Discrete Geometry for Comparing and Transferring Neural Representations

Master's thesis in Complex Adaptive Systems

ATHARVA KHANDAIT

DEPARTMENT OF MATHEMATICAL SCIENCES

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026
www.chalmers.se

Master's thesis 2026

Discrete Geometry for Comparing and Transferring Neural Representations

ATHARVA KHANDAIT

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mathematical Sciences
Division of Algebra and Geometry
Geometry, Algebra and Physics in Deep Neural Networks (GAPinDNNs)
Chalmers University of Technology
Gothenburg, Sweden 2026

Discrete Geometry for Comparing and Transferring Neural Representations
ATHARVA KHANDAIT

Supervisor: Jan E. Gerken, Department of Mathematical Sciences
Examiner: Jan E. Gerken, Department of Mathematical Sciences

Master's Thesis 2026
Department of Mathematical Sciences
Division of Algebra and Geometry
Geometry, Algebra and Physics in Deep Neural Networks (GAPinDNNs)
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2026

Discrete Geometry for Comparing and Transferring Neural Representations
ATHARVA KHANDAIT
Department of Mathematical Sciences
Chalmers University of Technology

Abstract

Neural networks transform data through a sequence of intermediate representations, but comparing and transferring the structure of these representations remains challenging. This thesis develops a geometric framework for neural representations based on manifold learning, representational similarity, and diffusion geometry, incorporating tools from multi-view learning into this field for the first time. A key contribution is an exact Markov reformulation of a broad class of centered, scale-invariant RSM-based similarity measures in terms of row-stochastic Markov matrices, which then opens the door to manipulations from diffusion geometry.

Building on this, the thesis introduces multi-scale variants of CKA and DistCorr, which compare powers of the associated Markov operators, and alternating-diffusion variants, which fuse the Markov matrices of several layers into a single network-level operator. Empirically, these diffusion-based measures achieve state-of-the-art performance in accuracy and output correlation for both language and vision tasks across different models, on the Representational Similarity (ReSi) benchmark. They also obtain the best results on an additional out-of-distribution challenge benchmark.

The thesis further applies the geometric viewpoint to knowledge distillation. A graph-Laplacian distillation objective is proposed, in which the student is trained to match the teacher's sample geometry rather than activation coordinates. Together, these results show that operator-based discrete geometry provides a useful language for comparing, aggregating, and transferring neural representations.

Keywords: neural representations, diffusion geometry, representational similarity, Markov operators, graph Laplacians, multi-view learning, sensor fusion, knowledge distillation.

Acknowledgements

The work presented in this thesis has been a remarkable journey over the past year, full of both challenges and rewarding moments. It has been an invaluable learning experience, and has helped define a sense of the career and life in research that I hope to pursue. Looking back, the process of developing an idea, refining it through theory and experiments, and turning it into a complete thesis has taught me far more than I could have anticipated at the beginning.

I would like to express my heartfelt gratitude to my supervisor, Jan E. Gerken, for believing in the project from the beginning and for his invaluable guidance throughout the last two semesters. His feedback, insight, and teaching during our many meetings were essential to shaping both the direction and the quality of this work. In retrospect, carrying out a research project of this scale would have seemed insurmountable without his support.

I am also grateful to the members of the GAPinDNNs research group. Attending the group seminars and journal club meetings gave me a valuable introduction to the world of academic research, and helped me better understand how mathematical and machine-learning ideas are developed, discussed, and challenged in a research environment.

The computations in this thesis were enabled by resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS), partially funded by the Swedish Research Council through grant agreement no. 2022-06725.

Finally, I am thankful to everyone who supported me during this process, directly or indirectly, and helped make this thesis possible.

Atharva Khandait, Gothenburg, June 2026

Contents

1

Introduction

Modern neural networks do not merely map inputs to outputs; they also construct a sequence of internal representations. Each layer transforms the data into a new coordinate system in which different aspects of the input may become more explicit, more compressed, or more geometrically organised. Understanding these intermediate representations is therefore central to understanding how neural networks learn, how different models relate to one another, and how useful information can be transferred between models. This thesis studies neural representations from a geometric point of view. The main idea is that a layer representation should not only be viewed as a matrix of activation vectors, but also as a finite sample from an underlying geometric object. This viewpoint makes it possible to use tools from manifold learning, graph Laplacians, diffusion geometry, and representational similarity analysis to compare and transfer neural representations.

The starting point is the manifold hypothesis: high-dimensional data often concentrate near lower-dimensional manifolds. If the input data lie on or near such a manifold, then the image of this data under a neural network layer can be interpreted as a representation manifold. In practice, only finitely many samples are observed, so the geometry of such a representation must be approximated from a point cloud of layer activations. A natural way to do this is to construct a weighted graph on the samples. The corresponding graph Laplacian can be interpreted as a finite-dimensional approximation of the Laplace-Beltrami operator on the underlying manifold, while the row-normalised graph weights define a Markov transition matrix describing a random walk on the representation. This connects local sample neighbourhoods, smoothness, spectral geometry, and diffusion in a single framework.

This geometric perspective is closely related to existing representational similarity methods. Measures such as Centered Kernel Alignment (CKA) and Distance Correlation (DistCorr) compare two neural representations by first converting each representation into a sample-indexed relational matrix, often called a representational similarity matrix (RSM). Instead of comparing activation coordinates directly, an RSM records how samples relate to one another within a representation. This is useful because two layers may have different widths or coordinate systems, but their RSMs always live in the same $(N \times N)$ sample space when evaluated on the same N inputs. Standard RSM-based measures are therefore already geometric in a broad discrete sense: they compare the relational structure induced by two representations.

The main contribution of this thesis is to show that this RSM-based viewpoint

can be connected exactly to diffusion geometry. For a broad class of centered, scale-invariant RSM-based similarity measures, the centered RSM can be embedded into the space of row-stochastic Markov matrices by an affine shift and rescaling. The resulting matrix is a valid Markov transition matrix. This gives an exact reformulation for measures satisfying the required centering and scale-invariance assumptions. CKA and DistCorr are shown to fall within this class.

Once an RSM has been rewritten as a Markov matrix, it can be manipulated using diffusion-geometric operations. This leads to two families of new similarity measures. The first family consists of multi-scale variants, MS-CKA and MS-DistCorr. These replace the Markov matrix by its t-th power before computing the final similarity score. Since powers of a Markov matrix describe multi-step random walks, these measures compare representations not only through direct pairwise relations, but also through coarser connectivity patterns at adjustable diffusion scales. The second family consists of alternating-diffusion variants, AD-CKA and AD-DistCorr. These treat several layers of a network as multiple aligned views of the same input samples. Each layer gives a Markov operator, and the product of these operators fuses the layerwise geometries into a single network-level diffusion operator. This changes the comparison from layer-to-layer to network-to-network.

The empirical results show that the diffusion-geometric reformulation is not merely a formal reinterpretation of existing RSM-based measures, but leads to substantially stronger representation comparison methods. On the ReSi benchmark, the proposed diffusion-based measures achieve state-of-the-art performance in seven of the eight reported prediction-grounded comparisons, across language and vision architectures. The comparison includes a large set of established baselines, including CKA, DistCorr, CCA-based methods, Procrustes-style alignment measures, neighborhood-based measures, topology-based measures, and other RSM-based methods. In language models, the strongest gains come from alternating diffusion. AD-CKA gives the best accuracy-difference correlations for both BERT and SmolLM2, showing that model-level functional differences are better captured by aggregating several layers than by comparing a single hidden representation. For output-distribution differences, AD-DistCorr is particularly effective, obtaining the strongest result for SmolLM2 by a large margin. In vision, the best diffusion-based variant depends on the architecture: Multi-Scale CKA is strongest for ResNet18 and ResNet101, while AD-DistCorr is strongest for VGG11 and VGG19. Thus, the results do not simply show that one proposed measure performs well in one setting; they show that diffusion geometry provides two complementary mechanisms, multi-scale single-layer comparison and alternating-diffusion multi-layer comparison, which together improve over existing similarity measures across several distinct architectural regimes.

The thesis further evaluates the proposed network-level diffusion comparison on the GRS benchmark, focusing on its Benchmark 4, an out-of-distribution 'challenge' setting. This benchmark is especially significant because it was explicitly designed as a difficult case in which existing representation similarity measures achieve only weak correlations with functional behaviour. In this setting, AD-CKA obtains the

best results on both the Antonym and Numerical stress tests and improves over the reported baselines under both Kendall's τ and Spearman's ρ by substantial margins. On the Antonym stress test, AD-CKA improves Kendall's τ from 0.24 to 0.41 and Spearman's ρ from 0.18 to 0.29. On the Numerical stress test, it improves Kendall's τ from 0.12 to 0.20 and Spearman's ρ from 0.08 to 0.14. These results are important because they show that the proposed method improves precisely in a setting where previous methods struggle. They support the central empirical claim: behaviorally relevant similarity between neural networks is often not fully visible in any single layer, but can be recovered more effectively by fusing multi-layer diffusion geometry into a network-level operator. The diffusion-based framework presented in Chapters 4 and 5 is based on the arXiv preprint: From Layers to Networks: Comparing Neural Representations via Diffusion Geometry; and is currently under review.

In addition to post-hoc representation comparison, the thesis applies the same geometric viewpoint to knowledge distillation. In this setting, a smaller student network is trained using structural information supplied by a larger teacher network. Instead of forcing the student to match teacher feature coordinates directly, the proposed manifold knowledge distillation method constructs weighted k-nearest-neighbour graphs from teacher and student representations and matches their symmetric normalised graph Laplacians. Thus, the teacher transfers a discrete geometric operator describing local sample connectivity in representation space. The method also studies practical graph-construction choices, including Gaussian teacher kernels, heavy-tailed and hybrid student kernels, graph symmetrisation, diffusion-power matching, trainable student kernels, and the role of graph batch size.

Using controlled CIFAR-10 experiments, the thesis then shows that graph-Laplacian distillation can recover a substantial part of the teacher-student performance gap. The teacher has about 10.4× more trainable parameters than the student. Without distillation, the student reaches 70.46% test accuracy, while the teacher reaches 77.03%. Adding the proposed manifold distillation loss improves the student to 73.39%, recovering a significant fraction of the gap while keeping the student much smaller. Additional CIFAR-100 experiments on RepDistiller/CRD teacher-student pairs show that the method is competitive with related representation-based and relation-based distillation baselines, especially for teacher-student pairs from the same family.

Overall, the thesis develops a unified geometric view of neural representations. The theoretical part connects RSM-based similarity measures to Markov operators and thereby opens standard similarity measures to diffusion-geometric manipulation. The empirical part shows that multi-scale and alternating-diffusion extensions improve representation comparison on prediction-grounded and out-of-distribution benchmarks. The distillation part shows that graph Laplacians can also serve as training-time objects for transferring representation geometry from a teacher to a student. The main conclusion is that neural representation geometry is often better understood at the level of operators on samples than at the level of raw activation coordinates: graph Laplacians and Markov diffusion matrices provide a flexible lan-

guage for both comparing and transferring the structure learned by neural networks.

2

Mathematical Background for Representation Geometry

This chapter develops the mathematical background for the representation-geometry viewpoint used in this thesis. It treats layer activations as finite samples from representation manifolds, introduces Laplace-Beltrami and graph Laplacian operators, and then describes diffusion geometry, representational similarity matrices, and centered alignment measures.

2.1 The Manifold Hypothesis

A recurring viewpoint in modern machine learning is that high-dimensional data are not distributed uniformly throughout the ambient input space. Instead, natural data often appear to have a much smaller number of effective degrees of freedom than the ambient dimension suggests. This idea is commonly referred to as the *manifold hypothesis* [1, 2]: although an input data point may be represented as a vector $\boldsymbol{x} \in \mathbb{R}^D$, the set of plausible inputs is assumed to concentrate near, or on, a lower-dimensional manifold [3, 4]

$$\mathcal{M} \subset \mathbb{R}^D, \qquad \dim(\mathcal{M}) = m \ll D. \tag{2.1}$$

For example, an image with D greyscale pixels is formally a point in $\mathbb{R}^D$, but the set of images corresponding to meaningful variations of the same object, scene, or semantic class is expected to occupy a much smaller and highly structured subset of this space. The relevant geometry is therefore not necessarily the full Euclidean geometry of $\mathbb{R}^D$, but rather the intrinsic geometry of the data manifold $\mathcal{M}$.

2.1.1 Representation Manifolds

This geometric viewpoint becomes especially relevant when studying neural representations. A neural network defines a sequence of representation maps

$$\Phi_\ell : \mathbb{R}^D \to \mathbb{R}^{d_\ell}, \qquad \ell = 1, \ldots, L, \tag{2.2}$$

where $\Phi_\ell(\boldsymbol{x})$ denotes the activation vector of input $\boldsymbol{x}$ at layer ℓ. If the input data lie on a manifold $\mathcal{M}$, then the representations of the data at layer ℓ lie in the image

$$\mathcal{M}_\ell = \Phi_\ell(\mathcal{M}) \subset \mathbb{R}^{d_\ell}. \tag{2.3}$$

Thus, each layer may be viewed as inducing a new geometric object, the representation manifold $\mathcal{M}_\ell$. This statement should be understood under suitable regularity assumptions. For example, if the restriction $\Phi_\ell|_{\mathcal{M}}$ (the map Φ restricted to $\mathcal{M}$) is a smooth embedding, then $\mathcal{M}_\ell$ is itself a smooth embedded submanifold of $\mathbb{R}^{d_\ell}$. In particular, this requires that Φ_ℓ is smooth on $\mathcal{M}$, is injective on $\mathcal{M}$, and has full-rank differential on the tangent space $T_x\mathcal{M}$ at each point $x \in \mathcal{M}$. For neural networks with nonsmooth activations, such as ReLU, this smooth manifold picture is an idealisation: the layer map is generally only piecewise smooth, and the image may be better regarded as a piecewise-smooth or stratified geometric object.

Even when the ambient dimension d_ℓ is large, the effective intrinsic dimension of $\mathcal{M}_\ell$ may remain much smaller. Empirical studies of deep networks support this perspective: learned representations often have intrinsic dimension far below the number of units in the layer, and this intrinsic dimension can vary across layers [5, 6]. Similarly, other geometric properties, such as curvature, can change through the network and may be related to generalisation or to differences between trained and untrained networks [7, 8].

2.1.2 Finite Samples

According to the manifold hypothesis, the representation of a layer should not be regarded merely as an arbitrary cloud of vectors in $\mathbb{R}^{d_\ell}$. It may instead be useful to regard it as a finite sampling of an underlying geometric object. Given a dataset of N samples

$$\mathcal{X} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\} \subset \mathcal{M}, \tag{2.4}$$

the corresponding layer-ℓ representation point cloud is

$$\mathcal{R}^{(\ell)} = \{\boldsymbol{r}_1^{(\ell)}, \dots, \boldsymbol{r}_N^{(\ell)}\}, \qquad \boldsymbol{r}_i^{(\ell)} = \Phi_\ell(\boldsymbol{x}_i) \in \mathbb{R}^{d_\ell}. \tag{2.5}$$

The object available in practice is only this finite set of samples. Nevertheless, if these samples are assumed to be drawn from a smooth underlying manifold $\mathcal{M}_\ell$, then tools from differential geometry and spectral graph theory provide a way to approximate geometric operators on the manifold using finite-dimensional matrices computed from the point cloud.

2.2 Laplace-Beltrami Geometry

Once a representation is viewed as a sampled geometric object, the next question is which object captures its intrinsic structure. This section introduces the Laplace-Beltrami operator and describes its variational, spectral, and diffusion interpretations.

2.2.1 The Laplace-Beltrami Operator

A particularly natural operator in this setting is the Laplace-Beltrami operator. This choice is canonical: once the Riemannian metric on $\mathcal{M}$ is fixed, the Laplace-

Beltrami operator is fixed as well [3, 4]. In this sense, each Riemannian manifold comes with a distinguished differential operator encoding its intrinsic geometry. Let $\mathcal{M}$ be a smooth, compact Riemannian manifold, possibly embedded in an ambient Euclidean space. For a smooth function $f : \mathcal{M} \to \mathbb{R}$, the Laplace-Beltrami operator $\mathcal{L}_{\mathcal{M}}$ is the manifold analogue of the Euclidean Laplacian Δ. In this thesis, we use the positive semidefinite sign convention [3, 4]

$$\mathcal{L}_{\mathcal{M}} f = -\operatorname{div}_{\mathcal{M}} (\nabla_{\mathcal{M}} f), \tag{2.6}$$

where $\nabla_{\mathcal{M}} f$ is the Riemannian gradient of f and $\operatorname{div}_{\mathcal{M}}$ is the corresponding divergence operator [3, 4]. A useful interpretation is obtained through the associated Dirichlet energy

$$\mathcal{E}_{\mathcal{M}}(f) = \int_{\mathcal{M}} \|\nabla_{\mathcal{M}} f(\boldsymbol{x})\|^2 \, d\operatorname{vol}(\boldsymbol{x}), \tag{2.7}$$

and, under appropriate regularity and boundary assumptions [3, 4],

$$\mathcal{E}_{\mathcal{M}}(f) = \int_{\mathcal{M}} f(\boldsymbol{x}) \, \mathcal{L}_{\mathcal{M}} f(\boldsymbol{x}) \, d\operatorname{vol}(\boldsymbol{x}). \tag{2.8}$$

Thus, the Laplace-Beltrami operator measures how rapidly functions vary over the manifold. Functions with small Dirichlet energy change slowly along the manifold, whereas functions with large energy oscillate rapidly. This variational interpretation is important for representation learning because it connects geometry to functions defined on data.

2.2.2 Spectral and Diffusion Interpretations

A function $f : \mathcal{M}_\ell \to \mathbb{R}$ can be interpreted as a scalar observable on the representation manifold: for example, a coordinate function, a class-score function, or a test function used to probe local geometry. The Laplace-Beltrami operator describes how such observables diffuse, smooth, or vary along the intrinsic geometry of the representation. The eigenvalue problem [3, 4]

$$\mathcal{L}_{\mathcal{M}} \varphi_k = \lambda_k \varphi_k, \qquad 0 = \lambda_0 \le \lambda_1 \le \lambda_2 \le \cdots, \tag{2.9}$$

therefore provides a spectral description of the manifold. The low-frequency eigenfunctions (corresponding to the smallest eigenvalues) are the smoothest nontrivial functions on $\mathcal{M}$ and encode large-scale geometric structure, while higher-frequency eigenfunctions encode progressively finer-scale variation.

The same operator also appears naturally through diffusion [3, 4]. If $u(\boldsymbol{x}, t)$ denotes a time-dependent density or signal on the manifold, then the heat equation can be written as

$$\frac{\partial u}{\partial t} = -\mathcal{L}_{\mathcal{M}} u. \tag{2.10}$$

The solution describes how mass or information diffuses locally along the manifold. Consequently, the Laplace-Beltrami operator can be understood both as a

differential operator and as the infinitesimal generator of a diffusion process. This dual interpretation is one reason that Laplacian-based methods are central in manifold learning: they connect local neighbourhoods, smoothness, heat flow, spectral geometry, and clustering within a single mathematical framework.

2.3 Graph Laplacians

The continuous manifold operator is not directly available from a finite dataset. This section therefore describes how weighted graphs, degree matrices, transition matrices, and graph Laplacians provide finite-dimensional operators on sampled representations.

2.3.1 Weighted Graph Construction

In practice, however, the full manifold $\mathcal{M}_\ell$ is not available. We only observe the finite representation set $\mathcal{R}^{(\ell)} = \{\boldsymbol{r}_1^{(\ell)}, \ldots, \boldsymbol{r}_N^{(\ell)}\}$. The practical replacement for the Laplace-Beltrami operator is therefore a graph Laplacian built on the sampled points. One first constructs a weighted graph

$$G = (V, E, \boldsymbol{W}), \tag{2.11}$$

where $V = \{1, \ldots, N\}$ is the vertex set indexing the sampled representation points, and $E \subseteq V \times V$ is the edge set specifying which pairs of samples are connected. The matrix $\boldsymbol{W} \in \mathbb{R}^{N \times N}$ assigns a nonnegative weight W_{ij} to each edge $(i, j) \in E$, measuring the local similarity between $\boldsymbol{r}_i$ and $\boldsymbol{r}_j$.

A common way to construct the weights is to use local distances between representation vectors. There are two related but distinct choices. In an ε-neighbourhood graph, edges are retained only between points whose distance is below a fixed radius:

$$W_{ij} = \begin{cases} \exp\left(-\dfrac{\|\boldsymbol{r}_i - \boldsymbol{r}_j\|^2}{4\varepsilon}\right), & \text{if } \|\boldsymbol{r}_i - \boldsymbol{r}_j\| \leq \varepsilon, \\ 0, & \text{otherwise.} \end{cases} \tag{2.12}$$

Here ε controls the neighbourhood radius. Alternatively, one may use a fully connected Gaussian kernel graph with bandwidth $\sigma > 0$:

$$W_{ij} = \exp\left(-\frac{\|\boldsymbol{r}_i - \boldsymbol{r}_j\|^2}{2\sigma^2}\right). \tag{2.13}$$

In this case, all pairs of points are connected, but the strength of the edge decays with distance. The parameter σ controls the scale at which two points are considered close. In practice, this Gaussian weighting can also be combined with sparsification, for example, by keeping only the k nearest neighbour (kNN) edges, or as in Equation 2.12.

2.3.2 Degree Matrices and Normalised Laplacians

The degree matrix is the diagonal matrix

$$\boldsymbol{D} = \mathrm{diag}(d_1, \ldots, d_N), \tag{2.14}$$

with diagonal entries

$$D_{ii} = d_i = \sum_{j=1}^{N} W_{ij}. \tag{2.15}$$

Its diagonal entry $D_{ii} = d_i$ is the weighted degree of node i: it measures the total connection strength of sample $\boldsymbol{r}_i$ to the rest of the graph. In a weighted graph, it can be interpreted as a local density or connectivity measure, while in an unweighted graph, this reduces to the number of neighbours of node i. An unweighted graph is a special case of $G = (V, E, \boldsymbol{W})$ in which edges only record whether two vertices are connected, typically with $W_{ij} = 1$ for $(i, j) \in E$ and $W_{ij} = 0$ otherwise.

