full eigendecomposition?

Truncated SVD as a substitute for eigh. The non-zero eigenvectors of $\mathbf{C}_i$ are precisely the right singular vectors of X_c. The compact singular value decomposition $X_c = U \operatorname{diag}(\mathbf{s}) V^\top$, with $U \in \mathbb{R}^{k\times k}$, $\mathbf{s} \in \mathbb{R}^k$, $V \in \mathbb{R}^{D\times k}$, satisfies

$$\mathbf{C}_i \;=\; \frac{X_c^\top X_c}{k-1} \;=\; V \operatorname{diag}\!\left(\frac{\mathbf{s}^{\odot 2}}{k-1}\right) V^\top, \tag{3}$$

thus, the non-zero eigenvalues of $\mathbf{C}_i$ are $\lambda_\ell = s_\ell^2/(k-1)$ and the corresponding eigenvectors are the columns of V (line 14–15). Computing the compact SVD of $X_c \in \mathbb{R}^{k\times D}$ costs $O(k^2 D)$, a factor of $(D/k)^2$ cheaper than the full eigendecomposition of $\mathbf{C}_i$, and for $k \ll D$ this represents a speedup of several orders of magnitude (e.g., $\approx 1000\times$ for $D = 1000$, $k = 8$).

The null-space contribution. Substituting the low-rank decomposition equation 3 into equation 2 and splitting the sum over eigenvalues into the p non-zero and $D-p$ zero components, one finds that the term $\boldsymbol{\lambda}_i^\top (C_i^{\odot 2}\,\mathbf{1}_D)$ decomposes as

$$\boldsymbol{\lambda}_i^\top \left(C_i^{\odot 2}\,\mathbf{1}_D\right) \;=\; \underbrace{\boldsymbol{\lambda}_r^\top \left(C_r^{\odot 2}\,\mathbf{1}_D\right)}_{t_{\text{range}}} + \underbrace{\sum_{s\in\mathcal{I}_0} \lambda_s \sum_{\ell=1}^{D} C_i[s,\ell]^2}_{=0,\ \text{since } \lambda_s=0} + \underbrace{\boldsymbol{\lambda}_r^\top \left(C_{r0}^{\odot 2}\,\mathbf{1}_{D-p}\right)}_{t_{\text{null}}}, \tag{4}$$

where $\boldsymbol{\lambda}_r \in \mathbb{R}^p$ collects the non-zero eigenvalues, $C_r[s,\ell] = \sum_a V_{as} V_{a\ell}^2$ for s,ℓ in the range space, $C_{r0}[s,\ell] = \sum_a V_{as}\, V_{0,a\ell}^2$ for ℓ in the null space, and $\mathcal{I}_0$ indexes the zero eigenvalues. The middle term vanishes because $\lambda_s = 0$ for $s \in \mathcal{I}_0$, which is the crucial cancellation that makes the truncated approach viable. The term t_{range} involves only V_r (the $D\times p$ matrix of non-zero eigenvectors) and is computable at cost $O(kDp)$ without forming $\mathbf{C}_i$; the term t_{null}, however, involves the $D \times (D-p)$ matrix of null-space eigenvectors V_0, which are *not* computed by the truncated SVD, and would require $O(D^3)$ to be obtained explicitly.

Analytical approximation of the null-space term. The central theoretical contribution of FAST mode is the observation that, while the null-space eigenvectors of $\mathbf{C}_i$ are not unique, any orthonormal basis of the $(D-p)$-dimensional null space is equally valid, their *expected contribution* to t_{null} under the uniform (Haar) distribution over orthonormal null-space bases admits a closed-form expression.

Specifically, let $V_0 \in \mathbb{R}^{D\times(D-p)}$ be drawn uniformly from the Stiefel manifold of orthonormal frames in the null space of $V_r^\top$. For a fixed V_r, the expected outer product of two null-space columns satisfies (Meckes, 2019):

$$\mathbb{E}_{V_0}[G_{\text{null}}[i,j]] \;=\; \mathbb{E}_{V_0}\left[\sum_{\ell=1}^{D-p} V_{0,i\ell}^2\, V_{0,j\ell}^2\right] \;=\; \frac{P_{\perp,ii}\,P_{\perp,jj} + 2\,P_{\perp,ij}^2}{D-p+2}, \tag{5}$$

