Self-Supervised Learning by Curvature Alignment

Benyamin Ghojogh[1*] M. Hadi Sepanj[2] Paul Fieguth[2]
[1]Artificial Intelligence Scientist, Waterloo, Ontario, Canada
[2]Vision and Image Processing Group, Systems Design Engineering, University of Waterloo
{bghojogh, mhsepanj, paul.fieguth}@uwaterloo.ca

Abstract

Non-contrastive self-supervised learning (SSL) shapes first- and second-order feature statistics but largely ignores the local geometry of the data manifold. We introduce CurvSSL *and its RKHS extension* kernel CurvSSL*, which augment a standard two-view encoder–projector and Barlow Twins–style loss with a curvature-based regularizer. Each embedding obtains a discrete curvature score from its k nearest neighbors—via cosine interactions on the unit hypersphere or, in the kernel version, a normalized local Gram matrix. A Barlow-style loss aligns curvature across augmentations and decorrelates curvature patterns across samples, enforcing both invariance and consistent local manifold bending. Experiments on MNIST and CIFAR-10 with ResNet-18 show competitive or improved linear evaluation over Barlow Twins and VICReg, highlighting curvature as an effective complement to statistical SSL regularization.*

1. Introduction

Self-supervised learning (SSL) replaces labels with surrogate objectives over augmented views [18, 23]. Contrastive InfoNCE methods maximize agreement between positives while repelling negatives [3, 22], whereas non-contrastive approaches avoid negatives using invariance plus variance/covariance or redundancy-reduction penalties [2, 9, 21, 24]; examples include Barlow Twins [25] and VICReg [2]. These methods mostly regularize first- and second-order statistics in a flat Euclidean embedding space, but high-dimensional data typically lie near lower-dimensional manifolds and standard SSL does not explicitly control the *local* manifold geometry, so augmentations can be Euclidean-close yet induce different local neighborhoods or tangents—distorting structure important for retrieval, clustering, or semi-supervised learning.

Curvature quantifies local bending: classical angular-defect constructions measure deviation from 2π around a vertex [4, 5, 11, 16, 19], and one can view each data point as a vertex whose k nearest neighbors form faces [6]; translating neighbors to the origin, projecting them onto the unit hypersphere, and aggregating cosine similarities yields a scalar *curvature score*. This idea extends to RKHS via kernel inner products and normalized Gram matrices [8, 24].

Motivated by this, we propose *curvature-regularized SSL (CurvSSL)* and its kernel variant. Using a standard two-view encoder–projector and a Barlow Twins–style redundancy-reduction term [25], we compute discrete curvature scores from each projected embedding's k nearest neighbors and add a geometry-aware regularizer that aligns curvature across augmentations and decorrelates curvature patterns across samples via a Barlow Twins–style loss on curvature matrices. Thus CurvSSL preserves invariance and redundancy reduction while explicitly encouraging consistency and diversity in local manifold bending (and in RKHS via kernelized curvature), yielding competitive ResNet-based [10] representations and showing that shaping local geometry complements purely statistical regularizers.

2. Background on Polyhedron Curvature and Angular Defect

A *polytope* in $\mathbb{R}^d$ has planar faces; in $\mathbb{R}^2$ and $\mathbb{R}^3$ these are polygons and polyhedra. Examples include cube, tetrahedron, octahedron, icosahedron, and dodecahedron [4]. For a polygon, interior and exterior angles satisfy $\tau_j + \mu_j = \pi$. Similarly, for polyhedra (Fig. 1-a), the intersection of the unit sphere at a vertex with its opposite cone forms a spherical polygon. Harriot's 1603 theorem [16] states that for a spherical triangle:

$$\mu_1 + \mu_2 + \mu_3 - \pi = 2\pi - (\tau_1 + \tau_2 + \tau_3),$$

generalizing for a k-gon to

$$\mu_1 + \cdots + \mu_k - k\pi + 2\pi = 2\pi - \sum_{a=1}^{k} \tau_a, \qquad (1)$$

*Benyamin Ghojogh and M. Hadi Sepanj contributed equally to this work.