The unnormalised graph Laplacian is

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}. \tag{2.16}$$

Two common normalised variants are symmetric-normalised Laplacian and random-walk-normalised Laplacian:

$$\boldsymbol{L}_{\mathrm{sym}} = \boldsymbol{I} - \boldsymbol{D}^{-1/2}\boldsymbol{W}\boldsymbol{D}^{-1/2}, \qquad \boldsymbol{L}_{\mathrm{rw}} = \boldsymbol{I} - \boldsymbol{D}^{-1}\boldsymbol{W}, \tag{2.17}$$

where $\boldsymbol{I} \in \mathbb{R}^{N \times N}$ denotes the identity matrix. The matrix

$$\boldsymbol{P} = \boldsymbol{D}^{-1}\boldsymbol{W} \tag{2.18}$$

is row-stochastic, meaning that $P_{ij} \geq 0$ and $\sum_{j=1}^{N} P_{ij} = 1$ for each i. Its entries can therefore be interpreted as transition probabilities:

$$P_{ij} = \mathbb{P}(X_{t+1} = j \mid X_t = i), \tag{2.19}$$

where X_t denotes the position of a random walker on the graph at discrete time t. Starting from node i, the walker moves to node j with probability proportional to the edge weight W_{ij}. Thus, strongly connected nearby samples are more likely to be visited in one step. With this interpretation, applying $\boldsymbol{P}$ to a function on the graph performs one step of local averaging or diffusion over neighboring nodes. The random-walk-normalised Laplacian

$$\boldsymbol{L}_{\mathrm{rw}} = \boldsymbol{I} - \boldsymbol{P} \tag{2.20}$$

therefore measures the difference between a function value and its local weighted average. In this sense, $\boldsymbol{L}_{\mathrm{rw}}$ is the discrete analogue of a diffusion generator, playing the role that the Laplace-Beltrami operator plays for diffusion on a continuous manifold.

Although these objects are written as matrices, they should be interpreted as discrete operators on functions defined over the graph. A vector $\boldsymbol{f} \in \mathbb{R}^N$ can be viewed as a function $\boldsymbol{f} : V \to \mathbb{R}$ assigning a scalar value to each sampled point. Therefore, multiplication by $\boldsymbol{L}$, $\boldsymbol{L}_{\mathrm{sym}}$, $\boldsymbol{L}_{\mathrm{rw}}$, or $\boldsymbol{P}$ defines an operator acting on such functions with finite domains, analogous to how a differential operator acts on functions defined on a continuous manifold.

2.3.3 Discrete Dirichlet Energy and Laplacian Eigenmaps

The graph Laplacian has a discrete Dirichlet energy interpretation directly parallel to the manifold case. For a function $\boldsymbol{g} : V \to \mathbb{R}$, identified with a vector $\boldsymbol{g} = (g_1, \ldots, g_N)^\top \in \mathbb{R}^N$, one has

$$\boldsymbol{g}^\top \boldsymbol{L}\boldsymbol{g} = \frac{1}{2} \sum_{i,j=1}^{N} W_{ij}(g_i - g_j)^2. \tag{2.21}$$

This expression is small when strongly connected neighbouring points have similar function values. Therefore, minimising the graph Laplacian energy produces functions that vary smoothly over the data graph. This is precisely the principle behind Laplacian Eigenmaps [9]. In that method, one solves the generalised eigenvalue problem

$$\boldsymbol{L}\boldsymbol{y} = \lambda \boldsymbol{D}\boldsymbol{y}, \tag{2.22}$$

and uses the nontrivial low-frequency eigenvectors as embedding coordinates. The objective being minimised is

$$\sum_{i,j=1}^{N} W_{ij}\|y_i - y_j\|^2, \tag{2.23}$$

subject to normalisation constraints that avoid the trivial collapsed solution. The resulting embedding preserves local neighbourhoods because nearby points in the original space are penalised if they are mapped far apart.

2.3.4 Continuum Limits of Graph Laplacians

The finite graph construction is useful because it can be connected back to the continuous geometric object. This section states the continuum-limit viewpoint, clarifies the role of graph construction scales, and explains when graph Laplacians can be treated as approximations to manifold operators.

The connection between graph Laplacians and Laplace-Beltrami operators is not only heuristic. Under suitable assumptions, graph Laplacians constructed from random samples on a smooth manifold converge to differential operators on the underlying manifold as the number of samples grows and the kernel bandwidth is appropriately decreased [10, 11, 12]. Informally, if $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N$ are sampled from an m-dimensional manifold $\mathcal{N}$ and ε satisfies

$$\varepsilon \to 0, \qquad N\varepsilon^m \to \infty, \tag{2.24}$$

or in case we use σ, it satisfies

$$\sigma \to 0, \qquad N\sigma^{m/2} \to \infty, \tag{2.25}$$

then local kernel averages over the graph approximate local geometric averages on the manifold. In this regime, a suitably rescaled graph Laplacian applied to a sampled smooth function approximates the Laplace-Beltrami operator:

$$c_\varepsilon(\boldsymbol{I} - \boldsymbol{P}_\varepsilon)f \approx \mathcal{L}_{\mathcal{M}} f, \tag{2.26}$$

up to constants, sampling-density effects, and higher-order error terms. Here, $\boldsymbol{P}_\varepsilon$ is calculated from $\boldsymbol{D}_\varepsilon$ and $\boldsymbol{W}_\varepsilon$ (graph constructed at scale ε using Equation 2.12) using Equation 2.18, and c_ε is a scale-dependent normalisation constant, typically of order $1/\varepsilon$, chosen so that the discrete graph operator has a nontrivial continuum limit as $\varepsilon \to 0$. Without correcting for nonuniform sampling density, the limit generally contains density-dependent drift terms. With appropriate density normalisation, the limiting operator can be made to recover the intrinsic Laplace-Beltrami operator [12].

This convergence result gives a precise sense in which a graph built from finite samples can serve as a discrete proxy for the geometry of an unknown manifold. It also clarifies the role of the graph construction parameters. The distance cutoff radius ε, the bandwidth σ, or the number of nearest neighbours k must be small enough to capture local geometry, but large enough to average out sampling noise and keep the graph connected. If the scale is too small, the graph may fragment, and the operator becomes unstable. If it is too large, the graph no longer reflects local manifold structure and the approximation becomes biased by global Euclidean distances. Thus, the graph Laplacian is not merely a generic similarity matrix but is also a finite-sample approximation to a differential operator, but only when the graph construction respects the relevant local geometry.

2.4 Caveats and Scope of Manifold and Graph-Based Geometry

The preceding sections use several idealised mathematical objects to describe finite neural representations. The following caveats summarise the main limitations of the manifold and graph viewpoints.

First, the manifold hypothesis is an idealisation. Real data may concentrate near a stratified set, a union of manifolds, or a noisy manifold rather than exactly on one smooth manifold. Second, finite-sample graph Laplacians are affected by sample size, kernel bandwidth, graph connectivity, anisotropic sampling density, and ambient noise. These effects must be considered when interpreting graph spectra or operator-based distances between representations.

Despite these limitations, the manifold and Laplacian viewpoint provides a useful mathematical bridge between representation geometry and finite computations.

2.5 Diffusion Geometry

The continuum-limit perspective connects graph Laplacians to differential operators on manifolds. Diffusion geometry expresses the same connection through random walks: the matrix $\boldsymbol{P}$ describes local transitions on the sampled point cloud, while its powers describe how information propagates over increasingly larger regions of

the underlying geometry. In this sense, diffusion geometry provides the probabilistic counterpart of the graph-Laplacian viewpoint introduced above.

2.5.1 Markov Operators and Diffusion Powers

The graph construction above gives the matrix $\boldsymbol{P}$ defined using Equation 2.18. It is also called a Markov matrix, or a Markov transition matrix, as it can be used to describe the transitions of a Markov chain. As also noted earlier, the random-walk-normalised graph Laplacian $\boldsymbol{L}_{\text{rw}} = \boldsymbol{I} - \boldsymbol{P}$ is the discrete analogue of a diffusion generator. Thus, the graph Laplacian and the Markov transition matrix are two closely related ways of encoding the same finite-sample geometry: one emphasises variation and smoothness through $\boldsymbol{L}_{\text{rw}}$, while the other emphasises propagation and random-walk dynamics through $\boldsymbol{P}$.

The entry P_{ij} is the probability of moving from sample i to sample j in one step of the random walk. The t-th power of this matrix, $\boldsymbol{P}^t$ has entries $(\boldsymbol{P}^t)_{ij}$ equal to the probability of moving from sample i to sample j in t steps. Thus, while $\boldsymbol{P}$ describes one-step local transitions, $\boldsymbol{P}^t$ describes connectivity through paths of length t.

This is the basic multiscale mechanism of diffusion geometry. For small t, the operator $\boldsymbol{P}^t$ emphasises local neighbourhoods. Two samples are close if there is a strong direct connection between them. For larger t, the random walk has had time to move through intermediate samples, and the resulting transition probabilities depend on larger-scale connectivity patterns. Two points may then be considered close not only because they are direct neighbours, but also because they have similar transition behaviour through the rest of the dataset. In this way, diffusion geometry moves from local pairwise similarity to global manifold structure by repeatedly applying the same Markov operator [12].

2.5.2 Diffusion Distances

This multiscale mechanism can be expressed through diffusion distances. Let $\boldsymbol{e}_i \in \mathbb{R}^N$ denote the i-th standard basis vector. The row vector

$$\boldsymbol{p}_i^{(t)} = \boldsymbol{e}_i^\top \boldsymbol{P}^t \tag{2.27}$$

is the distribution of a t-step random walk started at sample i. A diffusion distance compares two samples by comparing these transition distributions:

$$D_t^2(i,j) = \sum_{k=1}^{N} \frac{\left((\boldsymbol{P}^t)_{ik} - (\boldsymbol{P}^t)_{jk}\right)^2}{\pi_k}, \tag{2.28}$$

where $\boldsymbol{\pi}$ is the stationary distribution of the Markov chain [12, 13]. The denominator weights the coordinates according to the stationary density of the random walk. Intuitively, $D_t(i,j)$ is small when random walks started at i and j spread through the dataset in similar ways. Therefore, the diffusion distance compares points by their connectivity profiles rather than by direct Euclidean distance alone.

2.5.3 Spectral Diffusion Maps

When the Markov operator is reversible, this diffusion geometry can also be represented spectrally. Suppose that $\boldsymbol{P}$ has eigenvalues

$$1 = \lambda_0 \geq |\lambda_1| \geq |\lambda_2| \geq \cdots \tag{2.29}$$

with corresponding eigenvectors $\psi_0, \psi_1, \psi_2, \ldots$. The diffusion map at time t embeds each sample i into coordinates of the form

$$\Psi_t(i) = \left(\lambda_1^t \psi_1(i), \lambda_2^t \psi_2(i), \ldots, \lambda_q^t \psi_q(i)\right), \tag{2.30}$$

for some truncation dimension q. Here $\psi_k(i)$ is the i-th component of the eigenvector ψ_k. $\boldsymbol{\Psi}_t \in \mathbb{R}^{N \times q}$ denotes the matrix whose rows are the diffusion-map coordinates, and $\Psi_t(i)$ is the i-th row of $\boldsymbol{\Psi}_t$. The powers λ_k^t suppress directions with small eigenvalues as t grows. Hence, increasing t progressively removes high-frequency or fine-scale components and leaves only the dominant large-scale geometric modes. This is the same conceptual role played by heat diffusion on a continuous manifold [12].

2.5.4 Implications for Neural Representation Analysis

For neural network representations, this perspective suggests a way to study layers geometrically. At each layer ℓ, the representation point cloud $\mathcal{R}^{(\ell)}$ induces a weighted graph, a random-walk matrix $\boldsymbol{P}^{(\ell)}$, and a graph Laplacian $\boldsymbol{L}^{(\ell)}$. These objects can be interpreted as finite approximations to diffusion or Laplacian operators on the corresponding representation manifold $\mathcal{M}_\ell$. Changes in representation geometry can therefore be studied through changes in these operators. For example, the spectrum of $\boldsymbol{L}^{(\ell)}$ may reflect the number and organisation of large-scale modes of variation in the representation. The eigenvectors may reveal smooth semantic directions or cluster structure. The heat kernel or random-walk operator may describe how local neighbourhoods propagate across the representation space.

This operator-based viewpoint differs from comparing representations only through their raw coordinates. Coordinate representations are sensitive to choices of basis in the ambient space. In contrast, distance-based relational objects, such as pairwise distance matrices, graph weights constructed from Euclidean distances, and the corresponding transition matrices or graph Laplacians, are invariant to translations and orthogonal transformations of the representation vectors. More importantly, they emphasise the relational geometry of the data: which samples are close, how local neighbourhoods connect, and how functions diffuse over the representation. This is particularly useful when comparing representations across layers or across different models, where the ambient dimensions and coordinate systems may differ. Even if two layers live in different Euclidean spaces, their sampled manifold geometries can still be compared through graph-based or operator-based objects.

2.6 Representations

The geometric constructions above describe representation point clouds through manifolds, graph Laplacians, and Markov diffusion operators. We now turn to the

sample-indexed relational objects used to compare such representations across layers and networks. The central objects are representational similarity matrices and scalar similarity measures derived from them.

Let

$$\Phi_L = \phi^{(L)} \circ \phi^{(L-1)} \circ \cdots \circ \phi^{(1)} \tag{2.31}$$

be a neural network with L layers, and let

$$\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \tag{2.32}$$

be a fixed set of input samples. The representation of the i-th sample at layer ℓ is denoted by

$$\boldsymbol{r}_i^{(\ell)} = \Phi_\ell(\boldsymbol{x}_i) \in \mathbb{R}^{d_\ell}, \tag{2.33}$$

where $\Phi_\ell = \phi^{(\ell)} \circ \cdots \circ \phi^{(1)}$ is the layer-ℓ representation map. It is often convenient to collect these as rows in a representation matrix

$$\boldsymbol{R}^{(\ell)} = \begin{pmatrix} (\boldsymbol{r}_1^{(\ell)})^\top \\ \vdots \\ (\boldsymbol{r}_N^{(\ell)})^\top \end{pmatrix} \in \mathbb{R}^{N \times d_\ell}. \tag{2.34}$$

The row $\boldsymbol{R}_{i,:}^{(\ell)}$ is therefore the representation of the input $\boldsymbol{x}_i$ at layer ℓ. This matrix representation is simply another way of writing the finite representation point cloud $\mathcal{R}^{(\ell)} = \{\boldsymbol{r}_1^{(\ell)}, \ldots, \boldsymbol{r}_N^{(\ell)}\}$ introduced earlier in this chapter.

2.6.1 Representational Similarity Measures

Similarity measures of neural representations have become a central tool for analysing and comparing trained networks [14, 15, 16], as well as for network optimisation, for instance in knowledge distillation [17]. A representational similarity measure, or simply a similarity measure, is a function that assigns a scalar score to a pair of representations. If

$$\boldsymbol{R}_1 \in \mathbb{R}^{N \times d_1}, \qquad \boldsymbol{R}_2 \in \mathbb{R}^{N \times d_2}, \tag{2.35}$$

are two representations of the same N samples, then a similarity measure is a map

$$m : \mathbb{R}^{N \times d_1} \times \mathbb{R}^{N \times d_2} \to \mathbb{R}, \tag{2.36}$$

where $m(\boldsymbol{R}_1, \boldsymbol{R}_2)$ is intended to quantify how similar the two representations are. The two representations may come from different layers of the same network, or even layers of different networks. Since d_1 and d_2 may differ, it is useful to compare representations through objects that do not require a direct coordinate-wise alignment of their ambient spaces.

2.6.2 Representational Similarity Matrices (RSMs)

A standard way to remove the dependence on the ambient representation dimension is to pass from representations to representational similarity matrices (RSMs) [14, 16]. In this thesis, the abbreviation RSM is used only for representational similarity matrix, not for representational similarity measure (from the previous subsection). Given a representation matrix $\boldsymbol{R} \in \mathbb{R}^{N \times d}$ and an instance-wise similarity function

$$s : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}, \tag{2.37}$$

the corresponding RSM is the matrix

$$\boldsymbol{S} \in \mathbb{R}^{N \times N}, \qquad S_{ij} = s(\boldsymbol{R}_{i,:}, \boldsymbol{R}_{j,:}). \tag{2.38}$$

Thus, $\boldsymbol{S}$ records the pairwise similarity structure among the samples in a single representation. The RSM does not store the coordinates of the samples directly. Instead, it stores how the samples relate to one another according to the chosen similarity function s.

Common choices of s include the linear kernel

$$s_{\text{lin}}(\boldsymbol{r}_i, \boldsymbol{r}_j) = \boldsymbol{r}_i^\top \boldsymbol{r}_j, \tag{2.39}$$

and the Gaussian or RBF kernel

$$s_{\text{RBF}}(\boldsymbol{r}_i, \boldsymbol{r}_j) = \exp\left(-\frac{\|\boldsymbol{r}_i - \boldsymbol{r}_j\|^2}{2\sigma^2}\right), \tag{2.40}$$

where $\sigma > 0$ is the bandwidth [18, 19]. Other choices, such as Pearson correlation or negative Euclidean distance, are also possible depending on whether one wants the RSM to encode angular similarity, kernel similarity, or relationships based on distances. Crucially, after constructing $\boldsymbol{S}$, the representation is described by the geometry of pairwise sample relationships rather than by the original coordinates of the activation vectors.

This abstraction is useful for comparing neural representations because different layers or networks may have different widths, different bases, or different coordinate systems. Two representation matrices $\boldsymbol{R}_1 \in \mathbb{R}^{N \times d_1}$ and $\boldsymbol{R}_2 \in \mathbb{R}^{N \times d_2}$ cannot always be compared directly in coordinate space, especially when $d_1 \neq d_2$. Their RSMs $\boldsymbol{S}_1, \boldsymbol{S}_2 \in \mathbb{R}^{N \times N}$, however, always live in the same sample-indexed space, provided that both representations are computed on the same N inputs. RSM-based comparison therefore shifts the problem from comparing activation coordinates to comparing sample geometries.

2.6.3 RSM-Based Similarity Measures

Formally, an RSM-based similarity measure is a measure of the form

$$m(\boldsymbol{R}_1, \boldsymbol{R}_2) = m_{\text{RSM}}(\boldsymbol{S}_1, \boldsymbol{S}_2), \tag{2.41}$$

where $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ are the RSMs induced by $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$ [16]. Such a measure ignores any aspect of the representations that is not reflected in the chosen pairwise similarity function [15]. This is both a strength and a limitation. It is a strength because it makes the measure less dependent on arbitrary coordinates. It is a limitation because the choice of s determines which geometric information is retained. For example, a linear kernel emphasises inner products, while an RBF kernel emphasises local Euclidean neighbourhoods at the scale set by σ.

2.7 Centered Alignment Measures

Many RSM-based comparison methods first center the sample-indexed relational matrices and then compare the centered objects. This section describes the centering operation and then presents CKA and DistCorr as two standard examples of normalised alignment measures.

2.7.1 Centering Matrices

Let

$$\boldsymbol{H} = \boldsymbol{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^\top, \tag{2.42}$$

where $\boldsymbol{I} \in \mathbb{R}^{N\times N}$ is the identity matrix and $\mathbf{1} \in \mathbb{R}^N$ is the vector whose entries are all equal to one. The matrix $\boldsymbol{H}$ is the centering matrix. Left multiplication by $\boldsymbol{H}$ centers the columns of a matrix, which means we subtract the mean row vector from every row, so that each column has a mean of zero. Similarly, right multiplication by $\boldsymbol{H}$ centers its rows. Thus

$$\widetilde{\boldsymbol{S}} = \boldsymbol{H}\boldsymbol{S}\boldsymbol{H} \tag{2.43}$$

is the double-centered version of $\boldsymbol{S}$. centering removes global mean effects and ensures that the comparison focuses on fluctuations of pairwise similarities rather than on the overall average similarity level [20, 15].

2.7.2 Centered Kernel Alignment (CKA)

An example is Centered Kernel Alignment (CKA) [15]. Given two kernel RSMs $\boldsymbol{S}_1, \boldsymbol{S}_2 \in \mathbb{R}^{N\times N}$, the empirical estimator of the Hilbert-Schmidt Independence Criterion (HSIC) can be written as

$$\mathrm{HSIC}(\boldsymbol{S}_1, \boldsymbol{S}_2) = \frac{1}{(N-1)^2}\,\mathrm{tr}(\boldsymbol{S}_1\boldsymbol{H}\boldsymbol{S}_2\boldsymbol{H}). \tag{2.44}$$

CKA normalises this quantity by the self-alignment of each representation:

$$m_{\mathrm{CKA}}(\boldsymbol{S}_1, \boldsymbol{S}_2) = \frac{\mathrm{HSIC}(\boldsymbol{S}_1, \boldsymbol{S}_2)}{\sqrt{\mathrm{HSIC}(\boldsymbol{S}_1, \boldsymbol{S}_1)\,\mathrm{HSIC}(\boldsymbol{S}_2, \boldsymbol{S}_2)}}. \tag{2.45}$$

Equivalently, ignoring the common normalisation constant in HSIC, CKA can be viewed as the cosine similarity between the centered RSMs:

$$m_{\mathrm{CKA}}(\boldsymbol{S}_1, \boldsymbol{S}_2) = \frac{\langle \widetilde{\boldsymbol{S}_1}, \widetilde{\boldsymbol{S}_2} \rangle_F}{\|\widetilde{\boldsymbol{S}_1}\|_F \|\widetilde{\boldsymbol{S}_2}\|_F}, \tag{2.46}$$

where

$$\langle \boldsymbol{A}, \boldsymbol{B} \rangle_F = \mathrm{tr}(\boldsymbol{A}^\top \boldsymbol{B}) \tag{2.47}$$

is the Frobenius inner product and $\|\boldsymbol{A}\|_F = \sqrt{\langle \boldsymbol{A}, \boldsymbol{A} \rangle_F}$ is the Frobenius norm, and using Equation 2.43. This form clarifies that CKA compares the centered pairwise similarity structures of the two representations [21, 20, 15].

2.7.3 Distance Correlation (DistCorr)

Another important RSM-based measure is Distance Correlation (DistCorr) [22]. One first constructs pairwise distance or dissimilarity matrices for both representations:

$$(D_1)_{ij} = \|\boldsymbol{R}_{1,i,:} - \boldsymbol{R}_{1,j,:}\|, \qquad (D_2)_{ij} = \|\boldsymbol{R}_{2,i,:} - \boldsymbol{R}_{2,j,:}\|. \tag{2.48}$$

These matrices are then double-centered using Equation 2.43:

$$\widetilde{\boldsymbol{D}_1} = \boldsymbol{H}\boldsymbol{D}_1\boldsymbol{H}, \qquad \widetilde{\boldsymbol{D}_2} = \boldsymbol{H}\boldsymbol{D}_2\boldsymbol{H}. \tag{2.49}$$

The squared sample distance covariance is proportional to the Frobenius inner product of the centered distance matrices:

$$\mathrm{dCov}^2(\boldsymbol{D}_1, \boldsymbol{D}_2) = \frac{1}{N^2} \langle \widetilde{\boldsymbol{D}_1}, \widetilde{\boldsymbol{D}_2} \rangle_F. \tag{2.50}$$

The normalised distance correlation score is then

$$m_{\mathrm{DistCorr}}(\boldsymbol{D}_1, \boldsymbol{D}_2) = \frac{\langle \widetilde{\boldsymbol{D}_1}, \widetilde{\boldsymbol{D}_2} \rangle_F}{\|\widetilde{\boldsymbol{D}_1}\|_F \|\widetilde{\boldsymbol{D}_2}\|_F}. \tag{2.51}$$

Thus, like CKA, DistCorr can be understood as a normalised alignment between centered pairwise relational matrices [22]. The difference lies primarily in the type of pairwise relation being compared: CKA is naturally formulated through kernels, whereas DistCorr is naturally formulated through distances. Thus, in what follows, m_{RSM} should be understood more broadly as acting on sample-indexed relational matrices. These may encode similarities, as in kernel RSMs, or distances/dissimilarities, as in DistCorr, as long as the corresponding centering and normalisation operations are well defined.