where $P_\perp = I_D - V_r V_r^\top$ is the orthogonal projector onto the null space. In summary, this formula is derived by the second order moments of uniform vectors in the sphere $\mathbb{S}^{D-p-1}$, with a orthonormality correction imposed by the Haar measure over the Stiefel manifold $\mathrm{St}(D-p,\,D)$ (Meckes, 2019). Substituting this expectation into t_{null} and expanding, the null-space contribution becomes

$$t_{\text{null}} \;\approx\; \frac{t_A + 2\,t_B}{D-p}, \tag{6}$$

where

$$t_A \;=\; \frac{\|X_c\,\mathbf{d}_P\|^2}{k-1}, \qquad \mathbf{d}_P = \mathrm{diag}(P_\perp) = \mathbf{1}_D - (V_r^{\odot 2})\mathbf{1}_p, \tag{7}$$

$$t_B \;=\; \tau - 2\,\frac{\|X_c V_r\|_F^2}{k-1} + \frac{\|G_{\text{tens}}\|_F^2}{k-1}, \tag{8}$$

with $\tau = \boldsymbol{\lambda}_r^\top \mathbf{1}_p = \mathrm{tr}(\mathbf{C}_i)$ and $G_{\text{tens}}[a,l_1,l_2] = \sum_i X_c[a,i]\,V_r[i,l_1]\,V_r[i,l_2]$ the order-3 tensor encoding all pairwise eigenvector interactions within X_c. Every quantity in equation 7–equation 8 is expressible as a norm or inner product involving only X_c and V_r, with no reference to V_0. The most expensive operation is the computation of $G_{\text{tens}} \in \mathbb{R}^{k\times p\times p}$, which costs $O(kDp^2)$; for fixed k and $p = k-1$, this is $O(kDk^2) = O(k^3 D)$, dominated by the truncated SVD cost $O(k^2 D)$ when k is moderate. The final FAST mode curvature estimate is

$$\mathcal{K}_i \;=\; \left|\frac{1}{2}\,(t_{\text{range}} + t_{\text{null}}) + \frac{1}{2}\,\tau\right|, \tag{9}$$

at a total per-point cost of $\boldsymbol{O(k^2 D + kDp^2)}$, compared to $O(D^3)$ for EXACT mode, a reduction of $(D/k)^2$ in the dominant term.

Summary of computational complexity. The overall per-point complexities are:

- EXACT: $O(kD^2 + D^3)$, dominated by `eigh`;
- FAST: $O(k^2 D + kDp^2)$ for fixed k, dominated by the truncated SVD and G_{tens};
- Original MCBP (Levada, 2025): $O(kD^2 + D^4)$, dominated by $\mathbf{H}_i\mathbf{H}_i^\top$.

For the regime $k \ll D$ that characterises high-dimensional ML applications, FAST mode achieves a reduction of $(D/k)^2$ over EXACT mode and $(D/k)^3$ over the original algorithm, improvements of three and four orders of magnitude.

3 COMPUTATIONAL EXPERIMENTS AND RESULTS

In the present work, we address three fundamental questions: (i) Does MeCuCo preserve the geometric information captured by the original estimator?; (ii) how does its computational cost scale with dimensionality?; and (iii) does scalable curvature estimation become practical for datasets where the original method is infeasible?

Experiments were conducted on a diverse collection of more than 40 publicly available datasets from OpenML (Vanschoren et al., 2013; Bischl et al., 2021). The benchmark data spans a broad range of sample sizes, ambient dimensions, and geometric structures, including near-linear manifolds, highly curved manifolds, and heterogeneous-density datasets. This diversity enables the objective evaluation of MeCuCo across the wide range of conditions encountered in modern ML applications. All datasets, preprocessing scripts, and experimental results are publicly available to support reproducibility.

The first group of experiments evaluates the agreement between the curvature scores produced by MeCuCo and those obtained from the original formulation. We report the mean absolute error (MAE) together with two rank-based measures, the Spearman correlation coefficient ρ_S, and the Chatterjee coefficient ξ (Chatterjee, 2021). While MAE quantifies absolute differences between curvature values, rank-based metrics assess whether the relative ordering of points is preserved.

In practice, curvature scores are often used to rank or weight samples rather than as absolute geometric quantities. Because their scale depends on dataset-specific factors, direct comparisons across datasets are generally not meaningful. We therefore consider a normalized version of the curvature scores in the interval $[0, 1]$, which preserves sample ordering while providing a standardized representation suitable for downstream ML tasks. We compare the original and proposed estimators using this normalized formulation.