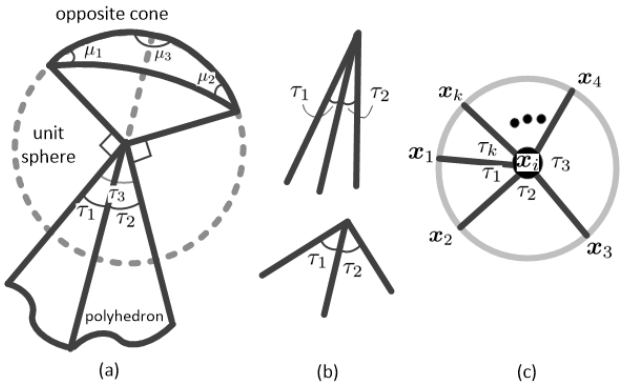


Figure 1. (a) Polyhedron vertex, unit sphere, and the opposite cone, (b) large and small curvature, (c) a point and its neighbors normalized on a unit hyper-sphere around it.

where k faces meet at the vertex.

Descartes defined the *angular defect* at a vertex $\boldsymbol{x}$ [5]:

$$\mathcal{D}(\boldsymbol{x}) := 2\pi - \sum_{a=1}^{k} \tau_a. \tag{2}$$

The total defect over all vertices with v vertices, e edges, and f faces is

$$\mathcal{D} := \sum_{i=1}^{v} \mathcal{D}(\boldsymbol{x}_i) = 2\pi(v - e + f), \tag{3}$$

where $v-e+f$ is the Euler–Poincaré characteristic [11, 19]. Smaller interior angles τ correspond to sharper corners (Fig. 1-b), so angular defect naturally defines vertex *curvature*.

3. Curvature Calculation for Data Points

3.1. Curvature Calculation in the Input Space

Following [6], each data point $\boldsymbol{x}_i$ is treated as a vertex of a hypothetical polyhedron whose k nearest neighbors form the k faces meeting at that vertex (Fig. 1-a). Points with higher curvature are more anomalous, motivating a *curvature score* $c(\boldsymbol{x}_i)$.

Since curvature is inversely proportional to the angles τ_a between edges, we use cosine to define the score:

$$c(\boldsymbol{x}_i) \propto \frac{1}{\tau_a} \propto \cos(\tau_a). \tag{4}$$

Initially, summing over faces gives:

$$c(\boldsymbol{x}_i) := \sum_{a=1}^{k} \cos(\tau_a) = \sum_{a=1}^{k} \frac{\breve{\boldsymbol{x}}_a^\top \breve{\boldsymbol{x}}_{a+1}}{||\breve{\boldsymbol{x}}_a||_2 ||\breve{\boldsymbol{x}}_{a+1}||_2}, \tag{5}$$

where $\breve{\boldsymbol{x}}_a := \boldsymbol{x}_a - \boldsymbol{x}_i$ and $\breve{\boldsymbol{x}}_{k+1} = \breve{\boldsymbol{x}}_1$.

To simplify, we relax this by summing cosines over all edge pairs:

$$c(\boldsymbol{x}_i) := \sum_{a=1}^{k-1} \sum_{b=a+1}^{k} \frac{\breve{\boldsymbol{x}}_a^\top \breve{\boldsymbol{x}}_b}{||\breve{\boldsymbol{x}}_a||_2 ||\breve{\boldsymbol{x}}_b||_2}, \tag{6}$$

with $\breve{\boldsymbol{x}}_a = \boldsymbol{x}_a - \boldsymbol{x}_i$, $\breve{\boldsymbol{x}}_b = \boldsymbol{x}_b - \boldsymbol{x}_i$. This corresponds to normalizing neighbors onto the unit hypersphere (Fig. 1-c).

The relaxation is valid because if edges belong to the same face, the calculation is exact; if not, changes in their angles correlate with changes in angles of edges on the same faces, maintaining consistency.