2.8 Caveats and Scope of Diffusion Operators and RSMs

The following caveats summarise the main limitations of the diffusion-operator and RSM viewpoints. First, an RSM is not a complete description of a representation.

It retains only the pairwise relations specified by the similarity function s. Different choices of s can therefore lead to different conclusions. Second, centering can remove useful global information if the global mean similarity is itself meaningful for a particular question. Third, diffusion operators depend on bandwidths, sparsification choices, and normalisation conventions, just as graph Laplacians do. These choices affect the geometry being compared.

Another practical issue is excessive mixing. For an ergodic Markov chain, powers of the transition matrix eventually converge toward a rank-one stationary operator [13]. Similarly, products of many Markov matrices may wash out the information that distinguishes samples. Thus, while diffusion powers and products of Markov matrices are useful for capturing coarse geometry, they should not be pushed so far that all sample-specific structure disappears. The useful regime is typically intermediate: enough diffusion or alternation to suppress noise and view-specific artefacts, but not so much that the operator has mixed completely.

3

Applications: Knowledge Distillation and Multi-View Learning

The preceding chapter introduced representation manifolds, graph-based operators, representational similarity matrices, and RSM-based similarity measures as tools for comparing neural representations. This chapter collects application-oriented background material. It first describes knowledge distillation as an application of representation alignment, and then turns to multi-view learning and alternating diffusion.

3.1 Knowledge Distillation

Knowledge distillation (KD) refers to a broad family of methods in which a student model is trained using information supplied by a teacher model. The teacher is usually larger, more accurate, or otherwise better informed, while the student is usually smaller, faster, or more suitable for deployment. In its classical form, KD trains the student to match the softened output distribution of the teacher, so that the student learns not only the hard class labels but also the relative probabilities assigned by the teacher to different classes [23, 24].

Although KD was initially developed as a model-compression technique, it has since become a broad research area with many variants. Depending on the setting, the transferred information may include output logits, class probabilities, intermediate features, attention maps, pairwise relations between samples, or higher-order structure in the teacher's representation space [24]. In the context of this thesis, the relevant observation is that certain forms of KD go beyond output matching and can be interpreted as representation-alignment objectives between teacher and student networks. The teacher provides a fixed representation, and the student is trained so that one or more of its representations become similar to the corresponding teacher representations.

3.1.1 Teacher-Student Formulation

Let T denote a fixed teacher network and let S_θ denote a student network with trainable parameters θ. For a mini-batch of inputs

$$\mathcal{X} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\}, \tag{3.1}$$

let

$$\boldsymbol{R}_T^{(a)} \in \mathbb{R}^{N \times d_T^{(a)}}, \qquad \boldsymbol{R}_{S_\theta}^{(b)} \in \mathbb{R}^{N \times d_S^{(b)}} \tag{3.2}$$

denote the representation matrices obtained from layer a of the teacher and layer b of the student, respectively. The teacher representation is fixed during student training, whereas the student representation depends on θ.

A generic representation-level distillation loss can therefore be written as

$$\mathcal{L}_{\text{rep}}(\theta) = d\left(\boldsymbol{R}_T^{(a)}, \boldsymbol{R}_{S_\theta}^{(b)}\right), \tag{3.3}$$

where d is a discrepancy between representations. The full training objective may combine this representation-level term with the usual supervised task loss and, possibly, with an output-level KD loss:

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{task}}(\theta) + \alpha \mathcal{L}_{\text{KD}}(\theta) + \lambda \mathcal{L}_{\text{rep}}(\theta). \tag{3.4}$$

Here α and λ control the contribution of the distillation losses. In this form, KD can be viewed as an optimisation problem in which the student is encouraged to match some teacher-defined target, either at the output level or at the representation level, or both.

3.1.2 Feature-Level Distillation and Layer Matching

Feature-level distillation methods make the intermediate representations of the teacher part of the training signal. Early representation-based KD methods introduced this idea by using intermediate teacher representations as additional training targets for the student. Since the hidden dimensions of the teacher and student layers may differ, these methods typically include an additional projection from the student representation space to the teacher representation space, and penalise the discrepancy between the projected student feature and the corresponding teacher feature [25].

In the notation used here, a single layer-matching objective has the general form

$$\mathcal{L}_{a,b}(\theta) = d\left(h_\theta\left(\boldsymbol{R}_{S_\theta}^{(b)}\right), \boldsymbol{R}_T^{(a)}\right), \tag{3.5}$$

where h_θ is a learned projection or adaptation map. This type of objective compares teacher and student features directly. It is therefore a representation-alignment method, but not necessarily an RSM-based one: the loss is applied to the feature coordinates after some form of dimensional alignment.

This layer-matching view is useful for connecting feature-level KD to the similarity-measure perspective. A teacher layer and a student layer define two representations of the same mini-batch. The distillation term then acts as a training-time penalty that encourages these two representations to become close according to a chosen discrepancy. In standard feature matching, this discrepancy may be a Euclidean or mean-squared error loss. In RSM-based methods, the discrepancy is instead applied after converting the two representations into sample-indexed relational matrices.

3.1.3 RSM-Based Distillation Objectives

The RSM perspective gives a way to compare teacher and student representations even when their hidden dimensions differ. Given the teacher and student representation matrices $\boldsymbol{R}_T^{(a)}$ and $\boldsymbol{R}_{S_\theta}^{(b)}$ one may construct corresponding relational matrices according to Equation 2.38:

$$\boldsymbol{S}_T^{(a)} \in \mathbb{R}^{N\times N}, \qquad (S_T^{(a)})_{i,j} = s_T((\boldsymbol{R}_T^{(a)})_{i,:}, \boldsymbol{R}_T^{(a)})_{j,:}), \tag{3.6}$$

$$\boldsymbol{S}_{S_\theta}^{(b)} \in \mathbb{R}^{N\times N}, \qquad (S_{S_\theta}^{(b)})_{i,j} = s_S((\boldsymbol{R}_{S_\theta}^{(b)})_{i,:}, \boldsymbol{R}_{S_\theta}^{(b)})_{j,:}), \tag{3.7}$$

where s_T and s_S are the chosen pairwise similarity functions. An RSM-based distillation loss can then be written abstractly as

$$\mathcal{L}_{\mathrm{RSM}}(\theta) = d_{\mathrm{RSM}}\left(\boldsymbol{S}_T^{(a)}, \boldsymbol{S}_{S_\theta}^{(b)}\right), \tag{3.8}$$

or, equivalently, as the negative of a similarity score,

$$\mathcal{L}_{\mathrm{RSM}}(\theta) = 1 - m_{\mathrm{RSM}}\left(\boldsymbol{S}_T^{(a)}, \boldsymbol{S}_{S_\theta}^{(b)}\right), \tag{3.9}$$

when the similarity score is normalised so that larger values indicate better alignment.

This formulation makes the connection with the previous chapter direct. The teacher and student do not need to have the same number of hidden units, because both RSMs live in the same $N \times N$ sample-indexed space. What is transferred is not the coordinate representation itself, but the teacher's relational geometry over the mini-batch. In this sense, RSM-based KD uses a representation similarity measure as a training objective rather than only as a post-hoc analysis tool.

3.1.4 Similarity-Preserving Knowledge Distillation

Similarity-Preserving Knowledge Distillation (SPKD) is a direct example of this idea [17]. Instead of requiring the student to reproduce the teacher's feature coordinates, SPKD encourages the student to preserve the teacher's pairwise activation similarities. The motivating principle is that if two inputs produce similar activation patterns in the teacher, then they should also produce similar activation patterns in the student.

For a teacher layer and a student layer, one can form Gram-like matrices from the corresponding batch representations,

$$\boldsymbol{G}_T = \boldsymbol{R}_T^{(a)}(\boldsymbol{R}_T^{(a)})^\top, \qquad \boldsymbol{G}_S = \boldsymbol{R}_{S_\theta}^{(b)}(\boldsymbol{R}_{S_\theta}^{(b)})^\top, \tag{3.10}$$

where the corresponding pairwise similarity functions from Equation 3.7 are the linear kernels

$$s_T((\boldsymbol{R}_T^{(a)})_{i,:}, (\boldsymbol{R}_T^{(a)})_{j,:}) = (\boldsymbol{R}_T^{(a)})_{i,:}^\top (\boldsymbol{R}_T^{(a)})_{j,:}, \tag{3.11}$$

$$s_S((\boldsymbol{R}_{S_\theta}^{(b)})_{i,:}, (\boldsymbol{R}_{S_\theta}^{(b)})_{j,:}) = (\boldsymbol{R}_{S_\theta}^{(b)})_{i,:}^\top (\boldsymbol{R}_{S_\theta}^{(b)})_{j,:}. \tag{3.12}$$

After normalisation, SPKD penalises the discrepancy between the teacher and student similarity matrices. In schematic form, this can be written as

$$\mathcal{L}_{\mathrm{SPKD}}(\theta) = \left\| \widehat{\boldsymbol{G}}_T - \widehat{\boldsymbol{G}}_S \right\|_F^2, \tag{3.13}$$

where $\widehat{\boldsymbol{G}}_T$ and $\widehat{\boldsymbol{G}}_S$ denote the normalised teacher and student similarity matrices, and d_{RSM} from Equation 3.8 is the Frobenius norm of the difference between those matrices. This is precisely an RSM-based distillation objective: the teacher provides a fixed relational matrix, and the student is trained so that its own relational matrix matches it.

3.1.5 Relational Knowledge Distillation

Relational Knowledge Distillation (RKD) extends this idea by explicitly transferring relations between data examples rather than individual teacher activations [26]. RKD proposes distance-wise and angle-wise losses. The distance-wise term compares normalised pairwise distances between samples in the teacher and student representation spaces, while the angle-wise term compares angles formed by triples of samples.

The distance-wise part is closest to the RSM-based viewpoint. If

$$(D_T)_{ij} = \left\| \boldsymbol{r}_{T,i}^{(a)} - \boldsymbol{r}_{T,j}^{(a)} \right\|, \qquad (D_S)_{ij} = \left\| \boldsymbol{r}_{S,i}^{(b)} - \boldsymbol{r}_{S,j}^{(b)} \right\|, \tag{3.14}$$

then RKD transfers the teacher's distance structure to the student by penalising discrepancies between such relational quantities. This is analogous to using a distance-based relational matrix rather than a kernel-based RSM. The angle-wise term goes beyond pairwise distances by incorporating higher-order relations among triples of samples, but the same general principle remains: the student is trained to match structural relations induced by the teacher representation.

From the perspective developed in the preceding chapter, RKD can therefore be interpreted as a relational distillation method. It does not merely align teacher and student outputs sample by sample. Instead, it uses the geometry of the teacher's representation space, as expressed through distances and angles among samples, as the object to be transferred.

3.1.6 Correlation and Kernel-Based Distillation

Correlation Congruence for Knowledge Distillation (CCKD) also emphasises relations between multiple instances [27]. Its motivation is that instance-level congruence alone may ignore correlations among samples that are useful for knowledge transfer. CCKD therefore transfers both instance-level information and correlation information, using a kernel-based construction to capture correlations within a mini-batch.

This again connects KD to the RSM and kernel perspectives. Kernel matrices and RSMs are both sample-indexed relational matrices. A kernel-based distillation loss can therefore be viewed as encouraging the student to match not only individual teacher outputs or features, but also the teacher-induced pattern of sample-sample relations. Such objectives sit between standard feature matching and explicit RSM alignment: they use the algebra of kernels to encode relational structure and then use this structure as part of the student training objective.

More recent work has also studied CKA directly as a distillation objective. Since CKA compares centered kernel matrices and is invariant to isotropic scaling and orthogonal transformations of the representations, it is naturally suited to teacher-student settings where hidden dimensions or coordinate systems differ. Recent CKA-based KD work explicitly uses CKA to reduce representation discrepancy between teacher and student models and analyses how such CKA losses function in distillation [28]. This provides another direct link between representational similarity measures and KD: the same measure used to compare representations after training can also be inserted into the training loss.

3.1.7 Multiple Layer Pairs

The preceding examples can be applied to a single teacher-student layer pair, but KD methods may also use several intermediate layers. Suppose that a set of layer correspondences is chosen,

$$\mathcal{P} = \{(a_1, b_1), \ldots, (a_M, b_M)\}, \tag{3.15}$$

where a_m indexes a teacher layer and b_m indexes the corresponding student layer. A multi-layer representation distillation objective can then be written as

$$\mathcal{L}_{\text{multi}}(\theta) = \sum_{m=1}^{M} \omega_m d_m \left(\boldsymbol{R}_T^{(a_m)}, \boldsymbol{R}_{S_\theta}^{(b_m)} \right), \tag{3.16}$$

or, in an RSM-based form,

$$\mathcal{L}_{\text{multi-RSM}}(\theta) = \sum_{m=1}^{M} \omega_m d_{\text{RSM},m} \left(\boldsymbol{S}_T^{(a_m)}, \boldsymbol{S}_{S_\theta}^{(b_m)} \right). \tag{3.17}$$

Here ω_m controls the contribution of each matched layer pair. This formulation captures the common situation in which several teacher layers are selected, each

is paired with a student layer, and the resulting losses are summed and optimised jointly.

This multi-layer form is still a layer-pairing objective rather than a full network-level comparison. Each term compares one teacher layer with one student layer, and the final objective aggregates these comparisons.

3.1.8 Relation to the Similarity-Measure Viewpoint

The RSM perspective of KD discussed in the previous sections does not reduce all of KD to similarity-measure optimisation. KD includes output-level distillation, feature matching, attention transfer, data-free distillation, self-distillation, and many other variants [24]. However, it does identify a clear class of KD methods in which the training signal is a representation similarity or relational-structure objective. SPKD, RKD, CCKD, and CKA-based distillation are representative examples. They differ in the relational object they transfer, but they share the same basic form: a teacher network defines a geometry over samples, and the student is trained to reproduce that geometry as measured by an appropriate similarity or discrepancy.

3.2 Multi-View Learning

Another important idea for this thesis is multi-view learning [29, 30]. Multi-view learning studies settings in which the same underlying phenomenon is observed through several different views. These views may correspond to several modalities, sensors, or feature extractors. For example, the same object may be described by an image, a text description, and metadata, the same patient may be described by several medical measurements, or different feature maps may represent the same data points.

3.2.1 Multi-View Problem Formulation

The central assumption is that the views are aligned at the sample level. If there are N underlying samples, then the index i refers to the same sample across all views. Thus, one may write M views as matrices

$$\boldsymbol{V}^{(a)} = \begin{pmatrix} (\boldsymbol{v}_1^{(a)})^\top \\ \vdots \\ (\boldsymbol{v}_N^{(a)})^\top \end{pmatrix} \in \mathbb{R}^{N \times d_a}, \qquad a = 1, \dots, M. \tag{3.18}$$

Here, a indexes the view, d_a is the ambient dimension of the a-th view, and $\boldsymbol{v}_i^{(a)}$ is the representation of the i-th sample in that view. The dimensions d_a may differ across views, but the sample index set is shared.

Multi-view learning is based on two related principles. The first is a *consensus* principle: different views of the same sample should contain some common information, and a useful multi-view representation should preserve this shared structure.

The second is a *complementarity* principle: different views may also contain information that is not present in any single view alone, so combining views can be more informative than using one view in isolation [29, 30]. Different multi-view methods emphasise these principles in different ways. For instance, co-training methods exploit agreement between views, multiple-kernel learning combines view-specific kernels, and subspace or representation-learning methods aim to recover a latent representation shared across views.

A useful abstract model is that each view contains both common and view-specific information. One may write this informally as

$$\boldsymbol{v}_i^{(a)} = g_a(\boldsymbol{u}_i, \boldsymbol{\eta}_i^{(a)}), \tag{3.19}$$

where $\boldsymbol{u}_i$ denotes the shared latent content of the i-th sample and $\boldsymbol{\eta}_i^{(a)}$ denotes variation specific to the a-th view. The observation map g_a may be nonlinear and may differ from view to view. The goal of multi-view learning is not always to recover the coordinates of $\boldsymbol{u}_i$ explicitly. More generally, the aim is to combine the views in a way that preserves information common to them, while reducing the influence of view-specific nuisance variation.

This distinction between shared and view-specific information is important. If two samples appear close in only one view, this may reflect genuine shared structure, but it may also reflect a nuisance factor specific to that view. Conversely, if two samples are consistently close across several views, this provides stronger evidence that their relation belongs to the common structure. Multi-view methods differ in how they formalise and use this idea.

3.2.2 Markov Matrices from Individual Views

The diffusion-geometric approach to multi-view learning first associates each view with its own Markov operator. Given one view $\boldsymbol{V}^{(a)} \in \mathbb{R}^{N \times d_a}$, one constructs an affinity matrix

$$\boldsymbol{W}^{(a)} \in \mathbb{R}^{N \times N}, \tag{3.20}$$

for example, using a local Gaussian kernel

$$W_{ij}^{(a)} = \exp\left(-\frac{|\boldsymbol{v}_i^{(a)} - \boldsymbol{v}_j^{(a)}|^2}{\varepsilon_a}\right), \tag{3.21}$$

or another similarity function appropriate for that view. The corresponding degree matrix is

$$D_{ii}^{(a)} = \sum_{j=1}^{N} W_{ij}^{(a)}, \tag{3.22}$$

and the row-normalised Markov matrix is

$$\boldsymbol{P}^{(a)} = (\boldsymbol{D}^{(a)})^{-1} \boldsymbol{W}^{(a)}. \tag{3.23}$$

The entry $P_{ij}^{(a)}$ is the probability of moving from sample i to sample j in one step of the random walk defined by the a-th view. Thus, each view induces its own diffusion geometry on the same set of samples.

This construction separates the sample alignment from the geometry of each view. The rows and columns of all $\boldsymbol{P}^{(a)}$ are indexed by the same samples, so the Markov matrices can be composed or compared. However, the transition probabilities inside each $\boldsymbol{P}^{(a)}$ are determined only by the geometry of the corresponding view. This is what makes diffusion-based multi-view methods possible: each view contributes a Markov operator, and the multi-view method specifies how these operators are combined.

3.2.3 Alternating Diffusion

Multi-view learning includes a broad range of approaches, such as co-training, multiple-kernel methods, subspace learning, representation learning, and multimodal fusion. Alternating diffusion (AD) belongs to the nonlinear diffusion-geometric branch of this broader field, and aims to recover the geometry of the variables common across views while suppressing view-specific nuisance variables [31]. Instead of aligning all views in a common coordinate system or concatenating their features, AD combines the diffusion processes induced by the individual views through a product of Markov transition matrices, thereby using random-walk geometry to recover common latent structure.

For two views, AD is usually described through a latent-variable model. Let U denote the hidden variable shared by the two views, and let η^a and η^b denote view-specific nuisance variables. The observed views are generated by possibly nonlinear maps

$$\boldsymbol{V}^{(1)} = g_a(U, \eta^a), \qquad \boldsymbol{V}^{(2)} = g_b(U, \eta^b). \tag{3.24}$$

The important modelling assumption is that η^a and η^b are conditionally independent given U. This formalises the idea that the two views contain the same underlying content, represented by U, but are corrupted or modified by independent view-specific factors.

Let $\boldsymbol{P}^{(1)}$ and $\boldsymbol{P}^{(2)}$ be the Markov matrices constructed from the two views. The alternating-diffusion operator is defined as a product of these view-specific operators, for example

$$\boldsymbol{P}_{\mathrm{AD}} = \boldsymbol{P}^{(2)}\boldsymbol{P}^{(1)}. \tag{3.25}$$

Since both $\boldsymbol{P}^{(1)}$ and $\boldsymbol{P}^{(2)}$ are row-stochastic, their product is also row-stochastic:

$$\boldsymbol{P}_{\mathrm{AD}}\mathbf{1} = \boldsymbol{P}^{(2)}\boldsymbol{P}^{(1)}\mathbf{1} = \boldsymbol{P}^{(2)}\mathbf{1} = \mathbf{1}. \tag{3.26}$$

Thus, $\boldsymbol{P}_{\mathrm{AD}}$ is again a valid Markov transition matrix.

The entries of this product have a direct probabilistic interpretation:

$$(\boldsymbol{P}_{\mathrm{AD}})_{ij} = \sum_{k=1}^{N} P^{(2)}_{ik} P^{(1)}_{kj}. \tag{3.27}$$

A transition from sample i to sample j is large when there exist intermediate samples k such that the transition is supported by both views. In other words, AD favours relations that are compatible across the two view-specific geometries. If a pair of samples is close only because of a nuisance factor in one view, that relation is less likely to be reinforced by the other view. If the relation reflects the common latent variable X, it is more likely to persist through the alternating product.

This is the main difference between AD and simply averaging or concatenating views. Averaging affinities combines the view-specific similarities directly, while AD combines the corresponding random walks. The product structure means that a transition is filtered through more than one geometry. Consequently, the resulting diffusion operator is designed to emphasise the geometry shared by the views rather than the geometry specific to any one view.

The two possible orderings,

$$\boldsymbol{P}^{(2)}\boldsymbol{P}^{(1)} \qquad \text{and} \qquad \boldsymbol{P}^{(1)}\boldsymbol{P}^{(2)}, \tag{3.28}$$

are generally different matrices because matrix multiplication is not commutative. However, they share the same non-zero eigenvalues. Moreover, in [31], the two orderings induce equivalent diffusion geometries on the common latent variable. Thus, although the matrix entries depend on the chosen convention, the recovered common geometry is not tied to one particular ordering.

Under the conditional-independence assumption, AD has a precise geometric interpretation: after marginalising over the view-specific variables, the alternating operator acts as an effective diffusion operator on the latent space of the common variable X [31]. Thus, diffusion distances computed from $\boldsymbol{P}_{\mathrm{AD}}$ approximate diffusion distances on the common latent geometry, rather than on either observed view alone. This is why AD is useful as a multi-view method: it uses the agreement between view-specific diffusion processes to isolate the geometry shared across views. The construction also extends to more than two views. Given Markov matrices

$$\boldsymbol{P}^{(1)}, \ldots, \boldsymbol{P}^{(M)} \tag{3.29}$$

for M aligned views, one may form an alternating product such as

$$\boldsymbol{P}_{\mathrm{AD}} = \boldsymbol{P}^{(M)}\boldsymbol{P}^{(M-1)} \cdots \boldsymbol{P}^{(1)}. \tag{3.30}$$

This product is row-stochastic because it is a product of row-stochastic matrices. It describes a random walk that alternates across all views, so that transitions are filtered through several view-specific geometries. As in the two-view case, the ordering affects the matrix entries, but it can be shown that the geometries agree in all the orderings.

4

Comparing Neural Representations via Diffusion Geometry

The preceding chapters established two complementary viewpoints. First, a layer representation can be treated as a finite sample from a representation manifold, so that its geometry may be described through pairwise relations, graph-based operators, and Markov diffusion operators. Second, we looked at some applications of these concepts, such as knowledge distillation and multi-view learning. This chapter turns these ideas into concrete similarity measures for neural representations. The results presented in this chapter were published as a preprint [32].