Table 1 reports the fidelity results after applying min-max normalization to the curvature scores of each dataset. The results show that MeCuCo preserves the curvature structure of the original estimator with very high fidelity. Across the benchmark corpus, the median Spearman correlation reaches $\rho_S = 0.9998$ and the median Chatterjee coefficient reaches $\xi = 0.9832$, indicating near-perfect agreement in the ranking of samples according to their curvature. The normalized MAE is also extremely small (median below 10^{-3}), showing that the pointwise curvature values produced by MeCuCo closely match those of the original formulation. These results are particularly relevant because most downstream applications, such as boundary detection, anomaly scoring, clustering, and active learning, primarily depend on the relative ordering of curvature scores rather than on their absolute magnitude.

Importantly, the computational advantages of MeCuCo remain unchanged. Across the experiments on benchmark data, the proposed method consistently achieves one to three orders of magnitude speedup relative to the original estimator, with the largest gains occurring precisely in the high-dimensional regime where curvature estimation is most computationally challenging. These results demonstrate that MeCuCo preserves the geometric information captured by the original estimator, while enabling an efficient curvature computation at scales that was previously inaccessible.

Table 2 evaluates MeCuCo on 25 datasets with ambient dimensionalities ranging from $m = 400$ to $m = 54{,}621$. These datasets lie well beyond the practical range of the original MCBP estimator, whose $O(m^4)$ complexity makes curvature computation infeasible in high dimensions. In contrast, MeCuCo successfully processes all datasets, with running times ranging from 0.11 seconds to 78.37 seconds.

The results confirm that the computational cost of MeCuCo depends primarily on the number of samples rather than on the ambient dimensionality. Datasets containing tens of thousands of features are processed in only a few seconds, while the largest execution times are observed for datasets with $70{,}000$ samples. In addition, the estimated curvature distributions reveal substantial geometric variability across domains, indicating that local mean curvature remains an informative descriptor

Table 1: Average local curvatures (normalized to $[0, 1]$) and standard deviations obtained by the original method and the proposed computationally efficient variation (MeCuCo) for 40 OpenML datasets.