3.2. Curvature Calculation in the RKHS

To capture nonlinear patterns, we compute *kernel curvature* in the RKHS [6]. Let $\boldsymbol{\phi} : \mathcal{X} \to \mathcal{H}$ map $\boldsymbol{x} \in \mathcal{X}$ to RKHS $\mathcal{H}$, with $\boldsymbol{\phi}(\boldsymbol{x}) \in \mathbb{R}^t$ ($t \gg d$). The kernel is the inner product [7, 12]:

$$k(\boldsymbol{x}_1, \boldsymbol{x}_2) := \boldsymbol{\phi}(\boldsymbol{x}_1)^\top \boldsymbol{\phi}(\boldsymbol{x}_2), \tag{7}$$

and the RKHS distance is [20]:

$$||\boldsymbol{\phi}(\boldsymbol{x}_i) - \boldsymbol{\phi}(\boldsymbol{x}_j)||_2 = \sqrt{k(\boldsymbol{x}_i, \boldsymbol{x}_i) - 2k(\boldsymbol{x}_i, \boldsymbol{x}_j) + k(\boldsymbol{x}_j, \boldsymbol{x}_j)}. \tag{8}$$

This distance is used to find k-NN in the RKHS.

For curvature, edges $\breve{\boldsymbol{x}}_a, \breve{\boldsymbol{x}}_b$ are mapped to RKHS, replacing inner products with kernel values: $k(\breve{\boldsymbol{x}}_a, \breve{\boldsymbol{x}}_b) = \boldsymbol{\phi}(\breve{\boldsymbol{x}}_a)^\top \boldsymbol{\phi}(\breve{\boldsymbol{x}}_b)$. Let $\boldsymbol{K}_i \in \mathbb{R}^{k \times k}$ be the kernel matrix of neighbors of $\boldsymbol{x}_i$. Normalizing the vectors corresponds to normalized kernel [1, 7]:

$$k'(\breve{\boldsymbol{x}}_a, \breve{\boldsymbol{x}}_b) := \frac{k(\breve{\boldsymbol{x}}_a, \breve{\boldsymbol{x}}_b)}{\sqrt{k(\breve{\boldsymbol{x}}_a, \breve{\boldsymbol{x}}_a)\, k(\breve{\boldsymbol{x}}_b, \breve{\boldsymbol{x}}_b)}}. \tag{9}$$

The kernel curvature score is then:

$$c(\boldsymbol{x}_i) := \sum_{a=1}^{k-1} \sum_{b=a+1}^{k} \boldsymbol{K}'_{i,ab}, \tag{10}$$

where $\boldsymbol{K}'_{i,ab}$ is the (a, b)-th element of the normalized kernel $\boldsymbol{K}'_i$.

4. CurvSSL and Kernel CurvSSL

4.1. Network and Data Settings

The neural network for self-supervised learning contains an encoder f_θ followed by a projection head g. Let $\mathcal{X} \subset \mathbb{R}^d$ be the input space, $f_\theta : \mathcal{X} \to \mathbb{R}^{d_h}$ an encoder, and $g : \mathbb{R}^{d_h} \to \mathbb{R}^{d_z}$ a projection head. Suppose $\mathcal{T}(\boldsymbol{x})$ denotes the distribution of training data. For every training data instance,

we draw two stochastic augmentations $(\boldsymbol{x}, \boldsymbol{x}') \sim \mathcal{T}(\boldsymbol{x})$ and pass them through the encoder and the projection head:

$$\begin{aligned} \boldsymbol{h} &= f_\theta(\boldsymbol{x}) \in \mathbb{R}^{d_h}, \quad \boldsymbol{z} = g(\boldsymbol{h}) \in \mathbb{R}^{d_z}, \\ \boldsymbol{h}' &= f_\theta(\boldsymbol{x}') \in \mathbb{R}^{d_h}, \quad \boldsymbol{z}' = g(\boldsymbol{h}') \in \mathbb{R}^{d_z}. \end{aligned} \tag{11}$$

Every mini-batch, with size b, is $\{(\boldsymbol{z}_i, \boldsymbol{z}'_i)\}_{i=1}^b$.