The starting point is an RSM-based similarity measure. Standard measures such as Centered Kernel Alignment and Distance Correlation compare two representations by first converting each representation into a sample-indexed relational matrix and then comparing the centered relational matrices. The methods developed here keep this RSM-based foundation, but reinterpret the centered relational matrix as the nontrivial part of a row-stochastic Markov operator. For a broad class of centered, scale-invariant RSM-based measures, the original similarity score can be recovered exactly after an affine shift-and-rescale into the Markov simplex.

Once an RSM has been rewritten as a Markov matrix, the diffusion-geometric tools introduced earlier become available. Taking powers of the Markov matrix gives multi-scale variants of CKA and DistCorr, where the diffusion time controls whether the comparison emphasises direct pairwise relations or coarser multi-step connectivity. Multiplying the Markov matrices of several layers gives alternating-diffusion variants, where a set of layer representations is fused into a single network-level operator before comparison. This changes the object being compared from layer-to-layer to network-to-network. The similarity measures introduced in this chapter can also be referred to as diffusion-based similarity measures.

4.1 Markov Reformulation of RSM-Based Similarity Measures

Let $\boldsymbol{S} \in \mathbb{R}^{N \times N}$ denote an RSM associated with a representation $\boldsymbol{R} \in \mathbb{R}^{N \times d}$ (Equation 2.38). In the previous chapters, such matrices were used as static relational objects. Here, we show that, after centering and rescaling, such matrices can also be embedded into the space of Markov matrices.

This is useful because a Markov matrix is not merely a matrix of pairwise scores. It defines a random walk on the sample set, as discussed in Chapter 2. The central question of this section is therefore: under what conditions can an RSM-based similarity measure be rewritten in terms of such Markov matrices without changing its value?

The answer relies on two properties that are satisfied by many standard RSM-based similarity measures. First, the measure must act on centered RSMs. Second, it must be invariant to separate positive rescalings of its two arguments. Both CKA and DistCorr have this structure.

4.1.1 Weighted Centering Operators

Let

$$\boldsymbol{q} \in \mathbb{R}^N \tag{4.1}$$

be a probability vector with strictly positive entries:

$$q_i > 0, \qquad \boldsymbol{q}^\top \mathbf{1} = 1. \tag{4.2}$$

Here $\mathbf{1} \in \mathbb{R}^N$ denotes the vector of all ones. We consider a linear $\boldsymbol{q}$-weighted centering operator

$$\boldsymbol{C}_{\boldsymbol{q}} : \mathbb{R}^{N \times N} \to \mathbb{R}^{N \times N}. \tag{4.3}$$

Since $\boldsymbol{C}_{\boldsymbol{q}}$ is a centering operator, it satisfies

$$\boldsymbol{C}_{\boldsymbol{q}}^2 = \boldsymbol{C}_{\boldsymbol{q}}, \qquad \boldsymbol{C}_{\boldsymbol{q}}(\boldsymbol{M})\mathbf{1} = 0, \qquad \boldsymbol{C}_{\boldsymbol{q}}(\mathbf{1}\boldsymbol{q}^\top) = 0 \tag{4.4}$$

for any $\boldsymbol{M} \in \mathbb{R}^{N \times N}$. The first condition states that centering twice is the same as centering once. The second states that the centered matrix has zero row sums. The third states that the rank-one Markov matrix $\mathbf{1}\boldsymbol{q}^\top$ is removed by the centering operation.

The most important special case in this thesis is uniform centering, where

$$\boldsymbol{q} = \frac{1}{N}\mathbf{1}. \tag{4.5}$$

Then the usual centering matrix is

$$\boldsymbol{H} = \boldsymbol{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^\top, \tag{4.6}$$

and double-centering is given by

$$\boldsymbol{C_q}(\boldsymbol{M}) = \boldsymbol{H}\boldsymbol{M}\boldsymbol{H}. \tag{4.7}$$

4.1.2 The Markov Embedding Theorem

We now state the main result. It says that, under the above assumptions, a centered RSM can be shifted and rescaled into a Markov matrix without changing the value of any centered, scale-invariant similarity measure.

Theorem 4.1.1 (Markov reformulation of centered RSM-based measures)**.** *Let m_{RSM} be an RSM-based similarity measure that can be written as*

$$m_{\mathrm{RSM}}(\boldsymbol{S}_1, \boldsymbol{S}_2) = \psi\left(\boldsymbol{C_q}(\boldsymbol{S}_1), \boldsymbol{C_q}(\boldsymbol{S}_2)\right), \tag{4.8}$$

where ψ is invariant to separate positive rescalings, i.e.,

$$\psi(a\boldsymbol{A}, b\boldsymbol{B}) = \psi(\boldsymbol{A}, \boldsymbol{B}) \qquad \forall a, b > 0, \tag{4.9}$$

and $\boldsymbol{q} \in \mathbb{R}^N$ is a probability vector with positive entries and that $\boldsymbol{C_q}$ satisfies (4.4)*. Define*

$$\boldsymbol{P}(\boldsymbol{S}) = \mathbf{1}\boldsymbol{q}^\top + \alpha(\boldsymbol{S})\boldsymbol{C_q}(\boldsymbol{S}), \tag{4.10}$$

where, when $\boldsymbol{C_q}(\boldsymbol{S}) \neq 0$,

$$\alpha(\boldsymbol{S}) = \min_{(\boldsymbol{C_q}(\boldsymbol{S}))_{ij} \neq 0} \frac{q_j}{|(\boldsymbol{C_q}(\boldsymbol{S}))_{ij}|}. \tag{4.11}$$

If $\boldsymbol{C_q}(\boldsymbol{S}) = 0$, we set $\alpha(\boldsymbol{S}) = 1$. Then $\boldsymbol{P}(\boldsymbol{S}_1)$ and $\boldsymbol{P}(\boldsymbol{S}_2)$ are Markov matrices, and

$$m_{\mathrm{RSM}}(\boldsymbol{S}_1, \boldsymbol{S}_2) = \psi\left(\boldsymbol{C_q}(\boldsymbol{P}(\boldsymbol{S}_1)), \boldsymbol{C_q}(\boldsymbol{P}(\boldsymbol{S}_2))\right). \tag{4.12}$$

Proof. For $k \in \{1, 2\}$, define

$$\boldsymbol{A}_k = \boldsymbol{C_q}(\boldsymbol{S}_k). \tag{4.13}$$

By assumption,

$$\boldsymbol{A}_k\mathbf{1} = 0. \tag{4.14}$$

We first show that $\boldsymbol{P}(\boldsymbol{S}_k)$ is entrywise nonnegative. Its (i, j)-th entry is

$$(\boldsymbol{P}(\boldsymbol{S}_k))_{ij} = q_j + \alpha(\boldsymbol{S}_k)(\boldsymbol{A}_k)_{ij}. \tag{4.15}$$

If $\boldsymbol{A}_k = 0$, then

$$\boldsymbol{P}(\boldsymbol{S}_k) = \mathbf{1}\boldsymbol{q}^\top, \tag{4.16}$$

which is clearly nonnegative. Otherwise, since $q_j > 0$ for all j, the minimum in (4.11) is taken over strictly positive values, and therefore $\alpha(\boldsymbol{S}_k) > 0$.

If $(\boldsymbol{A}_k)_{ij} \geq 0$, then

$$q_j + \alpha(\boldsymbol{S}_k)(\boldsymbol{A}_k)_{ij} \geq q_j > 0. \tag{4.17}$$

If $(\boldsymbol{A}_k)_{ij} < 0$, then by definition of $\alpha(\boldsymbol{S}_k)$,

$$\alpha(\boldsymbol{S}_k) \leq \frac{q_j}{|(\boldsymbol{A}_k)_{ij}|} = \frac{q_j}{-(\boldsymbol{A}_k)_{ij}}. \tag{4.18}$$

Rearranging gives

$$q_j + \alpha(\boldsymbol{S}_k)(\boldsymbol{A}_k)_{ij} \geq 0. \tag{4.19}$$

Hence $\boldsymbol{P}(\boldsymbol{S}_k) \geq 0$ entrywise.

We next show that $\boldsymbol{P}(\boldsymbol{S}_k)$ has row sums equal to one. Since $\boldsymbol{q}^\top \mathbf{1} = 1$ and $\boldsymbol{A}_k \mathbf{1} = 0$,

$$\begin{aligned} \boldsymbol{P}(\boldsymbol{S}_k)\mathbf{1} &= \mathbf{1}\boldsymbol{q}^\top \mathbf{1} + \alpha(\boldsymbol{S}_k)\boldsymbol{A}_k \mathbf{1} && (4.20)\\ &= \mathbf{1} + 0 = \mathbf{1}. && (4.21) \end{aligned}$$

Together with nonnegativity, this proves that $\boldsymbol{P}(\boldsymbol{S}_k)$ is a Markov matrix.

It remains to show that the original similarity score is preserved. By linearity of $\boldsymbol{C}_{\boldsymbol{q}}$,

$$\boldsymbol{C}_{\boldsymbol{q}}(\boldsymbol{P}(\boldsymbol{S}_k)) = \boldsymbol{C}_{\boldsymbol{q}}(\mathbf{1}\boldsymbol{q}^\top) + \alpha(\boldsymbol{S}_k)\boldsymbol{C}_{\boldsymbol{q}}(\boldsymbol{A}_k). \tag{4.22}$$

Using (4.4),

$$\boldsymbol{C}_{\boldsymbol{q}}(\mathbf{1}\boldsymbol{q}^\top) = 0, \tag{4.23}$$

and, since $\boldsymbol{C}_{\boldsymbol{q}}$ is idempotent,

$$\boldsymbol{C}_{\boldsymbol{q}}(\boldsymbol{A}_k) = \boldsymbol{C}_{\boldsymbol{q}}(\boldsymbol{C}_{\boldsymbol{q}}(\boldsymbol{S}_k)) = \boldsymbol{C}_{\boldsymbol{q}}(\boldsymbol{S}_k). \tag{4.24}$$

Therefore,

$$\boldsymbol{C}_{\boldsymbol{q}}(\boldsymbol{P}(\boldsymbol{S}_k)) = \alpha(\boldsymbol{S}_k)\boldsymbol{C}_{\boldsymbol{q}}(\boldsymbol{S}_k). \tag{4.25}$$

Using the separate scale invariance of ψ,

$$\begin{aligned} \psi\left(\boldsymbol{C}_{\boldsymbol{q}}(\boldsymbol{P}(\boldsymbol{S}_1)), \boldsymbol{C}_{\boldsymbol{q}}(\boldsymbol{P}(\boldsymbol{S}_2))\right) &= \psi\left(\alpha(\boldsymbol{S}_1)\boldsymbol{C}_{\boldsymbol{q}}(\boldsymbol{S}_1), \alpha(\boldsymbol{S}_2)\boldsymbol{C}_{\boldsymbol{q}}(\boldsymbol{S}_2)\right) && (4.26)\\ &= \psi\left(\boldsymbol{C}_{\boldsymbol{q}}(\boldsymbol{S}_1), \boldsymbol{C}_{\boldsymbol{q}}(\boldsymbol{S}_2)\right) && (4.27)\\ &= m_{\mathrm{RSM}}(\boldsymbol{S}_1, \boldsymbol{S}_2). && (4.28) \end{aligned}$$

This proves the result. □

4.1.3 Double-Sided Weighted Centering

We now verify that the double-sided weighted centering operator satisfies the conditions required in Theorem 4.1.1.

Lemma 4.1.2 (Properties of double-sided weighted centering)**.** *Let*

$$\boldsymbol{C_q}(\boldsymbol{M}) = (\boldsymbol{I} - \mathbf{1}\boldsymbol{q}^\top)\boldsymbol{M}(\boldsymbol{I} - \mathbf{1}\boldsymbol{q}^\top) \tag{4.29}$$

for a probability vector $\boldsymbol{q} \in \mathbb{R}^N$. Then $\boldsymbol{C_q}$ satisfies

$$\boldsymbol{C_q^2} = \boldsymbol{C_q}, \qquad \boldsymbol{C_q}(\boldsymbol{M})\mathbf{1} = 0, \qquad \boldsymbol{C_q}(\mathbf{1}\boldsymbol{q}^\top) = 0. \tag{4.30}$$

For $\boldsymbol{q} = \frac{1}{N}\mathbf{1}$, this reduces to the usual double-centering operator:

$$\boldsymbol{C_q}(\boldsymbol{M}) = \boldsymbol{H}\boldsymbol{M}\boldsymbol{H}. \tag{4.31}$$

Proof. Let

$$\boldsymbol{E_q} = \boldsymbol{I} - \mathbf{1}\boldsymbol{q}^\top. \tag{4.32}$$

Then

$$\boldsymbol{C_q}(\boldsymbol{M}) = \boldsymbol{E_q}\boldsymbol{M}\boldsymbol{E_q}. \tag{4.33}$$

Since $\boldsymbol{q}^\top\mathbf{1} = 1$,

$$\boldsymbol{E_q}\mathbf{1} = \mathbf{1} - \mathbf{1}\boldsymbol{q}^\top\mathbf{1} = \mathbf{1} - \mathbf{1} = 0, \tag{4.34}$$

and

$$\boldsymbol{q}^\top\boldsymbol{E_q} = \boldsymbol{q}^\top - \boldsymbol{q}^\top\mathbf{1}\boldsymbol{q}^\top = \boldsymbol{q}^\top - \boldsymbol{q}^\top = 0^\top. \tag{4.35}$$

Thus,

$$\begin{aligned} \boldsymbol{E_q^2} &= (\boldsymbol{I} - \mathbf{1}\boldsymbol{q}^\top)\boldsymbol{E_q} && (4.36)\\ &= \boldsymbol{E_q} - \mathbf{1}\boldsymbol{q}^\top\boldsymbol{E_q} && (4.37)\\ &= \boldsymbol{E_q}. && (4.38) \end{aligned}$$

It follows that

$$\begin{aligned} \boldsymbol{C_q}(\boldsymbol{C_q}(\boldsymbol{M})) &= \boldsymbol{E_q}(\boldsymbol{E_q}\boldsymbol{M}\boldsymbol{E_q})\boldsymbol{E_q} && (4.39)\\ &= \boldsymbol{E_q^2}\boldsymbol{M}\boldsymbol{E_q^2} && (4.40)\\ &= \boldsymbol{E_q}\boldsymbol{M}\boldsymbol{E_q} = \boldsymbol{C_q}(\boldsymbol{M}). && (4.41) \end{aligned}$$

This proves idempotence.

Next, using $\boldsymbol{E_q}\mathbf{1} = 0$,

$$\boldsymbol{C_q}(\boldsymbol{M})\mathbf{1} = \boldsymbol{E_q}\boldsymbol{M}\boldsymbol{E_q}\mathbf{1} = 0. \tag{4.42}$$

Finally,

$$\boldsymbol{C}_{\boldsymbol{q}}(\mathbf{1}\boldsymbol{q}^\top) = \boldsymbol{E}_{\boldsymbol{q}}\mathbf{1}\boldsymbol{q}^\top\boldsymbol{E}_{\boldsymbol{q}} = 0. \tag{4.43}$$

For $\boldsymbol{q} = \frac{1}{N}\mathbf{1}$,

$$\boldsymbol{E}_{\boldsymbol{q}} = \boldsymbol{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^\top = \boldsymbol{H}, \tag{4.44}$$

so $\boldsymbol{C}_{\boldsymbol{q}}(\boldsymbol{M}) = \boldsymbol{H}\boldsymbol{M}\boldsymbol{H}$. □

Although the rest of the chapter focuses on CKA and DistCorr, the theorem is not restricted to these two cases; other centered, scale-invariant RSM-based measures can be treated analogously when their centering and normalisation operations fit the same template.

4.2 CKA and DistCorr as Markov-Matrix Similarities

Theorem 4.1.1 applies to a broad class of RSM-based similarity measures. In this section, we verify explicitly that CKA and DistCorr satisfy the theorem's assumptions. This is important because these are the two measures used as base measures for the diffusion-based constructions later in the chapter.

4.2.1 Centered Kernel Alignment

Recall that CKA compares two kernel RSMs $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ through

$$m_{\mathrm{CKA}}(\boldsymbol{S}_1, \boldsymbol{S}_2) = \frac{\mathrm{HSIC}(\boldsymbol{S}_1, \boldsymbol{S}_2)}{\sqrt{\mathrm{HSIC}(\boldsymbol{S}_1, \boldsymbol{S}_1)\,\mathrm{HSIC}(\boldsymbol{S}_2, \boldsymbol{S}_2)}}, \tag{4.45}$$

where

$$\mathrm{HSIC}(\boldsymbol{S}_1, \boldsymbol{S}_2) = \frac{1}{(N-1)^2}\,\mathrm{tr}(\boldsymbol{S}_1\boldsymbol{H}\boldsymbol{S}_2\boldsymbol{H}). \tag{4.46}$$

The centering matrix is

$$\boldsymbol{H} = \boldsymbol{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^\top. \tag{4.47}$$

Corollary 4.2.1 (Markov reformulation of CKA)**.** *CKA can be written in the Markov form*

$$m_{\mathrm{CKA}}(\boldsymbol{S}_1, \boldsymbol{S}_2) = \psi\left(\boldsymbol{C}_{\boldsymbol{q}}(\boldsymbol{P}(\boldsymbol{S}_1)), \boldsymbol{C}_{\boldsymbol{q}}(\boldsymbol{P}(\boldsymbol{S}_2))\right), \tag{4.48}$$

with Markov matrices $\boldsymbol{P}(\boldsymbol{S}_1)$ and $\boldsymbol{P}(\boldsymbol{S}_2)$ obtained from Theorem 4.1.1.

Proof. Let

$$\boldsymbol{C}(\boldsymbol{M}) = \boldsymbol{H}\boldsymbol{M}\boldsymbol{H}. \tag{4.49}$$

Since $\boldsymbol{H}^\top = \boldsymbol{H}$ and $\boldsymbol{H}^2 = \boldsymbol{H}$,

$$\begin{aligned}
\mathrm{HSIC}(\boldsymbol{C}(\boldsymbol{S}_1), \boldsymbol{C}(\boldsymbol{S}_2)) &= \frac{1}{(N-1)^2} \operatorname{tr}\left(\boldsymbol{H}\boldsymbol{S}_1\boldsymbol{H}\boldsymbol{H}\boldsymbol{H}\boldsymbol{S}_2\boldsymbol{H}\boldsymbol{H}\right) && (4.50)\\
&= \frac{1}{(N-1)^2} \operatorname{tr}\left(\boldsymbol{H}\boldsymbol{S}_1\boldsymbol{H}\boldsymbol{S}_2\boldsymbol{H}\right) && (4.51)\\
&= \frac{1}{(N-1)^2} \operatorname{tr}\left(\boldsymbol{S}_1\boldsymbol{H}\boldsymbol{S}_2\boldsymbol{H}\right) && (4.52)\\
&= \mathrm{HSIC}(\boldsymbol{S}_1, \boldsymbol{S}_2). && (4.53)
\end{aligned}$$

Define

$$\psi(\boldsymbol{A}, \boldsymbol{B}) = \frac{\mathrm{HSIC}(\boldsymbol{A}, \boldsymbol{B})}{\sqrt{\mathrm{HSIC}(\boldsymbol{A}, \boldsymbol{A})\,\mathrm{HSIC}(\boldsymbol{B}, \boldsymbol{B})}}. \tag{4.54}$$

Then

$$m_{\mathrm{CKA}}(\boldsymbol{S}_1, \boldsymbol{S}_2) = \psi(\boldsymbol{C}(\boldsymbol{S}_1), \boldsymbol{C}(\boldsymbol{S}_2)). \tag{4.55}$$

Moreover, for $a, b > 0$,

$$\mathrm{HSIC}(a\boldsymbol{A}, b\boldsymbol{B}) = ab\ \mathrm{HSIC}(\boldsymbol{A}, \boldsymbol{B}), \tag{4.56}$$

so the normalisation in ψ cancels the factors a and b. Therefore, ψ is invariant to separate positive rescalings. Lemma 4.1.2 shows that the centering operator satisfies the assumptions of Theorem 4.1.1. Hence CKA admits the claimed Markov reformulation. □

4.2.2 Distance Correlation

Distance Correlation is naturally formulated in terms of distance or dissimilarity matrices. Let $\boldsymbol{S}_1, \boldsymbol{S}_2 \in \mathbb{R}^{N\times N}$ denote sample-indexed distance or dissimilarity matrices. After double-centering, the distance covariance can be written using the Frobenius inner product:

$$\mathrm{dCov}^2(\boldsymbol{S}_1, \boldsymbol{S}_2) = \frac{1}{N^2} \left\langle \boldsymbol{H}\boldsymbol{S}_1\boldsymbol{H}, \boldsymbol{H}\boldsymbol{S}_2\boldsymbol{H} \right\rangle_F. \tag{4.57}$$

The normalised distance correlation score is

$$m_{\mathrm{DistCorr}}(\boldsymbol{S}_1, \boldsymbol{S}_2) = \frac{\left\langle \boldsymbol{H}\boldsymbol{S}_1\boldsymbol{H}, \boldsymbol{H}\boldsymbol{S}_2\boldsymbol{H} \right\rangle_F}{\|\boldsymbol{H}\boldsymbol{S}_1\boldsymbol{H}\|_F \|\boldsymbol{H}\boldsymbol{S}_2\boldsymbol{H}\|_F}. \tag{4.58}$$

Corollary 4.2.2 (Markov reformulation of DistCorr)**.** *DistCorr can be written in the Markov form*

$$m_{\mathrm{DistCorr}}(\boldsymbol{S}_1, \boldsymbol{S}_2) = \psi\left(\boldsymbol{C}_{\boldsymbol{q}}(\boldsymbol{P}(\boldsymbol{S}_1)), \boldsymbol{C}_{\boldsymbol{q}}(\boldsymbol{P}(\boldsymbol{S}_2))\right), \tag{4.59}$$

with Markov matrices $\boldsymbol{P}(\boldsymbol{S}_1)$ and $\boldsymbol{P}(\boldsymbol{S}_2)$ obtained from Theorem 4.1.1.