	Original		MeCuCo		Metrics		
Datasets	**Average**	**Time (s)**	**Average**	**Time (s)**	**MAE**	ρ_S	ξ
iris	0.1588 ± 0.1571	0.02	0.1588 ± 0.1571	0.03	0.0000	1.0000	0.9801
seeds	0.1645 ± 0.1522	0.01	0.1645 ± 0.1522	0.02	0.0000	1.0000	0.9857
thoracic_surgery	0.0926 ± 0.1502	0.07	0.0920 ± 0.1490	0.06	0.0008	0.9999	0.9915
page-blocks	0.0025 ± 0.0301	0.62	0.0025 ± 0.0301	0.53	0.0000	0.9999	0.9992
segment	0.0088 ± 0.0535	0.61	0.0088 ± 0.0535	0.26	0.0000	0.9999	0.9942
hill-valley	0.0155 ± 0.0850	7.91	0.0154 ± 0.0844	0.37	0.0001	0.9999	0.9964
cardiotocography	0.0352 ± 0.0599	1.35	0.0353 ± 0.0597	0.56	0.0003	0.9997	0.9811
collins	0.1397 ± 0.1097	0.25	0.1377 ± 0.1097	0.09	0.0029	0.9990	0.9610
artificial-characters	0.0685 ± 0.0782	0.87	0.0685 ± 0.0783	0.58	0.0001	0.9999	0.9977
GesturePhaseSegmentation	0.0289 ± 0.0602	5.61	0.0291 ± 0.0607	1.81	0.0003	0.9999	0.9977
letter	0.1836 ± 0.1146	3.32	0.1835 ± 0.1146	1.42	0.0007	0.9999	0.9920
JapaneseVowels	0.2114 ± 0.1132	1.85	0.2116 ± 0.1132	1.14	0.0007	0.9999	0.9911
gas-drift	0.0009 ± 0.0104	232.14	0.0009 ± 0.0105	8.38	0.0000	0.9999	0.9899
USPS	0.1393 ± 0.1212	1396.68	0.1383 ± 0.1211	9.72	0.0017	0.9998	0.9848
qsar-biodeg	0.0735 ± 0.1277	0.96	0.0742 ± 0.1289	0.29	0.0009	0.9998	0.9844
smartphone	0.2587 ± 0.2493	0.37	0.2566 ± 0.2463	0.04	0.0031	0.9998	0.9739
ionosphere	0.1176 ± 0.1493	0.20	0.1176 ± 0.1497	0.09	0.0010	0.9998	0.9831
satimage	0.0875 ± 0.1000	4.30	0.0879 ± 0.1005	1.44	0.0007	0.9999	0.9879
steel-plates-fault	0.1428 ± 0.1168	1.07	0.1414 ± 0.1155	0.45	0.0022	0.9997	0.9787
depression_2020	0.3323 ± 0.2345	0.44	0.3283 ± 0.2320	0.13	0.0048	0.9997	0.9838
one-hundred-plants-shape	0.0466 ± 0.1019	3.30	0.0471 ± 0.1036	0.45	0.0008	0.9996	0.9789
eye_movements	0.0716 ± 0.0767	4.82	0.0723 ± 0.0776	1.82	0.0011	0.9996	0.9773
Satellite	0.0378 ± 0.0503	3.38	0.0375 ± 0.0498	1.16	0.0004	0.9998	0.9851
texture	0.0315 ± 0.0421	4.44	0.0319 ± 0.0424	1.29	0.0005	0.9998	0.9833
vowel	0.2413 ± 0.1309	0.15	0.2437 ± 0.1325	0.13	0.0028	0.9997	0.9836
mfeat-factors	0.1935 ± 0.1261	156.58	0.1927 ± 0.1255	1.55	0.0010	0.9999	0.9896
breast_cancer	0.0919 ± 0.1086	0.31	0.0901 ± 0.1063	0.14	0.0020	0.9997	0.9781
arrhythmia	0.0873 ± 0.1022	71.48	0.0888 ± 0.1030	0.24	0.0019	0.9996	0.9743
pendigits	0.0612 ± 0.0528	1.90	0.0591 ± 0.0510	0.85	0.0021	0.9999	0.9917
one-hundred-plants-texture	0.2085 ± 0.1411	3.31	0.2052 ± 0.1398	0.42	0.0041	0.9993	0.9703
optdigits	0.0186 ± 0.0471	11.40	0.0177 ± 0.0469	1.41	0.0008	0.9996	0.9772
digits	0.0447 ± 0.0627	3.75	0.0427 ± 0.0622	0.46	0.0020	0.9995	0.9745
sylvine	0.2818 ± 0.1295	1.39	0.2806 ± 0.1285	0.40	0.0035	0.9993	0.9665
solar-flare	0.0097 ± 0.0468	0.12	0.0097 ± 0.0469	0.10	0.0000	0.9999	0.9969
TuningSVMs	0.1582 ± 0.1501	0.50	0.1592 ± 0.1475	0.03	0.0032	0.9993	0.9592
wine	0.1640 ± 0.1483	0.02	0.1579 ± 0.1431	0.02	0.0000	0.9970	0.9272
splice	0.5086 ± 0.1227	5.77	0.5049 ± 0.1209	0.93	0.0045	0.9994	0.9721
Indian_pines	0.2468 ± 0.0859	905.01	0.2458 ± 0.0868	10.11	0.0027	0.9989	0.9594
mfeat-pixel	0.2467 ± 0.1077	208.73	0.2411 ± 0.1059	1.23	0.0056	0.9988	0.9601

even in very high-dimensional spaces. These results further substantiate the capacity of MeCuCo to render curvature-based geometric analysis computationally tractable at scales that were previously unattainable.

Beyond scalability, the estimated curvature distributions reveal meaningful geometric variation across domains. Some datasets exhibit nearly flat local geometry, as reflected by low average curvature values, while others present substantially higher curvature and broader distributions, indicating stronger local nonlinearity and geometric heterogeneity. These results suggest that local mean curvature remains an informative geometric descriptor, even in very high-dimensional spaces.

4 Conclusion

We present MeCuCo, a scalable framework for estimating local mean curvature on high-dimensional data manifolds. The method integrates an exact algebraic reformulation of an existing curvature

Table 2: Number of samples, features, average normalized local mean curvatures, quantiles and the elapsed time for high-dimensional datasets (the original method is computationally unfeasible).