4.2. Loss Function

We now describe the proposed self-supervised objective. As before, let $\{(\boldsymbol{z}_i, \boldsymbol{z}'_i)\}_{i=1}^b$ denote the projected embeddings of two augmentations of a mini-batch of size b, where $\boldsymbol{z}_i = g(f_\theta(\boldsymbol{x}_i))$ and $\boldsymbol{z}'_i = g(f_\theta(\boldsymbol{x}'_i)) \in \mathbb{R}^{d_z}$. Our loss has two components: (i) a redundancy-reduction term on the embedding coordinates, in the spirit of Barlow Twins, and (ii) a curvature-based term that aligns and decorrelates curvature patterns across the batch.

4.2.1. Redundancy reduction in embedding space

We first normalize the projected embeddings per feature dimension:

$$\tilde{\boldsymbol{z}}_i = \frac{\boldsymbol{z}_i - \boldsymbol{\mu}_z}{\boldsymbol{\sigma}_z + \varepsilon}, \qquad \tilde{\boldsymbol{z}}'_i = \frac{\boldsymbol{z}'_i - \boldsymbol{\mu}'_z}{\boldsymbol{\sigma}'_z + \varepsilon}, \tag{12}$$

where the division is element-wise, $\boldsymbol{\mu}_z, \boldsymbol{\sigma}_z \in \mathbb{R}^{d_z}$ are the batch-wise mean and standard deviation of $\{\boldsymbol{z}_i\}_{i=1}^b$, and similarly for $\boldsymbol{\mu}'_z, \boldsymbol{\sigma}'_z$ and $\{\boldsymbol{z}'_i\}_{i=1}^b$; $\varepsilon > 0$ is a small constant for numerical stability. We then form the cross-correlation matrix:

$$\boldsymbol{C} \in \mathbb{R}^{d_z \times d_z}, \qquad \boldsymbol{C}_{uv} := \frac{1}{b} \sum_{i=1}^{b} \tilde{z}_{i,u}\, \tilde{z}'_{i,v}, \tag{13}$$

where $\boldsymbol{C}_{uv}$ is the (u, v)-th element of $\boldsymbol{C}$ and $\tilde{z}_{i,u}$ denotes the u-th component of $\tilde{\boldsymbol{z}}_i$. Following Barlow Twins [25], we enforce that the diagonal entries of $\boldsymbol{C}$ are close to 1 (strong agreement between views in each feature) while off-diagonal entries are close to 0 (low redundancy between different features):

$$\mathcal{L}_{\text{emb}} := \sum_{u=1}^{d_z} (\boldsymbol{C}_{uu} - 1)^2 \;+\; \lambda_{\text{emb}} \sum_{\substack{u,v=1 \\ u \neq v}}^{d_z} \boldsymbol{C}_{uv}^2, \tag{14}$$

where $\lambda_{\text{emb}} > 0$ controls the strength of the off-diagonal penalty.

4.2.2. Curvature-based regularization

In addition to redundancy reduction at the coordinate level, we regularize the *local geometry* of the learned manifold via curvature. For each embedding $\boldsymbol{z}_i$, we compute a discrete curvature score $c(\boldsymbol{z}_i)$ by treating $\boldsymbol{z}_i$ as a vertex of a hypothetical polyhedron whose faces are spanned by its k-nearest neighbors in the embedding space. Let $\{\boldsymbol{z}_{i,a}\}_{a=1}^k$ denote these neighbors, and define edge vectors $\breve{\boldsymbol{z}}_{i,a} := \boldsymbol{z}_{i,a} - \boldsymbol{z}_i$. Normalizing these edges onto the unit hypersphere and aggregating the pairwise cosine similarities between neighbor directions yields the curvature score (see Eq. (6)):

$$c(\boldsymbol{z}_i) := \sum_{a=1}^{k-1} \sum_{b=a+1}^{k} \frac{\breve{\boldsymbol{z}}_{i,a}^\top \breve{\boldsymbol{z}}_{i,b}}{\|\breve{\boldsymbol{z}}_{i,a}\|_2 \, \|\breve{\boldsymbol{z}}_{i,b}\|_2}, \tag{15}$$