Proof. Let

$$\boldsymbol{C}(\boldsymbol{M}) = \boldsymbol{H}\boldsymbol{M}\boldsymbol{H}. \tag{4.60}$$

For already centered matrices, distance covariance is proportional to the Frobenius inner product:

$$\mathrm{dCov}^2(\boldsymbol{S}_1, \boldsymbol{S}_2) = \frac{1}{N^2}\langle \boldsymbol{S}_1, \boldsymbol{S}_2\rangle_F. \tag{4.61}$$

For uncentered matrices, this becomes

$$\mathrm{dCov}^2(\boldsymbol{S}_1, \boldsymbol{S}_2) = \frac{1}{N^2}\langle \boldsymbol{C}(\boldsymbol{S}_1), \boldsymbol{C}(\boldsymbol{S}_2)\rangle_F. \tag{4.62}$$

Define

$$\psi(\boldsymbol{A}, \boldsymbol{B}) = \frac{\langle \boldsymbol{A}, \boldsymbol{B}\rangle_F}{\|\boldsymbol{A}\|_F \|\boldsymbol{B}\|_F}. \tag{4.63}$$

Then

$$m_{\mathrm{DistCorr}}(\boldsymbol{S}_1, \boldsymbol{S}_2) = \psi(\boldsymbol{C}(\boldsymbol{S}_1), \boldsymbol{C}(\boldsymbol{S}_2)). \tag{4.64}$$

The function ψ is invariant to separate positive rescalings because

$$\frac{\langle a\boldsymbol{A}, b\boldsymbol{B}\rangle_F}{\|a\boldsymbol{A}\|_F \|b\boldsymbol{B}\|_F} = \frac{ab\langle \boldsymbol{A}, \boldsymbol{B}\rangle_F}{ab\|\boldsymbol{A}\|_F \|\boldsymbol{B}\|_F} = \psi(\boldsymbol{A}, \boldsymbol{B}) \tag{4.65}$$

for $a, b > 0$. Lemma 4.1.2 verifies the centering conditions. Therefore, Theorem 4.1.1 applies and gives the desired Markov reformulation. □

4.3 Multi-Scale Diffusion Similarity Measures

The Markov reformulation above turns an RSM into a transition matrix $\boldsymbol{P}(\boldsymbol{S})$. Once this has been done, the representation is no longer only described by direct pairwise affinities. It is also described by the random walk generated by those affinities. This allows us to use diffusion geometry (Section 2.5).

For an integer $t \geq 1$, the matrix power

$$\boldsymbol{P}(\boldsymbol{S})^t \tag{4.66}$$

describes t-step transitions of the random walk. Its (i, j)-th entry aggregates all paths of length t from sample i to sample j. Thus, $\boldsymbol{P}(\boldsymbol{S})$ probes the immediate relational structure encoded by the RSM, whereas $\boldsymbol{P}(\boldsymbol{S})^t$ probes increasingly coarse connectivity patterns as t increases. Two representations may differ in some local pairwise affinities but still induce similar large-scale diffusion behaviour. Conversely, two representations may have similar average pairwise alignments but differ in how neighbourhoods are connected through longer paths. The diffusion viewpoint therefore enriches RSM-based comparison by making scale an explicit parameter.

Since any power of a row-stochastic matrix is still row-stochastic, $\boldsymbol{P}(\boldsymbol{S})^t$ is again a valid Markov matrix. We can therefore substitute these powers into the Markov versions of CKA and DistCorr.

4.3.1 Multi-Scale CKA

Definition 4.3.1 (Multi-Scale CKA)**.** For RSMs $\boldsymbol{S}_1, \boldsymbol{S}_2 \in \mathbb{R}^{N\times N}$ and an integer $t \geq 1$, the Multi-Scale CKA similarity measure is defined as

$$m^{(t)}_{\text{MS-CKA}}(\boldsymbol{S}_1, \boldsymbol{S}_2) = \frac{\text{HSIC}\left(\boldsymbol{P}(\boldsymbol{S}_1)^t, \boldsymbol{P}(\boldsymbol{S}_2)^t\right)}{\sqrt{\text{HSIC}\left(\boldsymbol{P}(\boldsymbol{S}_1)^t, \boldsymbol{P}(\boldsymbol{S}_1)^t\right)\text{HSIC}\left(\boldsymbol{P}(\boldsymbol{S}_2)^t, \boldsymbol{P}(\boldsymbol{S}_2)^t\right)}}. \tag{4.67}$$

For the uniform choice $\boldsymbol{q} = \frac{1}{N}\mathbf{1}$, the Markov matrix used in (4.67) is

$$\boldsymbol{P}(\boldsymbol{S}) = \frac{1}{N}\mathbf{1}\mathbf{1}^\top + \alpha(\boldsymbol{S})\boldsymbol{H}\boldsymbol{S}\boldsymbol{H}, \tag{4.68}$$

with

$$\alpha(\boldsymbol{S}) = \min_{(\boldsymbol{H}\boldsymbol{S}\boldsymbol{H})_{ij}\neq 0} \frac{1}{N|(\boldsymbol{H}\boldsymbol{S}\boldsymbol{H})_{ij}|}. \tag{4.69}$$

The outer centering is already part of the HSIC expression, so it does not need to be written separately.

4.3.2 Multi-Scale DistCorr

Definition 4.3.2 (Multi-Scale DistCorr)**.** For RSMs or distance matrices $\boldsymbol{S}_1, \boldsymbol{S}_2 \in \mathbb{R}^{N\times N}$ and an integer $t \geq 1$, the Multi-Scale DistCorr similarity measure is defined as

$$m^{(t)}_{\text{MS-DistCorr}}(\boldsymbol{S}_1, \boldsymbol{S}_2) = \frac{\langle \boldsymbol{H}\boldsymbol{P}(\boldsymbol{S}_1)^t\boldsymbol{H}, \boldsymbol{H}\boldsymbol{P}(\boldsymbol{S}_2)^t\boldsymbol{H}\rangle_F}{\|\boldsymbol{H}\boldsymbol{P}(\boldsymbol{S}_1)^t\boldsymbol{H}\|_F \|\boldsymbol{H}\boldsymbol{P}(\boldsymbol{S}_2)^t\boldsymbol{H}\|_F}. \tag{4.70}$$

Here, the centering is written explicitly because the measure is expressed as a normalised Frobenius inner product.

For $t = 1$, the multi-scale measures reduce to standard CKA and DistCorr through the Markov equivalence in Corollaries 4.2.1 and 4.2.2. For larger t, the comparison shifts from direct pairwise relations toward diffusion geometry. This provides a tunable scale parameter: small t emphasises local sample geometry, while larger t emphasises coarser connectivity structure.

4.4 Alternating-Diffusion Similarity Measures

The multi-scale measures introduced above manipulate the Markov matrix of a single representation. Alternating diffusion gives a complementary construction: rather than changing the diffusion scale of one representation, it fuses the Markov matrices of several aligned representations. The following methods are developed by applying concepts from Section 3.2 in the context of representation similarity.

Multi-view learning is natural for neural networks. A network does not produce only one representation. It produces a sequence of layer representations

$$\boldsymbol{R}^{(1)}, \ldots, \boldsymbol{R}^{(L)}, \tag{4.71}$$

all computed on the same input samples. These layers can be regarded as multiple aligned views of the same dataset. Each layer may contain shared sample-level structure as well as layer-specific transformations. Alternating diffusion combines these views by multiplying their Markov matrices.

4.4.1 Multi-Representation Similarity

Let

$$\mathcal{A} = \{\boldsymbol{R}_i \in \mathbb{R}^{N \times D_i}\}_{i=1}^{n} \tag{4.72}$$

be a finite set of sample-wise aligned representations. The representations may have different ambient dimensions D_i, but their rows index the same N samples. Let $\mathfrak{A}$ denote the collection of all such finite sets, where both n and the dimensions D_i may vary. Let

$$\mathcal{Y} \subset \mathbb{R}^{N \times N} \tag{4.73}$$

denote the set of all $N \times N$ row-stochastic matrices.

Definition 4.4.1 (Multi-representation similarity measure)**.** Given a fusion map

$$\mathcal{F} : \mathfrak{A} \to \mathcal{Y} \tag{4.74}$$

and a Markov-matrix similarity measure $m_{\boldsymbol{P}}$, the multi-representation similarity between two representation sets $\mathcal{A}_1, \mathcal{A}_2 \in \mathfrak{A}$ is

$$m_{\mathrm{MR}}(\mathcal{A}_1, \mathcal{A}_2) = m_{\boldsymbol{P}}\left(\mathcal{F}(\mathcal{A}_1), \mathcal{F}(\mathcal{A}_2)\right). \tag{4.75}$$

The fusion map collapses an entire collection of aligned representations into a single Markov matrix, while $m_{\boldsymbol{P}}$ compares the resulting operators.

4.4.2 Alternating-Diffusion

Definition 4.4.2 (Alternating-diffusion fusion map)**.** For

$$\mathcal{A} = \{\boldsymbol{R}_i\}_{i=1}^{n} \in \mathfrak{A}, \tag{4.76}$$

let $\boldsymbol{S}_i$ denote the RSM associated with $\boldsymbol{R}_i$, and let $\boldsymbol{P}(\boldsymbol{S}_i)$ be the Markov matrix obtained from Theorem 4.1.1. The alternating-diffusion fusion map is

$$\mathcal{F}_{\mathrm{AD}}(\mathcal{A}) = \boldsymbol{P}(\boldsymbol{S}_n)\boldsymbol{P}(\boldsymbol{S}_{n-1}) \cdots \boldsymbol{P}(\boldsymbol{S}_1). \tag{4.77}$$

Because each factor is row-stochastic, the product is row-stochastic:

$$\mathcal{F}_{\mathrm{AD}}(\mathcal{A})\mathbf{1} = \mathbf{1}. \tag{4.78}$$

Therefore,

$$\mathcal{F}_{\mathrm{AD}}(\mathcal{A}) \in \mathcal{Y}. \tag{4.79}$$

For $n = 2$, this recovers the usual alternating-diffusion construction applied to two views. For $n > 2$, it extends the same idea to several aligned representations.

4.4.3 AD-CKA and AD-DistCorr

Combining the alternating-diffusion fusion map with the Markov reformulations of CKA and DistCorr gives network-level similarity measures.

Definition 4.4.3 (AD-CKA)**.** For two sets of sample-wise aligned representations $\mathcal{A}_1, \mathcal{A}_2 \in \mathfrak{A}$, the AD-CKA similarity measure is

$$m_{\text{AD-CKA}}(\mathcal{A}_1, \mathcal{A}_2) = \frac{\text{HSIC}\left(\mathcal{F}_{\text{AD}}(\mathcal{A}_1), \mathcal{F}_{\text{AD}}(\mathcal{A}_2)\right)}{\sqrt{\text{HSIC}\left(\mathcal{F}_{\text{AD}}(\mathcal{A}_1), \mathcal{F}_{\text{AD}}(\mathcal{A}_1)\right) \text{HSIC}\left(\mathcal{F}_{\text{AD}}(\mathcal{A}_2), \mathcal{F}_{\text{AD}}(\mathcal{A}_2)\right)}}. \tag{4.80}$$

Definition 4.4.4 (AD-DistCorr)**.** For two sets of sample-wise aligned representations $\mathcal{A}_1, \mathcal{A}_2 \in \mathfrak{A}$, the AD-DistCorr similarity measure is

$$m_{\text{AD-DistCorr}}(\mathcal{A}_1, \mathcal{A}_2) = \frac{\langle \boldsymbol{H}\mathcal{F}_{\text{AD}}(\mathcal{A}_1)\boldsymbol{H}, \boldsymbol{H}\mathcal{F}_{\text{AD}}(\mathcal{A}_2)\boldsymbol{H}\rangle_F}{\|\boldsymbol{H}\mathcal{F}_{\text{AD}}(\mathcal{A}_1)\boldsymbol{H}\|_F \|\boldsymbol{H}\mathcal{F}_{\text{AD}}(\mathcal{A}_2)\boldsymbol{H}\|_F}. \tag{4.81}$$

For singleton sets $\mathcal{A}_1 = \{\boldsymbol{R}_1\}$ and $\mathcal{A}_2 = \{\boldsymbol{R}_2\}$, the alternating-diffusion product consists of only one factor. In that case, AD-CKA and AD-DistCorr reduce to the standard CKA and DistCorr formulations above.

4.4.4 Assumptions and the Nonlocal Nature of the Markov Kernels

Following the discussion in Section 3.2.3, all observed view-specific variables corresponding to the different views are conditionally independent given the common variable.

In the neural-representation setting, this assumption is only an idealisation. Successive layers of a neural network are not conditionally independent views: later layers are deterministic functions of earlier layers. Nevertheless, the multi-view analogy is still useful. The common variable can be understood as the sample identity or semantic content shared across layers, while layer-specific structure corresponds to the transformations introduced at each depth.

Another important difference from classical alternating diffusion is locality. The usual analysis assumes local Markov kernels, often obtained from Gaussian affinities with small bandwidths. In contrast, the Markov matrices constructed from Theorem 4.1.1 are generally nonlocal. Indeed, from

$$\boldsymbol{P}(\boldsymbol{S}) = \mathbf{1}\boldsymbol{q}^\top + \alpha(\boldsymbol{S})\boldsymbol{C}_{\boldsymbol{q}}(\boldsymbol{S}), \tag{4.82}$$

the entries of $\boldsymbol{P}(\boldsymbol{S})$ fluctuate around the baseline distribution $\boldsymbol{q}$. Since $\alpha(\boldsymbol{S})$ is chosen so that all entries remain nonnegative, the perturbation away from $\mathbf{1}\boldsymbol{q}^\top$ is bounded entrywise. Consequently, a single transition is not necessarily localised in the ambient representation space, even when the original RSM was constructed from a local similarity such as an RBF kernel.

This means that the Markov matrices in this chapter should be interpreted as RSM-derived Markov operators rather than as classical local graph random walks. The diffusion-geometric manipulations remain algebraically valid because the matrices are row-stochastic, but their continuum interpretation is different from the local graph Laplacians discussed in the previous chapter. Empirically, the experiments in the next chapter show that AD-style fusion can still be useful in this nonlocal setting.

5

Diffusion-based Similarity Measures: Experimental Evaluation

The theoretical construction in the previous chapter gives a systematic way to lift centered RSM-based similarity measures into the space of Markov operators. The empirical question is whether this additional diffusion-geometric structure improves representation comparison in benchmark settings where similarity is grounded in model behavior. The answer is affirmative. Across the prediction-grounded ReSi settings reported below, the proposed diffusion-based measures obtain state-of-the-art (SoTA) performance in seven of the eight reported comparisons. The strongest gains occur when a single layer is replaced by a fused network-level diffusion operator, showing that the information relevant for functional comparison is often distributed across depth rather than concentrated in one final representation.

This chapter evaluates the proposed measures on two public benchmarks: the Representational Similarity (ReSi) benchmark and the Grounding Representational Similarity with statistical testing (GRS) benchmark [16, 33]. ReSi provides a broad comparison against many existing similarity measures across language and vision settings. GRS provides an additional out-of-distribution (OOD) stress test, where representational similarity is evaluated by how well it tracks OOD accuracy differences. Together, these benchmarks test whether diffusion-based similarity measures merely define a mathematically valid extension of CKA and DistCorr, or whether they identify a stronger empirical notion of network similarity. The results support the latter interpretation. Additional implementation details and extended ReSi results, including the full layer-selection table and supplementary benchmark settings, are reported in Appendix A.1.

5.1 ReSi Benchmark

The ReSi benchmark evaluates similarity measures on fourteen architectures trained with ten random seeds each on seven datasets spanning vision, language, and graph tasks. The prediction-grounded tests compare the similarities between all pairs of models trained under the same architecture-dataset setting and rank-correlate these similarities with functional differences between the corresponding models.

Let $\mathcal{F}$ denote a set of trained models under the same architecture-dataset setting, differing only by random seed. For models $f, f' \in \mathcal{F}$, let $\boldsymbol{R}_f$ and $\boldsymbol{R}_{f'}$ denote their representations, and let $m(\boldsymbol{R}_f, \boldsymbol{R}_{f'})$ be a similarity score. The benchmark asks whether this similarity score is informative about observable behavioral differences between f and f'.

5.1.1 Test 1: Correlation to Accuracy Difference

For each model pair (f, f'), ReSi computes the absolute difference in classification accuracy:

$$\Delta_{\text{Acc}}(f, f') = \left|\text{Acc}(\boldsymbol{O}_f, \boldsymbol{y}) - \text{Acc}(\boldsymbol{O}_{f'}, \boldsymbol{y})\right|, \tag{5.1}$$

where $\boldsymbol{O}_f$ denotes the output probabilities of model f on the evaluation set and $\boldsymbol{y}$ denotes the labels. The benchmark reports the Spearman rank correlation between the set of representation similarities $m(\boldsymbol{R}_f, \boldsymbol{R}_{f'})$ and the corresponding accuracy differences $\Delta_{\text{Acc}}(f, f')$ over all model pairs. A high correlation means that the similarity measure correctly orders model pairs by how close their predictive performance is.

5.1.2 Test 2: Correlation to Output Difference

Test 2 compares model outputs at the instance level. The hard-prediction disagreement between f and f' is

$$\Delta_{\text{Dis}}(f, f') = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}\left[\arg\max \boldsymbol{O}_f^{(i)} \neq \arg\max \boldsymbol{O}_{f'}^{(i)}\right], \tag{5.2}$$

and the average Jensen-Shannon divergence of the output probability vectors is

$$\Delta_{\text{JSD}}(f, f') = \frac{1}{n} \sum_{i=1}^{n} \text{JSD}\left(\boldsymbol{O}_f^{(i)} \,\|\, \boldsymbol{O}_{f'}^{(i)}\right). \tag{5.3}$$

The benchmark again reports the Spearman rank correlation between representation similarity and these output-difference quantities. This test is stricter than accuracy difference alone because it evaluates whether two models make similar predictions on the same individual examples, not merely whether their aggregate accuracies are close.

5.1.3 Integration of the Proposed Measures

For the multi-scale variants, integration into ReSi is direct because MS-CKA and MS-DistCorr operate on the same single-layer inputs as standard CKA and DistCorr. The only change is that the Markov matrix obtained from the RSM is replaced by a diffusion power before the final similarity score is computed. Thus, these variants test whether changing the diffusion scale of a representation improves prediction-grounded similarity. Computationally, the multi-scale variants are cheaper than the AD variants because they modify a single-layer Markov operator rather than constructing and multiplying operators from several layers.

For the multi-layer variants, the object being compared is no longer a single representation. For each network f, we construct a set of selected layer representations

$$\mathcal{A}_f = \{\boldsymbol{R}_f^{(\ell_1)}, \ldots, \boldsymbol{R}_f^{(\ell_M)}\}, \tag{5.4}$$

apply the AD fusion map to obtain

$$\mathcal{F}_{\mathrm{AD}}(\mathcal{A}_f), \tag{5.5}$$

and then compare networks using AD-CKA or AD-DistCorr. This turns the ReSi comparison from a layer-level comparison into a network-level comparison while leaving the benchmark task definitions and scoring procedures unchanged.

This distinction is central to the empirical contribution of the chapter. The AD measures do not simply reweight an existing final-layer comparison. They construct a new object: a Markov operator obtained by composing the diffusion geometry of several aligned layers. The benchmark therefore tests whether this network-level operator captures behaviorally relevant information that is missed by standard single-layer measures. The SoTA results below show that it does.

For language models, ReSi uses the CLS token representation for BERT and the final prompt token for SmolLM2. The layer selections used for the multi-layer measures in the settings reported in Table 5.2 are listed in Table 5.1.

Table 5.1: Representations included in the set $\mathcal{A}$ for the ReSi settings reported in Table 5.2. The indices refer to the representations extracted by the ReSi benchmark. For language models, the zeroth extracted representation is the [embedding + positional encoding] layer.

Domain	Dataset	Target difference	Architecture / indices
Language	SST-2	Accuracy	BERT base: $[1, 3, 6, 9, 12]$ SmolLM2: $[17, 21, 24]$
		JSD output	BERT base: $[11, 12]$ SmolLM2: $[21, 22, 23, 24]$
Vision	ImageNet100	Accuracy	ResNet18: $[1, 4, 7, 9]$ ResNet101: $[0, 5, 11, 17, 23, 28, 34]$ VGG11: $[4, 6, 8]$ VGG19: $[0, 3, 6, 9, 13, 16]$

5.1.4 Main Results

The main ReSi results are summarised in Table 5.2. The table reports Spearman correlations between similarity scores and prediction-grounded target quantities for a large set of baseline measures and the proposed diffusion-based measures. Higher

values are better. The result is strong: the proposed methods are SoTA in seven of the eight reported ReSi comparisons. AD-CKA is the best method for accuracy-difference correlation in both language settings, AD-DistCorr is the best method for SmolLM2 output-difference correlation in both VGG vision settings, and Multi-Scale CKA is the best method for both ResNet accuracy-difference settings.

Table 5.2: Spearman correlations on the ReSi prediction-grounded tests, comparing baseline measures with the proposed diffusion-based measures. Best performance is shown in bold. Higher values are better.

Dataset		SST-2				ImageNet100			
Correlation to difference		Accuracy ↑		JSD (Output) ↑		Accuracy ↑			
Architecture		BERT	SmolLM2	BERT	SmolLM2	RNet18	RNet101	VGG11	VGG19
CCA	PWCCA [34]	-0.33	–	-0.32	–	-0.02	-0.09	-0.18	0.11
	SVCCA [35]	-0.08	0.40	0.47	0.49	0.29	-0.00	-0.04	-0.30
Alignment	AlignCos [36]	0.02	0.44	0.49	0.39	-0.08	-0.01	-0.13	-0.12
	AngShape [37]	-0.14	0.46	0.40	0.36	0.21	0.15	-0.02	0.03
	HardCorr [38]	-0.33	0.53	-0.01	0.33	0.21	-0.01	-0.01	-0.03
	LinReg [15]	-0.38	0.03	0.02	0.34	0.19	0.09	-0.04	0.09
	OrthProc [33]	-0.14	0.46	0.40	0.36	0.21	0.15	-0.02	0.03
	PermProc [37]	-0.09	0.47	0.04	0.16	0.07	0.08	0.14	-0.02
	ProcDist [37]	0.10	0.57	0.49	0.36	0.08	0.14	0.13	0.08
	SoftCorr [38]	-0.33	0.55	-0.01	0.37	0.27	0.04	-0.03	-0.10
Neighbors	2nd-Cos [39]	0.30	0.10	0.49	0.18	-0.08	0.05	-0.20	-0.18
	Jaccard [40]	0.17	0.31	0.54	0.24	-0.11	-0.04	-0.22	0.07
	RankSim [40]	0.14	0.18	**0.56**	0.15	0.07	0.13	-0.01	0.03
Topology	IMD [41]	-0.06	0.31	0.02	0.23	0.17	0.12	0.08	-0.17
	RTD [42]	0.05	0.32	0.33	0.27	0.09	-0.20	-0.19	0.05
Statistic	ConcDiff [40]	0.12	0.06	-0.02	-0.11	-0.11	-0.04	-0.11	-0.13
	MagDiff [40]	-0.13	-0.10	0.11	-0.07	-0.16	-0.06	-0.07	-0.12
	UnifDiff [43]	0.35	-0.13	0.38	0.05	-0.18	–	0.17	-0.04
RSM	CKA [15]	-0.06	0.48	0.48	0.52	0.36	0.16	0.03	-0.20
	DistCorr [22]	-0.10	0.49	0.51	0.57	0.31	0.08	0.03	-0.21
	EOS [44]	-0.38	-0.27	-0.21	0.06	0.05	0.11	-0.22	0.08
	GULP [45]	-0.36	–	-0.30	–	0.02	0.12	-0.17	0.10
	RSA [14]	-0.23	0.39	0.44	0.26	0.06	0.09	0.24	-0.35
	RSMDiff [46]	0.20	0.36	0.24	-0.03	0.09	0.11	-0.04	-0.08
Diffusion-based (Ours)	Multi-Scale CKA	-0.06	0.46	0.39	0.51	**0.38**	**0.18**	0.01	-0.21
	AD-CKA	**0.43**	**0.58**	0.40	0.66	0.17	0.06	0.27	0.10
	Multi-Scale DistCorr	-0.12	0.49	0.41	0.57	0.31	0.06	0.02	-0.23
	AD-DistCorr	0.25	0.55	0.38	**0.74**	0.17	0.06	**0.32**	**0.19**

The language results provide the clearest evidence that network-level diffusion aggregation captures information not available to standard single-layer measures. In Test 1, where the target is the difference in accuracy, AD-CKA gives the best result for both language architectures. On BERT, the improvement is substantial: AD-CKA reaches 0.43, exceeding the strongest non-proposed baseline, UnifDiff at 0.35, and far exceeding standard CKA at -0.06 and DistCorr at -0.10. On SmolLM2, AD-CKA reaches the best score, 0.58, slightly above the strongest baseline, ProcDist at 0.57. The consistent SoTA performance across both encoder and decoder language models indicates that aggregating representations across depth gives a more behaviorally aligned similarity signal than selecting a single hidden layer.