Datasets	**n**	**m**	**Average K**	**25%**	**75%**	**Time (s)**
Speech	3,686	400	0.4034 ± 0.1394	0.3082	0.4893	2.55
madelon	2,600	500	0.4982 ± 0.1450	0.3993	0.5954	1.72
har	10,299	561	0.0362 ± 0.0435	0,0110	0.0462	9.19
isolet	7,797	617	0.1662 ± 0.0849	0.1149	0.1997	7.35
parkinson-speech-uci	756	753	0.0655 ± 0.0908	0.0232	0.0759	0.56
MNIST_784	70,000	784	0.0130 ± 0.0287	0.0057	0.0136	75.89
Fashion-MNIST	70,000	784	0.0110 ± 0.0176	0.0046	0.0116	77.66
Kuzushiji-MNIST	70,000	784	0.1030 ± 0.0698	0.0543	0.1310	78.37
cnae-9	1,080	856	0.1411 ± 0.1310	0.0431	0.1980	0.86
coil-20	1,440	1,024	0.1555 ± 0.1374	0.0627	0.2227	1.26
micro-mass	360	1,300	0.2492 ± 0.2192	0.0709	0.3936	0.52
SRBCT	83	2,308	0.4087 ± 0.2462	0.2326	0.5408	0.11
Olivetti_Faces	400	4,096	0.3520 ± 0.1804	0.2189	0.4620	0.88
DLBCL	77	5,469	0.1886 ± 0.1832	0.0688	0.2282	0.17
leukemia	72	7,129	0.3355 ± 0.2785	0.0988	0.5268	0.14
UMIST_Faces_Cropped	575	10,304	0.1451 ± 0.1079	0.0727	0.1985	2.10
AP_Omentum_Kidney	337	10,935	0.1619 ± 0.1363	0.0494	0.2400	1.14
AP_Lung_Kidney	386	10,935	0.2654 ± 0.2051	0.0764	0.3965	1.47
AP_Breast_Colon	630	10,935	0.2332 ± 0.1683	0.1103	0.2993	2.32
OVA_Breast	1,545	10,935	0.1886 ± 0.1295	0.1002	0.2490	7.93
MLL	72	12,582	0.2863 ± 0.2308	0.0910	0.4906	0.18
GCM	190	16,063	0.1248 ± 0.1536	0.0279	0.1582	0.76
SMK	187	19,993	0.1845 ± 0.1855	0.0947	0.1779	0.93
GLI	85	22,283	0.3725 ± 0.2457	0.1699	0.4995	0.31
hepatitisC	283	54,621	0.4604 ± 0.1846	0.3356	0.5838	3.05

estimator with a low-rank approximation suited to the neighborhood structures typical of many real-world datasets, thereby reducing computational cost by several orders of magnitude while retaining the geometric fidelity of the original formulation.

Theoretical analysis confirms the correctness of the reformulation and justifies the proposed approximation, while extensive experiments show that MeCuCo delivers substantial speedups across diverse real-world datasets, closely matching the original MCBP estimator's output, and remains computationally feasible in high-dimensional settings where prior methods fail.

Beyond its efficiency, MeCuCo is positioned as a foundational tool for geometry-aware ML: local mean curvature offers structural insight into manifolds beyond what density- or distance-based measures provide, with potential applications in representation learning, clustering, anomaly detection, active learning, and graph learning.

By converting curvature estimation from a computational bottleneck into a scalable procedure, this work extends the practical use of differential-geometric information in ML, positioning local curvature as a viable geometric primitive for future models to capture the intrinsic shape of data manifolds.

Future research should aim to establish tighter theoretical approximation guarantees, while pursuing algorithmic advances — such as randomized matrix decompositions and GPU-based implementations — to further enhance scalability. Of particular significance is the prospect of incorporating curvature directly into learning objectives, geometric regularization schemes, representation-, graph-, and deep-learning frameworks. The scalable accessibility of local curvature thus lays the groundwork for a novel class of curvature-aware ML methods, in which differential-geometric structure is treated as a fundamental inductive bias.

AI USE STATEMENT

In this work, we used generative AI tools to polish writing. We have not used generative AI tools to generate synthetic data sets, interpret results, implement methods, or tasks regarding the support for mathematical proofs. Lastly, we take responsibility for the final content of this work.