which measures how sharply the local neighborhood around $\boldsymbol{z}_i$ bends. The kernel curvature score is (see Eq. (10)):

$$c(\boldsymbol{z}_i) := \sum_{a=1}^{k-1} \sum_{b=a+1}^{k} \boldsymbol{K}'_{i,ab}, \tag{16}$$

where $\boldsymbol{K}'_{i,ab}$ is the (a, b)-th element of the normalized kernel kernel $\boldsymbol{K}'_i$ between $\breve{\boldsymbol{z}}_{i,a}$ and $\breve{\boldsymbol{z}}_{i,b}$.

Eqs. (15) and (16) can be used for curvature scores in CurvSSL and kernel CurvSSL loss functions, respectively. We compute analogous curvature scores $c(\boldsymbol{z}'_i)$ for the second view.

Stacking the curvature scores into vectors $\boldsymbol{c} = [c(\boldsymbol{z}_1), \ldots, c(\boldsymbol{z}_b)]^\top$ and $\boldsymbol{c}' = [c(\boldsymbol{z}'_1), \ldots, c(\boldsymbol{z}'_b)]^\top \in \mathbb{R}^b$, we first normalize them across the batch:

$$\tilde{\boldsymbol{c}} = \frac{\boldsymbol{c} - \mu_c \mathbf{1}}{\sigma_c + \varepsilon}, \qquad \tilde{\boldsymbol{c}}' = \frac{\boldsymbol{c}' - \mu'_c \mathbf{1}}{\sigma'_c + \varepsilon}, \tag{17}$$

where $\mu_c, \sigma_c \in \mathbb{R}$ are the mean and standard deviation of $\boldsymbol{c}$, μ'_c, σ'_c are those of $\boldsymbol{c}'$, the $\mathbf{1} \in \mathbb{R}^b$ is the all-ones vector, and $\varepsilon > 0$ is again a small constant. We then form a curvature-derived matrix:

$$\boldsymbol{M} \in \mathbb{R}^{b \times b}, \qquad \boldsymbol{M}_{ij} := \frac{1}{b}\, \tilde{c}_i\, \tilde{c}'_j, \tag{18}$$

where $\boldsymbol{M}_{ij}$ denotes the (i, j)-th element of $\boldsymbol{M}$, which plays an analogous role to the cross-correlation matrix $\boldsymbol{C}$, but now at the *sample* level in terms of curvature. We encourage the curvature of matched augmentations to agree (diagonal entries of $\boldsymbol{M}$ close to 1) and the curvature patterns of different samples to be decorrelated (off-diagonals close to 0):

$$\mathcal{L}_{\text{curv}} := \sum_{i=1}^{b} (\boldsymbol{M}_{ii} - 1)^2 \;+\; \lambda_{\text{curv}} \sum_{\substack{i,j=1 \\ i \neq j}}^{b} \boldsymbol{M}_{ij}^2, \tag{19}$$

where $\lambda_{\text{curv}} > 0$ controls the strength of curvature-based redundancy reduction.

4.2.3. Total objective and kernel extension

Our final self-supervised loss is a weighted sum of the embedding-level and curvature-level terms:

$$\mathcal{L} := \mathcal{L}_{\text{emb}} \;+\; \alpha_{\text{curv}}\, \mathcal{L}_{\text{curv}}, \tag{20}$$

where $\alpha_{\text{curv}} > 0$ balances the influence of curvature regularization. In the Euclidean case, i.e., CurvSSL, $c(\cdot)$ is given by Eq. (15). In the kernel curvature variant, i.e., kernel CurvSSL, Eq. (16) is used for $c(\cdot)$.

The proposed objective enforces view invariance and redundancy reduction at the level of embedding coordinates, while simultaneously shaping the local manifold geometry through curvature alignment and curvature-based decorrelation across the batch.