For output-level differences, the most informative representations are concentrated near the deepest layers. In the SmolLM2 JSD setting, AD-DistCorr obtains 0.74, a large improvement over the strongest standard RSM baseline, DistCorr at 0.57, and

over AD-CKA at 0.66. This is one of the strongest margins in Table 5.2. It suggests that the alternating-diffusion product is not merely smoothing or averaging layer information; rather, it is extracting a network-level relational structure that aligns closely with sample-wise output behavior. The only reported ReSi setting in which the proposed measures do not obtain the best score is BERT JSD, where RankSim reaches 0.56.

The vision results show that the best diffusion-based variant depends on the architecture. For ResNet18 and ResNet101, Multi-Scale CKA is SoTA, obtaining 0.38 and 0.18, respectively. These results indicate that, for residual architectures, modifying the diffusion scale of the final-layer relational geometry can be sufficient to improve prediction-grounded similarity. In contrast, for VGG11 and VGG19, AD-DistCorr is SoTA, reaching 0.32 and 0.19, respectively. The gain for VGG11 is especially clear, improving over the strongest baseline RSA at 0.24; for VGG19, AD-DistCorr improves over the strongest non-proposed baselines, which reach 0.11.

The main empirical finding from ReSi is therefore not simply that one new measure performs well in one setting. Rather, the diffusion framework introduces two complementary mechanisms:multi-scale single-layer comparison and alternating-diffusion multi-layer comparison, and at least one of these mechanisms gives SoTA performance in nearly every reported prediction-grounded setting. This supports the central claim of the chapter: behaviorally meaningful representation similarity is often better captured by diffusion geometry than by direct comparison of static RSMs.

5.2 GRS Benchmark

The GRS benchmark provides an additional test of whether representation similarity can track functional differences beyond in-distribution performance. We focus on Benchmark 4, which evaluates correlation with OOD accuracy under multiple sources of training randomness. This benchmark is particularly relevant here because it asks whether a similarity measure can detect functionally meaningful differences between models that are all variants of the same architecture. It is therefore well aligned with the network-level perspective of AD-based similarity.

Let $\mathcal{F}$ denote the set of 100 BERT-medium models obtained from all combinations of 10 pretraining seeds and 10 fine-tuning seeds. Let $\boldsymbol{R}_f^{(\ell)}$ denote the representation of model $f \in \mathcal{F}$ at layer $\ell \in \{1, \ldots, 8\}$. For a fixed out-of-distribution task, let $a_{\mathrm{OOD}}(f)$ be the OOD accuracy of model f. For each layer ℓ, the reference model is chosen as the model with the highest OOD accuracy:

$$f_\ell^\star = \arg\max_{f \in \mathcal{F}} a_{\mathrm{OOD}}(f). \tag{5.6}$$

Every other model is then compared to this reference. The layerwise benchmark score is the rank correlation

$$\mathrm{corr}\left(\left\{d\left(\boldsymbol{R}_{f_\ell^\star}^{(\ell)}, \boldsymbol{R}_f^{(\ell)}\right)\right\}_{f \in \mathcal{F}\setminus\{f_\ell^\star\}}, \left\{\left|a_{\mathrm{OOD}}(f_\ell^\star) - a_{\mathrm{OOD}}(f)\right|\right\}_{f \in \mathcal{F}\setminus\{f_\ell^\star\}}\right), \tag{5.7}$$

where

$$d(\boldsymbol{R}_1, \boldsymbol{R}_2) = 1 - m(\boldsymbol{R}_1, \boldsymbol{R}_2). \tag{5.8}$$

The original GRS setup computes this score separately at each layer and averages over layers.

For the multi-layer measures in this chapter, the adaptation is direct. All eight BERT-medium representations are included in $\mathcal{A}_f$.

Table 5.3: Performance on GRS Benchmark 4. The benchmark evaluates the correlation between representation dissimilarity and out-of-distribution accuracy difference. Both Kendall's τ and Spearman's ρ are reported. Best performance is shown in bold.

	Antonym stress test		Numerical stress test	
Method	τ	ρ	τ	ρ
Procrustes	0.24	0.18	0.07	0.05
CKA	0.23	0.16	0.12	0.08
PWCCA	0.20	0.15	0.03	0.02
AD-CKA	**0.41**	**0.29**	**0.20**	**0.14**

The results are shown in Table 5.3. AD-CKA is SoTA on both OOD stress tests and under both rank-correlation metrics. The margins are large relative to the baseline scores. On the Antonym stress test, AD-CKA improves Kendall's τ from 0.24 to 0.41 and Spearman's ρ from 0.18 to 0.29. On the Numerical stress test, AD-CKA improves Kendall's τ from 0.12 to 0.20 and Spearman's ρ from 0.08 to 0.14.

This is an important result because Benchmark 4 is designed as a challenging OOD setting. Standard layerwise measures achieve only weak correlations, indicating that OOD behavior is difficult to recover from ordinary single-layer representation similarity. AD-CKA substantially improves these correlations by comparing networks through a fused multi-layer diffusion operator. The GRS results therefore reinforce the ReSi conclusion: multi-layer diffusion geometry captures behaviorally relevant structure that is not fully exposed by standard single-layer measures.

5.3 Numerical Stability

A practical limitation of AD-based aggregation is that products of many Markov matrices can degenerate. For an ergodic Markov chain, repeated multiplication drives the transition matrix toward its stationary rank-one limit. Similarly, long products such as

$$\boldsymbol{P}_1 \boldsymbol{P}_2 \cdots \boldsymbol{P}_n \tag{5.9}$$

can lose informative sample-specific structure and become dominated by numerical noise.

This limitation is important because the empirical gains above depend precisely on multi-layer aggregation. The results therefore use a conservative implementation: computations are performed in double precision, and the number of representations in $\mathcal{A}$ is restricted to at most eight. With these choices, the AD measures remain numerically stable in the reported experiments. In particular, the final similarity values are unchanged under random perturbations of the input representations of magnitude up to 10^{-7}.

5.4 Discussion

The main contribution of this chapter is both mathematical and empirical. Mathematically, the chapter shows that a broad class of centered, scale-invariant RSM-based similarity measures can be reformulated exactly as similarities between Markov operators. Empirically, this reformulation leads to new diffusion-based measures that achieve SoTA performance across multiple prediction-grounded representation-similarity benchmarks.

The key technical step is the affine embedding, which turns a centered RSM into a row-stochastic Markov matrix. Since CKA and DistCorr are invariant to separate positive rescalings of their centered arguments, this transformation preserves the original similarity score at diffusion scale $t = 1$. The Markov formulation is therefore not an approximation to CKA or DistCorr. It is an exact reformulation that exposes additional structure: once the RSM has been embedded into the Markov simplex, it can be powered, composed, and interpreted as a diffusion operator.

Two extensions follow from this observation. First, powers of the Markov matrix yield multi-scale variants of standard measures. These measures compare the diffusion geometry induced by the representation at different random-walk scales. The ReSi results show that this is especially effective for residual vision architectures, where Multi-Scale CKA is SoTA for both ResNet18 and ResNet101. Second, alternating diffusion changes the comparison from layer-to-layer to network-to-network. The ReSi and GRS results show that this change is not merely conceptual: AD-CKA and AD-DistCorr obtain SoTA results in language, VGG vision, and OOD stress-test settings.

The most significant empirical conclusion is that the behaviorally relevant geometry of a neural network is often not localised in a single representation. For language models in particular, accuracy differences are best tracked by a broad selection of layers, while output-distribution differences are best tracked by deep layers. For VGG models, multi-layer aggregation is also beneficial, while for residual networks, a multi-scale final-layer comparison is sufficient. These patterns suggest that different architectures distribute behaviorally relevant information across depth in different ways. The proposed framework is able to exploit this because it treats layer representations as aligned views and compares their fused diffusion geometry.

There are also limitations. The Markov kernels constructed from centered RSMs are nonlocal, so their continuum interpretation is different from the local graph Laplacians discussed in the previous chapter. The alternating-diffusion assumptions are also only approximately satisfied by neural network layers, because consecutive layers are deterministic transformations rather than conditionally independent views. Finally, products of many Markov matrices can degenerate toward a stationary rank-one matrix, limiting the number of layers that can be stably aggregated. A possible direction for addressing this limitation is to replace the simple ordered product with multi-view extensions of alternating diffusion designed for larger collections of views [47].

Despite these caveats, the results show that diffusion geometry provides a substantially stronger representation-comparison framework than direct static RSM comparison in many benchmark settings. The chapter therefore establishes a concrete route from layerwise representation similarity to network-level similarity.

6

Knowledge Distillation via Manifold and Diffusion Geometry

The preceding chapters developed two ideas that are brought together in this chapter. The first is the representation-geometric view: a hidden layer can be treated as a finite sample from a representation manifold, and the local geometry of this sample can be encoded by graph Laplacians and Markov diffusion operators. The second is the teacher-student view of knowledge distillation: a student network can be trained not only from labels, but also from structural information supplied by a fixed teacher network. Chapter 4 introduced this second viewpoint through generic representation-level and RSM-based distillation objectives. Here, we introduce distillation objectives using graph operators (Sections 2.3 and 2.5).

The main construction is a graph-Laplacian distillation loss. For each selected layer, both teacher and student representations are converted into weighted k-nearest-neighbour graphs on the same mini-batch of input samples. These graphs define symmetric normalised graph Laplacians. The student is then trained so that its graph Laplacian matches the teacher's graph Laplacian. In this way, the teacher does not prescribe feature coordinates directly. Instead, it prescribes a discrete geometric operator describing how samples are locally connected in the teacher representation space.

6.1 Graph-Laplacian Distillation

Chapter 4 defined an RSM-based distillation objective of the form

$$\mathcal{L}_{\mathrm{RSM}}(\theta) = d_{\mathrm{RSM}}\left(\boldsymbol{S}_T^{(a)}, \boldsymbol{S}_{S_\theta}^{(b)}\right), \tag{6.1}$$

where $\boldsymbol{S}_T^{(a)}$ and $\boldsymbol{S}_{S_\theta}^{(b)}$ are sample-indexed relational matrices constructed from teacher and student representations. The important feature of this formulation is that the loss acts on sample-sample relations rather than on activation coordinates. This makes it suitable when the teacher and student hidden dimensions differ, because both relational matrices live in $\mathbb{R}^{N\times N}$ for a mini-batch of N samples.

The method in this chapter keeps this relational structure but replaces the RSM with a graph operator. Let T be a fixed teacher network and let S_θ be a student network. For a graph mini-batch

$$\mathcal{B} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}, \tag{6.2}$$

let

$$\boldsymbol{R}_T^{(\ell)} \in \mathbb{R}^{N \times d_T^{(\ell)}}, \qquad \boldsymbol{R}_S^{(\ell)}(\theta) \in \mathbb{R}^{N \times d_S^{(\ell)}} \tag{6.3}$$

denote the teacher and student representation matrices at a matched layer ℓ. The rows of these matrices index the same samples.

For each representation, we construct a weighted graph and its symmetric normalised Laplacian. The symmetric normalised adjacency is denoted by

$$\boldsymbol{A} = \boldsymbol{D}^{-1/2} \boldsymbol{W} \boldsymbol{D}^{-1/2}, \tag{6.4}$$

and the corresponding symmetric normalised graph Laplacian is

$$\boldsymbol{L}_{\text{sym}} = \boldsymbol{I} - \boldsymbol{A}. \tag{6.5}$$

Refer to Section 2.3 for context.

6.2 Graph Construction

The graph construction is performed separately for each selected layer. The teacher graph is fixed for a given graph mini-batch, while the student graph changes with the student parameters θ. The construction has four stages: pairwise distances, kNN support selection, kernel weighting, and normalisation.

6.2.1 Pairwise Distances and Directed kNN Supports

For layer ℓ, write the teacher and student row vectors as

$$\boldsymbol{r}_{T,i}^{(\ell)} = \boldsymbol{R}_{T,i:}^{(\ell)}, \qquad \boldsymbol{r}_{S,i}^{(\ell)}(\theta) = \boldsymbol{R}_{S,i:}^{(\ell)}(\theta). \tag{6.6}$$

The squared pairwise distances are

$$\delta_{T,ij}^{(\ell)} = \left\| \boldsymbol{r}_{T,i}^{(\ell)} - \boldsymbol{r}_{T,j}^{(\ell)} \right\|^2, \qquad \delta_{S,ij}^{(\ell)}(\theta) = \left\| \boldsymbol{r}_{S,i}^{(\ell)}(\theta) - \boldsymbol{r}_{S,j}^{(\ell)}(\theta) \right\|^2. \tag{6.7}$$

The diagonal entries are ignored when constructing nearest neighbours. Equivalently, one may set $\delta_{ii} = +\infty$ before selecting nearest neighbours.

Let $\mathcal{N}_{T,k}^{(\ell)}(i)$ be the set of the k nearest neighbours of sample i in the teacher representation at layer ℓ, and let $\mathcal{N}_{S,k}^{(\ell)}(i)$ be the corresponding student set. These define directed binary support masks

$$M_{T,ij}^{(\ell)} = \mathbb{1}\left[j \in \mathcal{N}_{T,k}^{(\ell)}(i) \right], \qquad M_{S,ij}^{(\ell)} = \mathbb{1}\left[j \in \mathcal{N}_{S,k}^{(\ell)}(i) \right]. \tag{6.8}$$

These masks determine which sample pairs can carry nonzero graph weight. Note that these masks can be asymmetric.

6.2.2 Masking

The teacher and student graphs can be masked in different ways. $\boldsymbol{M}_T^{(\ell)}$ and $\boldsymbol{M}_S^{(\ell)}$ denote the directed teacher and student kNN masks before masking. $\boldsymbol{M}_T^{(\ell),*}$ and $\boldsymbol{M}_S^{(\ell),*}$ denote the masks after masking.

$$\text{Union:} \qquad \boldsymbol{M}_T^{(\ell),\star} = \boldsymbol{M}_S^{(\ell),\star} = \boldsymbol{M}_T^{(\ell)} \vee \boldsymbol{M}_S^{(\ell)}, \tag{6.9}$$

$$\text{Mutual:} \qquad \boldsymbol{M}_T^{(\ell),\star} = \boldsymbol{M}_S^{(\ell),\star} = \boldsymbol{M}_T^{(\ell)} \wedge \boldsymbol{M}_S^{(\ell)}, \tag{6.10}$$

$$\text{Independent:} \qquad \boldsymbol{M}_T^{(\ell),\star} = \boldsymbol{M}_T^{(\ell)}, \qquad \boldsymbol{M}_S^{(\ell),\star} = \boldsymbol{M}_S^{(\ell)}, \tag{6.11}$$

$$k = N-1 \text{ (All edges kept):} \qquad \boldsymbol{M}_T^{(\ell),\star} = \boldsymbol{M}_S^{(\ell),\star} = \mathbf{1}\mathbf{1}^\top - \boldsymbol{I}. \tag{6.12}$$

Here, $\vee$ and $\wedge$ are applied entrywise to binary mask matrices, denoting logical OR and logical AND, respectively. The Union mode gives the teacher and student a common support containing an edge whenever either representation regards the pair as neighbouring. The Mutual mode keeps only edges that are present in both neighbourhood systems. The Independent mode allows the teacher and student graphs to use their own neighbourhoods. The last option ($k = N-1$: no kNN) removes sparsification and keeps all off-diagonal sample pairs.

6.2.3 Symmetrisation

After kernel weights have been computed on the masked edges (see Section 2.3 for details), a separate symmetrisation option may be applied. If $\widetilde{\boldsymbol{W}}$ denotes the directed weighted graph, then

$$\text{Union:} \qquad W_{ij} = \max\{\widetilde{W}_{ij}, \widetilde{W}_{ji}\}, \tag{6.13}$$

$$\text{Mutual:} \qquad W_{ij} = \min\{\widetilde{W}_{ij}, \widetilde{W}_{ji}\}, \tag{6.14}$$

$$\text{No symmetrisation:} \qquad W_{ij} = \widetilde{W}_{ij}. \tag{6.15}$$

For binary supports, the max operation corresponds to keeping the union of directed edges, while the min operation corresponds to keeping only reciprocated edges. For weighted graphs, the same operations also determine the final symmetric edge weight.

6.2.4 Teacher Graph: Gaussian Kernel

The teacher graph uses a Gaussian affinity. To account for local scale variation, each sample is assigned a teacher-side local bandwidth

$$\sigma_{T,i}^{(\ell)} = \operatorname{median}_{j:(M_T^{(\ell),\star})_{ij}=1} \left\| d_{T,ij}^{(\ell)} \right\|, \qquad d_{T,ij}^{(\ell)} = \boldsymbol{r}_{T,i}^{(\ell)} - \boldsymbol{r}_{T,j}^{(\ell)}. \tag{6.16}$$

The unnormalised directed teacher weight is then

$$\widetilde{W}_{T,ij}^{(\ell)} = (M_T^{(\ell),\star})_{ij} \exp\left(-\frac{d_{T,ij}^{(\ell),2}}{\sigma_{T,i}^{(\ell)} \sigma_{T,j}^{(\ell)}} \right). \tag{6.17}$$

The teacher graph is treated as a fixed target during student training.

6.2.5 Student Graph: Heavy-Tailed and Hybrid Kernels

The teacher graph is fixed, so its weights can be constructed with a Gaussian kernel. The student graph is different: its weights are part of the optimisation path, because they depend on the student representations. If a Gaussian kernel is used for all student edges, then teacher-neighbour pairs that are far apart in the current student representation may receive extremely small weights and, consequently, extremely small gradients. This is problematic for distillation: precisely those teacher edges that are missing in the student are the edges that should exert a strong attractive force.

To address this, the student graph uses a hybrid kernel. The Gaussian kernel is retained for edges that are already supported by the student geometry, while a heavy-tailed Student-t kernel is used only for edges that are present in the teacher neighbourhood graph but absent from the student neighbourhood graph. This choice is motivated by the same principle that underlies t-SNE: replacing a Gaussian affinity by a heavy-tailed Student-t affinity gives non-negligible attraction between points that are currently too far apart, thereby reducing crowding and improving optimisation [48, 49].

In the present setting, however, the heavy-tailed kernel is not used everywhere. It is applied selectively, only where the student is failing to represent a teacher-neighbour relation. Let

$$d_{S,ij}^{(\ell)}(\theta) = \left\| \boldsymbol{r}_{S,i}^{(\ell)}(\theta) - \boldsymbol{r}_{S,j}^{(\ell)}(\theta) \right\|^2 \tag{6.18}$$

denote the squared student distance at layer ℓ. As in the teacher graph, we use the teacher local scale parameters $\sigma_{T,i}^{(\ell)}$ and define

$$\Sigma_{T,ij}^{(\ell)} = \sigma_{T,i}^{(\ell)} \sigma_{T,j}^{(\ell)}. \tag{6.19}$$

The student distances are additionally rescaled by the mean distance over the student graph edges in the current graph mini-batch, so that the kernel operates on relative rather than raw distances. If $\mu_S^{(\ell)}$ denotes this mean distance, then the normalised squared distance is

$$\widetilde{d}_{S,ij}^{(\ell)}(\theta) = \frac{d_{S,ij}^{(\ell)}(\theta)}{(\mu_S^{(\ell)})^2}. \tag{6.20}$$

This keeps the student weights on a comparable scale across batches and layers. The Gaussian student affinity is

$$\widetilde{W}_{S,ij}^{\mathrm{G},(\ell)} = \exp\left(-\frac{\widetilde{d}_{S,ij}^{(\ell)}(\theta)}{\Sigma_{T,ij}^{(\ell)}\, c_{\ell,g}} \right), \tag{6.21}$$

where $c_{\ell,g} > 0$ is a trainable scale parameter. The heavy-tailed Student-t affinity is

$$\widetilde{W}_{S,ij}^{t,(\ell)} = \left(1 + \frac{\widetilde{\delta}_{S,ij}^{(\ell)}(\theta)}{\Sigma_{T,ij}^{(\ell)}\, c_{\ell,t} \nu_\ell} \right)^{-(\nu_\ell+1)/2}, \tag{6.22}$$

where $\nu_\ell > 1$ is the number of degrees of freedom of the distribution and $c_{\ell,t} > 0$ is a separate trainable scale parameter. Smaller values of ν_ℓ give heavier tails, while larger values move the kernel closer to a Gaussian-like decay. The constraint $\nu_\ell > 1$ is imposed by parameterising the trainable quantity through a softplus transformation and then adding one.

The hybrid kernel is defined using the teacher and student neighbourhood masks. Let

$$\boldsymbol{M}_T^{(\ell)}, \boldsymbol{M}_S^{(\ell)} \in \{0,1\}^{N\times N} \tag{6.23}$$

(as defined in Equation 6.8) denote the directed teacher and student kNN masks. In the hybrid setting, the steps in Section 6.2.2 are replaced by this: the final mask is the union

$$\boldsymbol{M}^{(\ell)} = \boldsymbol{M}_T^{(\ell)} \vee \boldsymbol{M}_S^{(\ell)}. \tag{6.24}$$

The teacher-only part of this mask is

$$\boldsymbol{M}_{\mathrm{T}\setminus\mathrm{S}}^{(\ell)} = \boldsymbol{M}^{(\ell)} \wedge \neg \boldsymbol{M}_S^{(\ell)}. \tag{6.25}$$

Here, $\vee$, $\wedge$, and $\neg$ are applied entrywise to binary mask matrices, denoting logical OR, logical AND, and logical NOT, respectively. The symmetrisation (Section 6.2.3) is applied after this. The student pre-symmetrisation weight is then

$$\widetilde{W}_{S,ij}^{\mathrm{hyb},(\ell)} = \begin{cases} \widetilde{W}_{S,ij}^{t,(\ell)}, & \text{if } (\boldsymbol{M}_{\mathrm{T}\setminus\mathrm{S}}^{(\ell)})_{ij} = 1, \\ \widetilde{W}_{S,ij}^{\mathrm{G},(\ell)}, & \text{otherwise.} \end{cases} \tag{6.26}$$

Finally, the active graph mask is applied:

$$\widetilde{W}_{S,ij}^{(\ell)} = M_{ij}^{(\ell)} \widetilde{W}_{S,ij}^{\mathrm{hyb},(\ell)}. \tag{6.27}$$

This construction has a direct interpretation. If an edge is already present in the student kNN graph, then the student already regards the two samples as local neighbours, and the Gaussian kernel is sufficient. If an edge appears in the teacher graph but not in the student graph, then the student currently places the two samples too far apart relative to the teacher geometry. In that case, the Student-t kernel produces a slower decay with distance and therefore provides a stronger corrective signal. Thus, the heavy-tailed kernel is used only where it is needed: on teacher-supported edges that the student has not yet recovered.