REFERENCES

E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition, 1999. ISBN 0-89871-447-8 (paperback).

Yasuhiko Asao and Yuichi Ike. Curvature of point clouds through principal component analysis. *arXiv preprint arXiv:2106.09972*, 2022. doi: 10.48550/arXiv.2106.09972. URL https://arxiv.org/abs/2106.09972.

Bernd Bischl, Giuseppe Casalicchio, Matthias Feurer, Pieter Gijsbers, Frank Hutter, Michel Lang, Rafael G. Mantovani, Jan N. van Rijn, and Joaquin Vanschoren. OpenML benchmarking suites. In *Advances in Neural Information Processing Systems (NeurIPS) Track on Datasets and Benchmarks*, 2021. URL https://arxiv.org/abs/1708.03731.

Sourav Chatterjee. A new coefficient of correlation. *Journal of the American Statistical Association*, 116(536):2009–2022, 2021. doi: 10.1080/01621459.2020.1758115.

Ming-Yen Cheng and Hau-Tieng Wu. Efficient Weingarten map and curvature estimation on manifolds. *Machine Learning*, 110(6):1467–1510, 2021. doi: 10.1007/s10994-021-05953-4.

Manfredo P. do Carmo. *Differential Geometry of Curves and Surfaces: Revised and Updated Second Edition*. Dover Publications, New York, UNITED STATES, 2016. ISBN 978-0-486-81797-2.

Charles Fefferman, Sanjoy Mitter, and Hariharan Narayanan. Testing the manifold hypothesis. *Journal of the American Mathematical Society*, 29(4):983–1049, 2016. doi: 10.1090/jams/852.

A. L. M. Levada. A mean curvature approach to boundary detection: Geometric insights for unsupervised learning. *arXiv preprint arXiv:2605.04274*, 2025. doi: 10.48550/arXiv.2605.04274. URL https://arxiv.org/abs/2605.04274.

Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction, 2018a. URL http://arxiv.org/abs/1802.03426. cite arxiv:1802.03426Comment: Reference implementation available at http://github.com/lmcinnes/umap.

Leland McInnes, John Healy, Nathaniel Saul, and Lukas Großberger. Umap: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3(29):861, 2018b. doi: 10.21105/joss.00861. URL https://doi.org/10.21105/joss.00861.

Elizabeth S. Meckes. *The Random Matrix Theory of the Classical Compact Groups*. Cambridge Tracts in Mathematics. Cambridge University Press, 2019.

Tristan Needham. *Visual Differential Geometry and Forms: A Mathematical Drama in Five Acts*. Princeton University Press, Princeton, NJ, 2021. ISBN 978-0-691-20369-0.

Barrett O'Neill. *Elementary Differential Geometry*. Academic Press, 2nd edition, 2006.

John Oprea. *Differential Geometry and its Applications*. The Mathematical Association of America, New York, 2 edition, 2007.

Mathilde Papillon, Sophia Sanborn, Johan Mathe, Louisa Cornelis, Abby Bertics, Domas Buracas, Hansen J. Lillemark, Christian Shewmake, Fatih Dinc, Xavier Pennec, and Nina Miolane. Beyond Euclid: An illustrated guide to modern machine learning with geometric, topological, and algebraic structures. *Machine Learning: Science and Technology*, 6, 2025. doi: 10.1088/2632-2153/adf375.

Sam T. Roweis and Lawrence K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000. doi: 10.1126/science.290.5500.2323.

A. Singer and H.-T. Wu. Vector diffusion maps and the connection laplacian. *Communications on Pure and Applied Mathematics*, 65(8):1067–1144, 2012. doi: https://doi.org/10.1002/cpa.21395.

Michael Spivak. *A Comprehensive Introduction to Differential Geometry*. Publish or Perish, 1999.

Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000. doi: 10.1126/science.290.5500.2319.

Loring W Tu. *Differential geometry: connections, curvature, and characteristic classes*, volume 275. Springer, 2017.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008. URL `http://www.jmlr.org/papers/v9/vandermaaten08a.html`.

Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. OpenML: Networked science in machine learning. *SIGKDD Explorations Newsletter*, 15(2):49–60, 2013. doi: 10.1145/2641190.2641198.

Melanie Weber. Geometric machine learning. *AI Magazine*, 46(1):e12210, 2025. doi: https://doi.org/10.1002/aaai.12210.