5. Experiments

We empirically evaluate the proposed curvature-regularized self-supervised learning on two standard benchmarks, MNIST [15] and CIFAR-10 [14], using a ResNet backbone [10] and a two-stage protocol: (i) self-supervised pretraining with the proposed CurvSSL and kernel CurvSSL objectives, and (ii) frozen-encoder linear evaluation. In addition, we visualize the learned representations with UMAP [17] to inspect the geometry induced by curvature regularization.

5.1. Experimental Setup

Datasets. We evaluate on MNIST [15] and CIFAR-10 [14], using the training split for SSL pretraining and standard train/test splits for linear evaluation.

Network and training. A ResNet-18 encoder [10] with a two-layer MLP projector maps features to $d_z = 128$. We train using the total loss (20), where $\mathcal{L}_{\text{emb}}$ is Barlow Twins (14) and $\mathcal{L}_{\text{curv}}$ uses discrete or kernel curvature scores (15),(16) with $k = 10$ neighbors. Models are trained with Adam [13] (lr 10^{-3}, wd 10^{-4}, batch 256) for 100 epochs on MNIST and 500 on CIFAR-10. Weights are fixed at $(\lambda_{\text{emb}}, \lambda_{\text{curv}}, \alpha_{\text{curv}}) = (1, 1, 1)$. Kernel CurvSSL uses an RBF kernel.

Augmentations. MNIST uses random crops, small rotations, RGB conversion, and normalization. CIFAR-10 uses random crops, flips, color jitter, grayscale, and normalization. Two augmented views per image pass through the shared encoder–projector.

5.2. Linear Evaluation

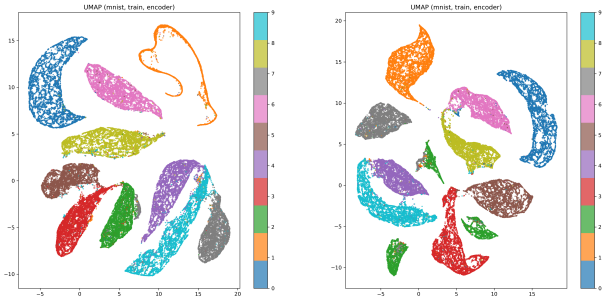
Following standard protocol [2], we freeze f_θ, discard g, and train a small linear classifier (one hidden layer + batch norm) for 50 epochs with SGD. Table 1 reports top-1 accuracy. CurvSSL and kernel CurvSSL achieve competitive results on MNIST and CIFAR-10; the curvature term preserves discriminative features, and kernel CurvSSL typically performs best due to improved nonlinear modeling.

5.3. UMAP Visualization

UMAP (with $n_{\text{neighbors}} = 15$, min_dist $= 0.1$) applied to the learned encoder features shows clear digit clusters on MNIST (Fig. 2), with curvature regularization producing tighter local neighborhoods. On CIFAR-10 (Fig. 3), both CurvSSL variants form meaningful class regions despite higher variability.

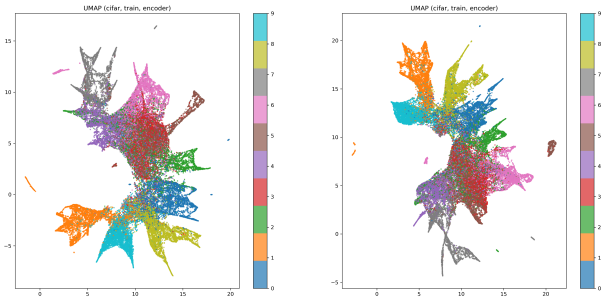
Table 1. Linear evaluation accuracy (%) on MNIST and CIFAR-10 using a frozen ResNet-18 pretrained with each SSL method.

Method	MNIST	CIFAR-10
VicReg [2]	*95.9*	*74.5*
Barlow Twins [25]	*94.9*	*73.6*
CurvSSL (ours)	***97.9***	***75.1***
Kernel CurvSSL (ours)	***98.4***	***76.5***



(a) CurvSSL (Euclidean). (b) Kernel CurvSSL.

Figure 2. UMAP of MNIST encoder features after curvature-regularized SSL, colored by digit class.