The hybrid-kernel view is conceptually compatible with the graph-based distillation framework. Combining different kernels for different roles is a standard idea in kernel methods and multiple-kernel learning [50, 51]. Here, the combination is an edge-dependent gating rule determined by the mismatch between the teacher and student neighbourhoods. After the hybrid weights are constructed, the same symmetrisation choices as in the teacher graph are applied. The graph Laplacians are then computed using Equations 6.4 and 6.5.

Empirically, this selective heavy-tailed construction is preferable to using the Student-t kernel on all student edges. Applying the Student-t kernel globally makes even already-correct local student edges too heavy-tailed, which can blur the local geometry that the Gaussian kernel captures well. The hybrid version keeps the Gaussian kernel where the student geometry is already locally reasonable and reserves the heavy-tailed correction for the specific failure mode it is intended to fix: teacher neighbours that are not yet neighbours in the student representation.

6.2.6 Normalised Graph Operators

After masking, kernel weighting, and symmetrisation, using the steps described in the previous sections, the teacher and student degree matrices are

$$D_{T,ii}^{(\ell)} = \sum_{j=1}^{N} W_{T,ij}^{(\ell)}, \qquad D_{S,ii}^{(\ell)} = \sum_{j=1}^{N} W_{S,ij}^{(\ell)}. \tag{6.28}$$

The symmetric normalised adjacencies are

$$\boldsymbol{A}_T^{(\ell)} = (\boldsymbol{D}_T^{(\ell)})^{-1/2} \boldsymbol{W}_T^{(\ell)} (\boldsymbol{D}_T^{(\ell)})^{-1/2}, \qquad \boldsymbol{A}_S^{(\ell)}(\theta) = (\boldsymbol{D}_S^{(\ell)})^{-1/2} \boldsymbol{W}_S^{(\ell)}(\theta) (\boldsymbol{D}_S^{(\ell)})^{-1/2}. \tag{6.29}$$

The corresponding symmetric normalised Laplacians are

$$\boldsymbol{L}_{T,\mathrm{sym}}^{(\ell)} = \boldsymbol{I} - \boldsymbol{A}_T^{(\ell)}, \qquad \boldsymbol{L}_{S,\mathrm{sym}}^{(\ell)}(\theta) = \boldsymbol{I} - \boldsymbol{A}_S^{(\ell)}(\theta). \tag{6.30}$$

For the optional diffusion loss, we also use the row-stochastic transition matrices

$$\boldsymbol{P}_T^{(\ell)} = (\boldsymbol{D}_T^{(\ell)})^{-1} \boldsymbol{W}_T^{(\ell)}, \qquad \boldsymbol{P}_S^{(\ell)}(\theta) = (\boldsymbol{D}_S^{(\ell)})^{-1} \boldsymbol{W}_S^{(\ell)}(\theta). \tag{6.31}$$

An optional Coifman-Lafon density normalisation can be inserted before constructing these operators [12]. With parameter $\alpha \geq 0$, the weights are replaced by

$$W_{ij} \leftarrow D_{ii}^{-\alpha} W_{ij} D_{jj}^{-\alpha}, \tag{6.32}$$

and the degrees are recomputed before forming $\boldsymbol{A}$ and $\boldsymbol{P}$. The default setting used here is $\alpha = 0$.

6.3 Graph-Laplacian Distillation Loss

The layerwise manifold distillation loss is defined by matching the symmetric normalised Laplacians:

$$\mathcal{L}_{\mathrm{Lap}}^{(\ell)}(\theta) = \left\| \boldsymbol{L}_{S,\mathrm{sym}}^{(\ell)}(\theta) - \boldsymbol{L}_{T,\mathrm{sym}}^{(\ell)} \right\|_F^2. \tag{6.33}$$

Substituting $\boldsymbol{L}_{\mathrm{sym}} = \boldsymbol{I} - \boldsymbol{A}$ gives

$$\mathcal{L}_{\mathrm{Lap}}^{(\ell)}(\theta) = \left\| (\boldsymbol{I} - \boldsymbol{A}_S^{(\ell)}(\theta)) - (\boldsymbol{I} - \boldsymbol{A}_T^{(\ell)}) \right\|_F^2 \tag{6.34}$$

$$= \left\| \boldsymbol{A}_S^{(\ell)}(\theta) - \boldsymbol{A}_T^{(\ell)} \right\|_F^2. \tag{6.35}$$

Thus, although the method is conceptually a graph-Laplacian matching loss, its implementation can compute the equivalent loss on the symmetric normalised adjacency matrices.

For multiple selected layers, the graph loss is summed with optional layer coefficients $\omega_\ell \geq 0$:

$$\mathcal{L}_{\text{man}}(\theta) = \sum_{\ell=1}^{L} \omega_\ell \mathcal{L}_{\text{Lap}}^{(\ell)}(\theta). \tag{6.36}$$

By default, all coefficients are set to one.

6.4 Optional Diffusion-Power Matching

The Laplacian loss in Eq. (6.33) matches one-step graph geometry. The diffusion viewpoint developed in the previous chapters suggests a stronger condition: the student and teacher should not only agree in their immediate neighbourhoods, but also in the geometry induced by multi-step random walks. This motivates an optional diffusion-power loss on the Markov matrices.

Let $\mathcal{T} = \{t_1, \ldots, t_q\}$ be a set of diffusion times. The diffusion loss at layer ℓ is

$$\mathcal{L}_{\text{diff}}^{(\ell)}(\theta) = \sum_{r=1}^{q} \beta^r \left\| \left(\boldsymbol{P}_S^{(\ell)}(\theta)\right)^{t_r} - \left(\boldsymbol{P}_T^{(\ell)}\right)^{t_r} \right\|_F^2, \tag{6.37}$$

where $\beta \in (0, 1]$ is the diffusion-loss factor.

The full graph loss can therefore be written as

$$\mathcal{L}_{\text{man}}(\theta) = \sum_{\ell=1}^{L} \omega_\ell \left(\mathcal{L}_{\text{Lap}}^{(\ell)}(\theta) + \mathbb{1}_{\text{diff}} \mathcal{L}_{\text{diff}}^{(\ell)}(\theta) \right), \tag{6.38}$$

where $\mathbb{1}_{\text{diff}}$ indicates whether the diffusion loss is enabled. The role of this term is to transfer not only the local teacher graph, but also the coarser diffusion geometry generated by that graph.

6.5 Trainable Student Kernels

The student kernel can either use fixed hyperparameters or trainable layerwise kernel parameters. In the trainable case, each selected layer has three raw parameters

$$\boldsymbol{a}_\ell = (a_{\ell,0}, a_{\ell,1}, a_{\ell,2}) \in \mathbb{R}^3. \tag{6.39}$$

These are mapped to positive kernel parameters by a softplus transformation:

$$\nu_\ell = 1 + \text{softplus}(a_{\ell,0}), \tag{6.40}$$

$$c_{\ell,g} = \text{softplus}(a_{\ell,1}), \tag{6.41}$$

$$c_{\ell,t} = \text{softplus}(a_{\ell,2}). \tag{6.42}$$

The first parameter ν_ℓ is the Student-t degrees-of-freedom parameter and is constrained to be larger than one. The other two parameters are scale factors: $c_{\ell,g}$ for the optional Gaussian student-kernel component and $c_{\ell,t}$ for the Student-t component in Eq. 6.22.

When these parameters are learned, they are optimised jointly with the student network during the distillation step. A log-scale regularisation term is added:

$$\mathcal{L}_{\text{reg}} = \lambda_{\text{reg}} \sum_{\ell=1}^{L} \|\log \boldsymbol{\kappa}_\ell\|_2^2, \qquad \boldsymbol{\kappa}_\ell = (\nu_\ell, c_{\ell,g}, c_{\ell,t}). \tag{6.43}$$

This regularisation keeps the learned kernel parameters close to a unit-scale prior unless the distillation loss provides evidence that another scale or tail parameter is useful.

6.6 Choice of Graph Batch Size

The manifold distillation loss is computed on a batch-induced graph rather than on individual examples. Consequently, the graph batch size is not merely a computational hyperparameter: it determines how well the finite graph Laplacian approximates the geometry of the underlying representation manifold. If too few samples are used, the kNN graph may be poorly connected, the degree estimates become noisy, and the low-frequency spectrum of the graph Laplacian becomes unstable. This is particularly important for the proposed method because the loss matches normalised graph Laplacians; instability in the smallest nonzero eigenvalues directly changes the smooth geometric modes that the loss is intended to transfer from teacher to student. Thus, before fixing the graph batch size for training, we use a spectral stability diagnostic based on the condition number of the graph Laplacian.

For a graph with a symmetric normalised Laplacian given by Equations 6.4 and 6.5, let

$$0 = \lambda_1(\boldsymbol{L}_{sym}) \leq \lambda_2(\boldsymbol{L}_{sym}) \leq \cdots \leq \lambda_N(\boldsymbol{L}_{sym}) \tag{6.44}$$

denote its eigenvalues, where N is the graph batch size being considered. For a connected graph, the diagnostic condition number is

$$\kappa(\boldsymbol{L}_{sym}, M) = \frac{\lambda_M(\boldsymbol{L}_{sym})}{\lambda_2(\boldsymbol{L}_{sym})}. \tag{6.45}$$

where $M \leq N$. This quantity is sensitive to the smallest nonzero eigenvalue, so it provides a compact way to monitor whether the low-frequency part of the graph spectrum has entered a stable finite-sample regime. If the graph is disconnected, then $\lambda_2(\boldsymbol{L}_{sym}) = 0$, and the condition number is not informative; this is precisely the failure mode expected for very small graph samples.

We evaluated this diagnostic by increasing the number of samples used to construct the graph and computing the mean and error bars of $\kappa(\boldsymbol{L}_{sym}, 200)$ across 10

repetitions for each extracted layer. The experimental setting from Section 6.7.1 is used, with varying graph batch sizes. The result is shown in Figure 6.1. For small values of N, the condition number is unstable, and for the smallest sample sizes, it is not always well defined. Around $N \approx 1200$, the curves enter a substantially more stable regime: the sharp changes seen at smaller sample sizes disappear, the error bars shrink, and subsequent increases in N mainly produce gradual changes rather than qualitative changes in the spectrum. This is taken as evidence that the graph has enough samples for the smallest nonzero eigenvalues to behave like stable finite-sample approximations of the corresponding continuum eigenvalues.

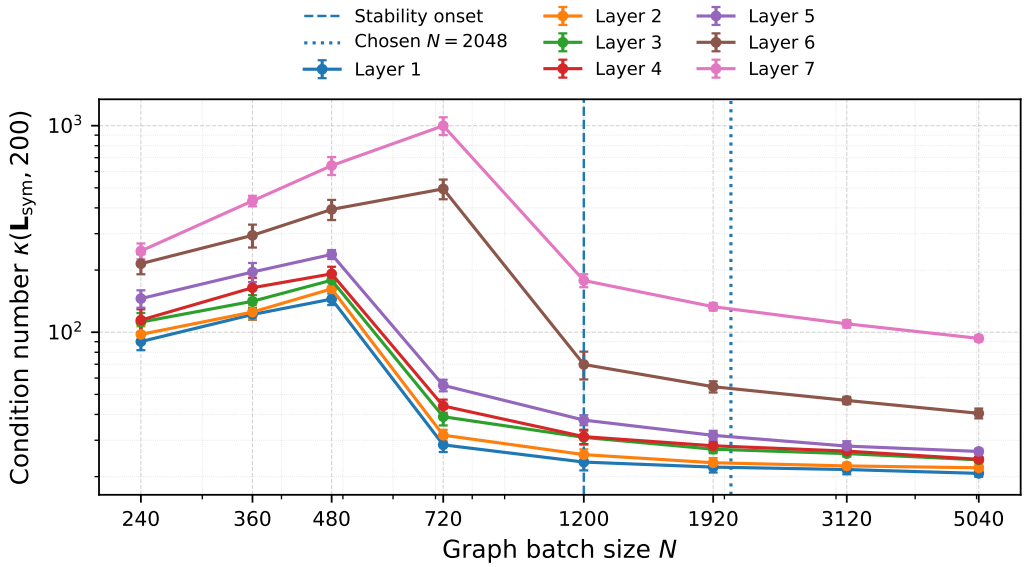

Figure 6.1: Condition number of the normalised graph Laplacian as the graph batch size N is increased. Each curve corresponds to one extracted teacher layer, and error bars show the range of values across all 10 repetitions. The diagnostic becomes substantially more stable around $N \approx 1200$.

Based on this diagnostic, we use a graph batch size of

$$N = 2048. \tag{6.46}$$

This value is above the observed stability threshold and convenient computationally as a power of two. This was also verified empirically, where smaller graph batch sizes degrade distillation performance. Below the stable regime, the graph loss matches noisy finite-sample graph operators rather than reliable approximations to the teacher's representation geometry. The task loss, in contrast, is still optimised with a smaller batch size because the supervised classification objective does not require constructing a stable graph Laplacian on the batch.

6.7 Experimental Evaluation

This section evaluates manifold knowledge distillation (MKD) in two settings. The first is a controlled CIFAR-10 setting designed to test whether the proposed graph-Laplacian loss improves a substantially smaller student network when the teacher and student have an aligned layer structure. The second uses teacher-student pairs from the RepDistiller/CRD benchmark on CIFAR-100, allowing MKD to be compared against established representation-based and relation-based distillation methods such as FitNet, Attention Transfer (AT), Similarity-Preserving distillation (SP), Correlation Congruence (CC), Relational Knowledge Distillation (RKD), and Factor Transfer (FT) [25, 52, 17, 27, 26, 53, 54].

6.7.1 Controlled CIFAR-10 Setting

Most experiments in this chapter are performed in a controlled convolutional CIFAR-10 setting. The teacher and student have the same macro-architecture but different widths. Both networks use two 3×3 convolutional layers followed by max-pooling, dropout, and three fully connected layers. The teacher has channel widths 48 and 96, followed by fully connected widths 256 and 128, and a 10-dimensional output layer. The student has channel widths 12 and 24, followed by fully connected widths 96 and 64, and the same 10-dimensional output layer. The teacher has 962,090 trainable parameters, while the student has 92,850 trainable parameters, giving approximately a $10.4\times$ reduction in parameter count.

The layer layout is deliberately aligned. This makes it possible to compare teacher and student graph geometries at corresponding stages of the forward pass, while still testing a genuine compression problem: the student has substantially less representational capacity and cannot reproduce the teacher merely by using the same hidden widths.

The graph batch size is chosen based on the spectral stability analysis described in the previous section. The results are shown in Table 6.2. The student trained without teacher information obtains 70.46% test accuracy, while the teacher obtains 77.03%. Adding the proposed manifold distillation loss improves the student to 73.39%, a 45% improvement over the baseline. Thus, in the main controlled setting, MKD recovers a substantial part of the teacher-student gap with a student architecture more than 10 times smaller than the teacher.

6.7.2 Comparison on RepDistiller Teacher-Student Pairs

To place MKD in context, additional experiments are conducted on teacher-student pairs from the RepDistiller/CRD benchmark on CIFAR-100 [54]. These experiments test whether the graph-Laplacian objective is competitive with methods that are conceptually close to it. The most relevant comparisons are representation-based and relation-based methods: FitNet transfers intermediate features; AT transfers attention maps; SP transfers pairwise activation similarities; CC transfers correlation

Table 6.1: Training settings for the controlled CIFAR-10 manifold distillation experiments.

Quantity	Value / role
Dataset	CIFAR-10
Epochs	600
Task batch size	64
Graph batch size	2048
Task learning rate	0.004
Distillation learning rate	0.004
Optimiser	SGD, separate task and distillation optimisers
Task loss	Negative log-likelihood
Graph neighbourhood size	$k = 40$
Default graph symmetrisation	union
Default graph mask	hybrid kernel
Number of manifold-distilled layers	all 5 unless otherwise specified
Graph loss weight	λ_{man}; default: 1
Logits KD weight	λ_{logit}; default: 0
Logits KD temperature	$T_{\text{KD}} = 4$
Diffusion loss	default: not used
Diffusion-loss factor	default: 0.5
Trainable kernels	default: used
Kernel regularisation	λ_{reg}; default: 1

Table 6.2: Controlled CIFAR-10 results. Accuracy is reported on the test set.

Model / training method	Test accuracy (%)
Student baseline (no distillation)	70.46
Teacher	77.03
Student with MKD	73.39

structure; RKD transfers pairwise and higher-order relations; and FT transfers factorised feature information [25, 52, 17, 27, 26, 53].

Tables 6.3 and 6.4 report CIFAR-100 top-1 accuracy for the teacher-student pairs considered. Baseline numbers are taken from the CRD benchmark. The MKD row reports the results obtained with the graph-Laplacian distillation method in this chapter.

For same-family teacher-student pairs, MKD is generally competitive with the related representation-based baselines. It gives the best result among the listed related methods for WRN-40-2 → WRN-16-2, where it reaches 74.34%. For WRN-40-2 → WRN-40-1, the MKD run also exceeds the listed baselines.

The different-architecture results are more mixed. MKD remains reasonable for ResNet50 → VGG8, where it reaches 71.62% and is close to AT and RKD. It also

Table 6.3: CIFAR-100 top-1 accuracy (%) on RepDistiller teacher-student pairs with the same architecture family. The baseline results are from the CRD benchmark; MKD denotes the proposed manifold knowledge distillation method. The best result in each column is shown in bold.

Teacher Student	WRN-40-2 WRN-16-2	WRN-40-2 WRN-40-1	ResNet56 ResNet20	ResNet110 ResNet20	ResNet110 ResNet32	ResNet32x4 ResNet8x4	VGG13 VGG8
Teacher	75.61	75.61	72.34	74.31	74.31	79.42	74.64
No KD	73.26	71.98	69.06	69.06	71.14	72.50	70.36
FitNet	73.58	72.24	69.21	68.99	71.06	**73.50**	71.02
AT	74.08	72.77	**70.55**	**70.22**	72.31	73.44	71.43
SP	73.83	72.43	69.67	70.04	**72.69**	72.94	**72.68**
CC	73.56	72.21	69.63	69.48	71.48	72.97	70.71
RKD	73.35	72.22	69.61	69.25	71.82	71.90	71.48
FT	73.25	71.59	69.84	**70.22**	72.37	72.86	70.58
MKD	**74.34**	**73.02**	70.09	69.67	72.21	73.37	72.21

Table 6.4: CIFAR-100 top-1 accuracy (%) on RepDistiller teacher-student pairs with different architecture families. The baseline results are from the CRD benchmark; MKD denotes the proposed manifold knowledge distillation method. The best result in each column is shown in bold.

Teacher Student	VGG13 MobileNetV2	ResNet50 MobileNetV2	ResNet50 VGG8	ResNet32x4 ShuffleNetV1	ResNet32x4 ShuffleNetV2	WRN-40-2 ShuffleNetV1
Teacher	74.64	79.34	79.34	79.42	79.42	75.61
No KD	64.60	64.60	70.36	70.50	71.82	70.50
FitNet	64.14	63.16	70.69	**73.59**	73.54	73.73
AT	59.40	58.58	71.84	71.73	72.73	73.32
SP	**66.30**	**68.08**	**73.34**	73.48	**74.56**	**74.52**
CC	64.86	65.43	70.25	71.14	71.29	71.38
RKD	64.52	64.43	71.50	72.28	73.21	72.21
FT	61.78	60.99	70.29	71.75	72.50	72.03
MKD	58.82	61.34	71.62	70.23	70.45	70.83

slightly improves over the undistilled student for WRN-40-2 → ShuffleNetV1. However, it performs poorly in the MobileNetV2 settings and is below most related methods for the ShuffleNet settings. These results indicate that the current MKD formulation benefits from an aligned layer structure. When the teacher and student architectures differ substantially, direct layerwise graph matching becomes less natural, and additional alignment, projection, or layer-selection mechanisms may be needed.

6.8 Discussion

The method developed in this chapter is best understood as graph-operator/manifold distillation. Standard feature-level distillation asks the student to match teacher co-

ordinates, possibly after a learned projection. RSM-based distillation asks the student to match a teacher relational matrix. MKD instead asks the student to match a normalised graph operator, namely the symmetric graph Laplacian, constructed from the teacher's representation geometry.

The controlled CIFAR-10 experiment shows that this idea is effective in the setting for which it is most directly designed. The student has the same macro-architecture as the teacher but roughly 10.4× fewer parameters. Under this substantial compression, MKD improves test accuracy from 70.46% to 73.39%, closing part of the gap to the 77.03% teacher. This result supports the central premise of the chapter: matching teacher-induced graph geometry can provide a useful training signal beyond ordinary supervised learning.

The spectral batch-size analysis is important for this conclusion. Graph Laplacian losses are only meaningful if the graph computed on a mini-batch is a reasonable finite-sample approximation to the representation geometry. The condition-number study indicates that the relevant graph-Laplacian spectra stabilise around $N \approx 1200$-1500, so the graph batch size $N = 2048$ is chosen to operate in a more stable regime. This also explains why the method uses separate task and distillation passes: supervised training is easier with small mini-batches, whereas graph distillation requires many samples to construct a stable neighbourhood operator.

The RepDistiller comparison gives a more cautious picture. On teacher-student pairs within the same family, MKD is competitive with several representation-based and relation-based KD methods. It outperforms the listed related baselines in the WRN-40-2 $\rightarrow$ WRN-16-2 setting and is close in several other aligned settings. However, on different-architecture pairs, the current formulation is weaker. This suggests that plain layerwise graph matching is most appropriate when teacher and student representations have corresponding stages of processing.

This limitation is conceptually informative. The graph loss assumes that the matched layers represent comparable stages of computation of similar architectures. Different classes of architectures might have different approaches to approximate the underlying function. Future versions of the method may therefore need learned layer matching, projection maps, or types of alignment based on optimal transport or similar techniques across layers before applying the graph-Laplacian loss.

The hybrid student kernel is another practical contribution. The teacher graph is built with a Gaussian affinity, while the student graph uses a Gaussian kernel on ordinary student-supported edges and a heavy-tailed Student-t kernel only on teacher-supported edges missing from the student neighbourhood graph. This selective use of the Student-t kernel gives stronger gradients exactly where the student must pull teacher-neighbouring samples closer, without making all student affinities heavy-tailed. Empirically, using the Student-t kernel globally is less effective, which supports the interpretation that the heavy tail should be used as a targeted correction rather than as the default graph kernel.