(a) CurvSSL (Euclidean). (b) Kernel CurvSSL.

Figure 3. UMAP of CIFAR-10 encoder features after curvature-regularized SSL, colored by class.

6. Conclusion

We introduced CurvSSL and kernel CurvSSL, which add curvature-based regularization to a Barlow Twins–style SSL objective to encourage both invariance and consistent local manifold geometry. Experiments on MNIST and CIFAR-10 show competitive accuracy and improved structure, suggesting that modeling local geometry is a simple and effective complement to standard SSL methods.

References

[1] Julien Ah-Pine. Normalized kernels as similarity indices. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 362–373. Springer, 2010. 2

[2] Adrien Bardes, Jean Ponce, and Yann LeCun. VICReg: Variance-invariance-covariance regularization for self-supervised learning. In *International Conference on Learning Representations*, 2022. 1, 4

[3] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PmLR, 2020. 1

[4] Harold Scott Macdonald Coxeter. *Regular polytopes*. Courier Corporation, 1973. 1

[5] René Descartes. Progymnasmata de solidorum elementis. *Oeuvres de Descartes*, X:265–276, 1890. 1, 2

[6] Benyamin Ghojogh, Fakhri Karray, and Mark Crowley. Anomaly detection and prototype selection using polyhedron curvature. In *Canadian Conference on Artificial Intelligence*, pages 238–250. Springer, 2020. 1, 2

[7] Benyamin Ghojogh, Mark Crowley, Fakhri Karray, and Ali Ghodsi. Background on kernels. *Elements of Dimensionality Reduction and Manifold Learning*, pages 43–73, 2023. 2

[8] Arthur Gretton. Introduction to RKHS, and some simple kernel algorithms. *Adv. Top. Mach. Learn. Lecture Conducted from University College London*, 16(5-3):2, 2013. 1

[9] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020. 1

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1, 4

[11] Peter Hilton and Jean Pedersen. Descartes, Euler, Poincare, Polya and polyhedra. *Séminaire de Philosophie et Mathématiques*, (8):1–17, 1982. 1, 2

[12] Thomas Hofmann, Bernhard Schölkopf, and Alexander J Smola. Kernel methods in machine learning. *The annals of statistics*, pages 1171–1220, 2008. 2

[13] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 4

[14] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, ON, Canada, 2009. 4

[15] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 4

[16] Steenn Markvorsen. Curvature and shape. In *Yugoslav Geometrical Seminar, Fall School of Differential Geometry, Yugoslavia*, pages 55–75, 1996. 1

[17] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018. 4

[18] Veenu Rani, Syed Tufael Nabi, Munish Kumar, Ajay Mittal, and Krishan Kumar. Self-supervised learning: A succinct review. *Archives of Computational Methods in Engineering*, 30(4):2761–2775, 2023. 1

[19] David S Richeson. *Euler's Gem: The Polyhedron Formula and the Birth of Topology*. Princeton University Press, 2019. 1, 2

[20] Bernhard Schölkopf. The kernel trick for distances. In *Advances in neural information processing systems*, pages 301–307, 2001. 2

[21] Hadi Sepanj and Paul Fieguth. Aligning feature distributions in VICReg using maximum mean discrepancy for enhanced manifold awareness in self-supervised representation learning. *Journal of Computational Vision and Imaging Systems*, 10(1):13–18, 2024. 1

[22] M Hadi Sepanj and Paul Fiegth. SinSim: Sinkhorn-regularized SimCLR. *arXiv preprint arXiv:2502.10478*, 2025. 1

[23] M. Hadi Sepanj, Benyamin Ghojogh, and Paul Fieguth. Self-supervised learning using nonlinear dependence. *IEEE Access*, 13:190582–190589, 2025. 1

[24] M Hadi Sepanj, Benyamin Ghojogh, and Paul Fieguth. Kernel VICReg for self-supervised learning in reproducing kernel Hilbert space. *arXiv preprint arXiv:2509.07289*, 2025. 1

[25] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *International conference on machine learning*, pages 12310–12320. PMLR, 2021. 1, 3, 4