Overall, MKD provides a direct route from the representation-geometry theory of the previous chapters to a trainable distillation objective. The teacher defines a graph Laplacian and, optionally, a diffusion geometry over samples; the student is trained so that its own representations induce similar graph operators. The results show that this approach can improve a compressed student in an aligned CIFAR-10 setting and can be competitive with established representation-based KD methods on several CIFAR-100 teacher-student pairs.

7

Outlook

A large body of work has developed methods for comparing layers, analysing representational similarity, and relating learned representations to behaviour. However, most of this work remains fundamentally local to a single layer: it asks how one layer compares to another layer, or which single layer best captures a particular notion of similarity. One of the central motivations of this thesis is that this viewpoint may be incomplete. A neural network is not only a collection of isolated representations, but a sequence of transformations acting on the same data. It is therefore natural to ask whether there is a meaningful representation geometry associated with the network as a whole, rather than with any individual layer alone.

The multi-view and alternating-diffusion constructions developed in this thesis are a first step in this direction. In the experiments presented here, these methods were used primarily to compare multiple networks by fusing several layerwise representation geometries into a single Markov operator. However, the same construction also suggests a standalone notion of a network-level representation for a single neural network. Until now, when a single representation of a network was required for downstream analysis (e.g., probing generalisation), one typically had to choose a layer. This choice often depends on the task, the architecture, empirical heuristics, or the researcher's intuition. The framework developed here offers a different possibility: instead of selecting one layer as representative of the network, one can combine several layerwise geometries into a single operator that summarises the shared sample geometry across depth.

In this sense, the thesis suggests that it may be possible to construct something like a canonical representation of a neural network. This object is not a feature matrix in the usual coordinate-space sense, but an operator on samples. It is a fused diffusion geometry constructed from multiple internal representations. Such an object may be useful beyond the specific benchmarks considered here. It could be used as a network-level descriptor for comparing models, studying training dynamics, analysing architectural differences, selecting checkpoints, probing generalisation, or relating neural networks to biological or human representations. More generally, it raises the possibility that some functionally relevant structure is distributed across depth and is only partially visible from any single layer.

However, several limitations remain. The alternating-diffusion assumptions only hold approximately in the context of neural networks, since consecutive layers are deterministic and strongly dependent rather than conditionally independent views.

Products of many Markov matrices can also degenerate numerically, limiting how many layers can be aggregated directly. Future work could therefore study more stable multi-view fusion schemes, alternative operator normalisations, learned layer weightings, or architectures in which network-level representation geometry can be analysed more systematically.

Beyond this main contribution, the thesis also develops graph-Laplacian and diffusion-based ideas for knowledge distillation and representation transfer. These directions remain relatively open. We hope that the geometric and operator-based perspective developed here will be useful not only in the contexts studied in this thesis, but also in future settings that are not yet apparent.

Bibliography

[1] Yoshua Bengio, Aaron Courville, and Pascal Vincent. "Representation Learning: A Review and New Perspectives". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.8 (2013), pp. 1798–1828.

[2] Charles Fefferman, Sanjoy Mitter, and Hariharan Narayanan. "Testing the Manifold Hypothesis". In: *Journal of the American Mathematical Society* 29.4 (2016), pp. 983–1049.

[3] John M. Lee. *Introduction to Riemannian Manifolds*. 2nd ed. Vol. 176. Graduate Texts in Mathematics. Springer, 2018.

[4] Frank Morgan. *Riemannian Geometry: A Beginner's Guide*. A K Peters, 1993.

[5] Alessio Ansuini et al. "Intrinsic dimension of data representations in deep neural networks". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.

[6] Lucrezia Valeriani et al. "The geometry of hidden representations of large transformer models". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2023.

[7] Ilya Kaufman and Omri Azencot. "Data Representations' Study of Latent Image Manifolds". In: *International Conference on Machine Learning (ICML)*. 2023.

[8] Tao Yu, Huan Long, and John E. Hopcroft. "Curvature-based Comparison of Two Neural Networks". In: *International Conference on Pattern Recognition (ICPR)*. 2018.

[9] Mikhail Belkin and Partha Niyogi. "Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2001.

[10] Mikhail Belkin and Partha Niyogi. "Convergence of Laplacian Eigenmaps". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2006.

[11] Matthias Hein, Jean-Yves Audibert, and Ulrike von Luxburg. "Graph Laplacians and their Convergence on Random Neighborhood Graphs". In: *Journal of Machine Learning Research* (2007).

[12] Ronald R. Coifman and Stéphane Lafon. "Diffusion maps". In: *Applied and Computational Harmonic Analysis* (2006).

[13] David A. Levin and Yuval Peres. *Markov Chains and Mixing Times*. American Mathematical Society, 2017.

[14] Nikolaus Kriegeskorte, Marieke Mur, and Peter A. Bandettini. "Representational similarity analysis – connecting the branches of systems neuroscience". In: *Frontiers in Systems Neuroscience* 2 (2008), p. 4. DOI: 10.3389/neuro.06.004.2008.

[15] Simon Kornblith et al. “Similarity of neural network representations revisited”. In: *Proceedings of the 36th International Conference on Machine Learning*. 2019, pp. 3519–3529.

[16] Max Klabunde et al. “ReSi: A comprehensive benchmark for representational similarity measures”. In: *International Conference on Learning Representations (ICLR)*. 2025.

[17] Frederick Tung and Greg Mori. “Similarity-Preserving Knowledge Distillation”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019.

[18] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2002.

[19] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[20] Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. “Algorithms for Learning Kernels Based on Centered Alignment”. In: *Journal of Machine Learning Research* (2012).

[21] Arthur Gretton et al. “Measuring Statistical Dependence with Hilbert-Schmidt Norms”. In: *Algorithmic Learning Theory*. Vol. 3734. Lecture Notes in Computer Science. Springer, 2005, pp. 63–77.

[22] Gábor J. Székely, Maria L. Rizzo, and Nail K. Bakirov. “Measuring and Testing Dependence by Correlation of Distances”. In: *The Annals of Statistics* (2007).

[23] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. “Distilling the Knowledge in a Neural Network”. In: *arXiv preprint arXiv:1503.02531* (2015).

[24] Jianping Gou et al. “Knowledge Distillation: A Survey”. In: *International Journal of Computer Vision* 129 (2021), pp. 1789–1819. DOI: 10.1007/s11263-021-01453-z.

[25] Adriana Romero et al. “FitNets: Hints for Thin Deep Nets”. In: *International Conference on Learning Representations (ICLR)*. 2015.

[26] Wonpyo Park et al. “Relational Knowledge Distillation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 3967–3976.

[27] Baoyun Peng et al. “Correlation Congruence for Knowledge Distillation”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 5007–5016.

[28] Zikai Zhou et al. “Rethinking Centered Kernel Alignment in Knowledge Distillation”. In: *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence (IJCAI)*. 2024, pp. 5680–5688. DOI: 10.24963/ijcai.2024/628.

[29] Chang Xu, Dacheng Tao, and Chao Xu. “A Survey on Multi-view Learning”. In: *arXiv preprint arXiv:1304.5634* (2013).

[30] Yingming Li, Ming Yang, and Zhongfei Zhang. “A Survey of Multi-View Representation Learning”. In: *IEEE Transactions on Knowledge and Data Engineering* (2019).

[31] Roy R. Lederman and Ronen Talmon. “Learning the Geometry of Common Latent Variables Using Alternating-Diffusion”. In: *Applied and Computational Harmonic Analysis* (2018).

[32] Atharva Khandait and Jan E. Gerken. “From Layers to Networks: Comparing Neural Representations via Diffusion Geometry”. In: *arXiv preprint arXiv:2605.15901* (2026).

[33] Frances Ding, Jean-Stanislas Denain, and Jacob Steinhardt. “Grounding representation similarity with statistical testing”. In: *Advances in Neural Information Processing Systems*. 2021.

[34] Ari S. Morcos, Maithra Raghu, and Samy Bengio. “Insights on representational similarity in neural networks with canonical correlation”. In: *Advances in Neural Information Processing Systems*. 2018. arXiv: 1806.05759.

[35] Maithra Raghu et al. “SVCCA: Singular vector canonical correlation analysis for deep learning dynamics and interpretability”. In: *Advances in Neural Information Processing Systems*. 2017. arXiv: 1706.05806.

[36] William L. Hamilton, Jure Leskovec, and Dan Jurafsky. “Diachronic word embeddings reveal statistical laws of semantic change”. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2016.

[37] Alex H. Williams et al. “Generalized shape metrics on neural representations”. In: *Advances in Neural Information Processing Systems*. 2021.

[38] Yixuan Li et al. “Convergent learning: Do different neural networks learn the same representations?” In: *Proceedings of the 1st International Workshop on Feature Extraction: Modern Questions and Challenges at NIPS 2015*. 2015.

[39] William L. Hamilton, Jure Leskovec, and Dan Jurafsky. “Cultural shift or linguistic drift? Comparing two computational measures of semantic change”. In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. 2016.

[40] Chenxu Wang et al. “Towards understanding the instability of network embedding”. In: *IEEE Transactions on Knowledge and Data Engineering* 34.2 (2022), pp. 927–941.

[41] Anton Tsitsulin et al. “The Shape of Data: Intrinsic Distance for Data Distributions”. In: *International Conference on Learning Representations (ICLR)*. 2020.

[42] Serguei Barannikov et al. “Representation topology divergence: A method for comparing neural network representations”. In: *Proceedings of the 39th International Conference on Machine Learning*. Vol. 162. Proceedings of Machine Learning Research. PMLR, 2022, pp. 1607–1626. arXiv: 2201.00058.

[43] Tongzhou Wang and Phillip Isola. “Understanding contrastive representation learning through alignment and uniformity on the hypersphere”. In: *Proceedings of the 37th International Conference on Machine Learning*. 2020.

[44] Avner May et al. “On the downstream performance of compressed word embeddings”. In: *Advances in Neural Information Processing Systems*. 2019.

[45] Enric Boix-Adsera et al. “GULP: a prediction-based metric between representations”. In: *Advances in Neural Information Processing Systems*. 2022.

[46] Zi Yin and Yuanyuan Shen. “On the dimensionality of word embedding”. In: *Advances in Neural Information Processing Systems*. 2018.

[47] Ori Katz et al. “Alternating diffusion maps for multimodal data fusion”. In: *Information Fusion* 45 (2019), pp. 346–360.

[48] Laurens van der Maaten and Geoffrey Hinton. “Visualizing Data using t-SNE”. In: *Journal of Machine Learning Research* 9.Nov (2008), pp. 2579–2605.

[49] Dmitry Kobak et al. “Heavy-Tailed Kernels Reveal a Finer Cluster Structure in t-SNE Visualisations”. In: *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*. 2019.

[50] Gert R. G. Lanckriet et al. “Learning the Kernel Matrix with Semidefinite Programming”. In: *Journal of Machine Learning Research* 5 (2004), pp. 27–72.

[51] Mehmet Gönen and Ethem Alpaydin. “Multiple Kernel Learning Algorithms”. In: *Journal of Machine Learning Research* (2011).

[52] Sergey Zagoruyko and Nikos Komodakis. “Paying More Attention to Attention: Improving the Performance of Convolutional Neural Networks via Attention Transfer”. In: *International Conference on Learning Representations*. 2017.

[53] Jangho Kim, Seonguk Park, and Nojun Kwak. “Paraphrasing Complex Network: Network Compression via Factor Transfer”. In: *Advances in Neural Information Processing Systems*. 2018.

[54] Yonglong Tian, Dilip Krishnan, and Phillip Isola. “Contrastive Representation Distillation”. In: *International Conference on Learning Representations*. 2020.

A

Appendix 1

A.1 Additional Results for Diffusion-Based Representation Comparison

This appendix contains some additional experimental details and results.

A.1.1 ReSi Implementation Details

In the original ReSi protocol, single-layer measures are primarily evaluated on the final hidden-layer representation, since this representation feeds directly into the classifier; with ten trained seeds per setting, each reported correlation is computed over the 45 unordered model pairs.

For the multi-scale DistCorr variant used in the ReSi experiments, the final score is the weighted combination

$$m_{\text{MS-DistCorr,final}}(\boldsymbol{S}_1, \boldsymbol{S}_2) = \frac{1}{3}\left(m^{(2)}_{\text{MS-DistCorr}}(\boldsymbol{S}_1, \boldsymbol{S}_2) + 2m_{\text{DistCorr}}(\boldsymbol{S}_1, \boldsymbol{S}_2)\right). \quad \text{(A.1)}$$

In the additional tables below, CKA denotes linear CKA, and RBF-CKA denotes CKA computed with an RBF kernel. Asterisks denote statistical significance for ReSi prediction-grounded correlations: $*$ indicates significance at the 5% level and $**$ indicates significance at the 1% level.

A.1.2 Layer Selections

Table A.1 gives the full layer selections used for the multi-layer AD variants. The corresponding table in the main chapter only includes the architectures appearing in the main ReSi summary table; the table below also includes ALBERT, MNLI, ResNet34, ViT-B/32, ViT-L/32, and the vision JSD-output settings. For graph-domain experiments, all extracted representations are used.

A.1.3 ReSi Results

The tables below add the remaining ReSi results.

Table A.1: Representation sets $\mathcal{A}$ used by the multi-layer measures. The indices refer to the representations extracted by ReSi. For language models, the zeroth representation is the embedding plus positional-encoding representation. For graph-domain experiments, all extracted representations are selected.

Domain	Dataset	Target difference	Architecture / representation indices
Language	SST-2, MNLI	Accuracy	BERT base: $[1, 3, 6, 9, 12]$ ALBERT: $[1, 3, 6, 9, 12]$ SmolLM2: $[17, 21, 24]$
		JSD output	BERT base: $[11, 12]$ ALBERT: $[11, 12]$ SmolLM2: $[21, 22, 23, 24]$
Vision	ImageNet100	Accuracy	ResNet18: $[1, 4, 7, 9]$ ResNet34: $[11, 14, 17]$ ResNet101: $[0, 5, 11, 17, 23, 28, 34]$ VGG11: $[4, 6, 8]$ VGG19: $[0, 3, 6, 9, 13, 16]$ ViT-B/32: $[7, 10, 12]$ ViT-L/32: $[7, 10, 12]$
		JSD output	ResNet18: $[8, 9]$ ResNet34: $[15, 16, 17]$ ResNet101: $[30, 31, 32, 33, 34]$ VGG11: $[7, 8]$ VGG19: $[14, 15, 16]$ ViT-B/32: $[11, 12]$ ViT-L/32: $[11, 12]$
Graphs	Cora, Flickr, OGBN-Arxiv	All reported ReSi graph tests	all extracted representations

A.1.3.1 ImageNet100 Accuracy Results

Table A.2 gives the ImageNet100 accuracy-correlation results for some additional architectures. Table A.3 also gives RBF-based rows for the ImageNet100 architectures.

A.1.3.2 Vision Output-Difference Results

Table A.4 reports the ImageNet100 Test 2 results. These results evaluate the correlation to Jensen-Shannon output divergence and hard-prediction disagreement.

A.1.3.3 More Language-Domain Results

Table A.5 reports additional language-domain ReSi results: all MNLI columns, all ALBERT columns, all disagreement columns, and the RBF variants. The SST-2 BERT/SmolLM2 accuracy and JSD columns already shown in the main chapter are omitted here, except where needed for disagreement results.

A.1.3.4 Graph-Domain Results

Table A.6 reports the graph-domain ReSi Test 1 results.

Table A.2: Additional ImageNet100 Test 1 results: correlation to accuracy difference for architectures not included in the main ReSi table. Higher values are better.

Category	Method	ResNet34	ViT-B/32	ViT-L/32
CCA	PWCCA	-0.20	-0.08	0.09
	SVCCA	0.27	-0.01	-0.17
Alignment	AlignCos	-0.35	0.07	0.05
	AngShape	-0.16	0.07	0.06
	HardCorr	0.13	0.35	-0.17
	LinReg	-0.11	0.15	0.05
	OrthProc	-0.16	0.07	0.06
	PermProc	0.09	-0.06	-0.33
	ProcDist	0.00	0.16	0.05
	SoftCorr	0.08	0.36	-0.19
RSM	CKA	-0.07	-0.26	0.05
	RBF-CKA	-0.03	-0.30	0.10
	DistCorr	-0.08	-0.26	0.03
	EOS	-0.17	**0.47**	0.03
	GULP	-0.18	0.18	0.04
	RSA	-0.17	-0.12	-0.11
	RSMDiff	-0.10	0.01	-0.06
Neighbors	2nd-Cos	-0.15	-0.22	0.17
	Jaccard	-0.13	-0.01	0.25
	RankSim	0.04	0.17	**0.35**
Topology	IMD	-0.20	-0.23	-0.19
	RTD	0.12	-0.11	0.10
Statistic	ConcDiff	**0.34**	0.00	0.18
	MagDiff	0.02	0.07	0.15
	UnifDiff	–	–	–
Diffusion-based (Ours)	Multi-Scale CKA	0.09	-0.23	-0.01
	AD-CKA	-0.05	-0.26	-0.22
	Multi-Scale RBF-CKA	-0.01	-0.28	0.03
	AD-RBF-CKA	-0.03	-0.47	-0.08
	Multi-Scale DistCorr	0.08	-0.23	-0.03
	AD-DistCorr	-0.06	-0.28	-0.19

Table A.3: Additional RBF-based ImageNet100 Test 1 rows for the architectures included in the main compact ReSi table. Higher values are better.

Method	ResNet18	ResNet101	VGG11	VGG19
RBF-CKA	0.25	-0.00	-0.03	-0.06
Multi-Scale RBF-CKA	0.25	-0.06	-0.03	-0.08
AD-RBF-CKA	0.14	0.11	0.08	**0.25**

Table A.4: ImageNet100 Test 2 results: correlation to output difference. Higher values are better.

	Test Architecture	JSD correlation ↑							Disagreement correlation ↑						
		RNet18	RNet34	RNet101	VGG11	VGG19	ViT-B/32	ViT-L/32	RNet18	RNet34	RNet101	VGG11	VGG19	ViT-B/32	ViT-L/32
CCA	PWCCA	0.13	0.15	0.15	-0.13	0.15	0.21	-0.24*	0.27**	0.33**	0.06	-0.12	-0.36**	0.02	-0.18
	SVCCA	0.21	-0.00	0.25	-0.11	0.16	0.05	**0.18**	0.39**	0.07	0.14	0.03	0.01	-0.06	0.07
Alignment	AlignCos	0.08	0.05	0.38*	0.10	-0.20	-0.22	-0.06	0.19	**0.50****	0.17	0.16	-0.13	-0.08	0.00
	AngShape	0.24	0.22	0.34*	-0.01	0.19	-0.26	-0.15	0.24	0.40**	0.19	-0.14	-0.33*	-0.11	-0.06
	HardCorr	0.28	0.31*	0.02	-0.22	-0.05	0.03	-0.24	0.28	0.06	-0.04	-0.15	-0.18	0.27	-0.16
	LinReg	0.21*	0.21*	**0.41****	-0.01	0.25*	-0.13	-0.14	0.19	0.25	0.29**	-0.17	-0.24*	0.04	-0.07
	OrthProc	0.24	0.22	0.34*	-0.02	0.19	-0.26	-0.15	0.24	0.40**	0.19	-0.14	-0.33*	-0.11	-0.06
	PermProc	0.18	0.18	0.27	-0.18	0.06	0.36*	0.06	0.13	0.25	-0.04	0.02	0.20	**0.37***	0.10
	ProcDist	0.10	0.14	0.39*	-0.05	0.27	-0.05	0.02	0.08	-0.08	0.11	-0.10	-0.07	-0.07	0.08
	SoftCorr	**0.45****	0.27	0.11	-0.04	-0.16	0.01	-0.31*	0.47**	-0.13	-0.07	-0.03	-0.29	0.27	-0.16
RSM	CKA	0.30*	0.08	0.30*	-0.13	-0.06	0.04	-0.07	0.37*	0.08	0.22	0.01	-0.24	0.00	-0.02
	RBF-CKA	0.17	-0.11	0.28	-0.07	0.20	0.01	-0.24	0.31	-0.13	0.35	-0.01	-0.21	0.02	-0.19
	DistCorr	0.26	0.05	0.31*	-0.10	0.04	0.05	-0.12	0.36*	0.01	0.28	0.00	-0.25	0.02	-0.05
	EOS	0.09	**0.49****	0.33*	-0.11	0.15	-0.18	-0.28	0.11	0.25	0.14	-0.31*	-0.41**	0.01	-0.15
	GULP	0.07	**0.49****	0.35*	-0.05	0.15	-0.05	-0.28	0.10	0.26	0.13	-0.27	-0.41**	0.19	**0.15**
	RSA	0.12	0.18	0.09	-0.18	0.19	-0.11	-0.20	0.19	0.33*	0.10	-0.05	0.04	0.09	-0.04
	RSMDiff	-0.41**	-0.22	0.30*	-0.27	0.07	0.02	-0.28	-0.17	-0.20	0.18	-0.01	-0.03	-0.21	-0.17
Neighbors	2nd-Cos	-0.13	0.16	0.29	0.07	-0.29	0.43*	-0.35*	-0.21	0.45*	0.10	0.11	-0.07	0.19	-0.27
	Jaccard	0.36*	0.26	0.32*	0.05	**0.33***	**0.47****	-0.30*	0.25	0.47**	0.23	0.14	-0.11	0.34*	-0.18
	RankSim	-0.15	-0.00	0.22	0.01	0.05	0.25	-0.09	-0.10	-0.05	0.02	0.01	-0.32*	0.14	-0.33*
Topology	IMD	-0.11	0.08	0.21	**0.20**	0.11	0.41**	0.07	-0.07	0.00	0.26	0.26	0.02	0.23	0.07
	RTD	-0.18	0.02	0.17	-0.02	-0.21	0.20	-0.27	-0.08	-0.06	0.33*	**0.27**	-0.29	0.36*	-0.23
Statistic	ConcDiff	-0.29*	0.24	-0.11	-0.17	-0.13	-0.11	-0.37*	-0.09	0.00	-0.21	-0.11	-0.06	-0.08	-0.29
	MagDiff	-0.38*	-0.20	0.02	-0.16	-0.28	0.02	-0.32*	-0.17	-0.22	-0.05	-0.09	0.04	-0.01	-0.22
	UnifDiff	-0.34*	–	–	0.04	-0.17	–	–	-0.02	–	–	0.17	**0.39****	–	–
Diffusion-based (Ours)	Multi-Scale CKA	0.28	0.06	0.32	-0.14	-0.02	0.10	-0.04	0.42	0.02	0.28	-0.01	-0.26	-0.04	0.02
	AD-CKA	0.33	-0.02	0.34	-0.09	-0.06	0.08	-0.02	**0.54**	0.04	0.32	0.12	0.10	-0.12	-0.03
	Multi-Scale RBF-CKA	0.17	-0.09	0.24	-0.05	0.18	0.15	-0.22	0.32	-0.25	**0.37**	-0.01	-0.21	-0.00	-0.16
	AD-RBF-CKA	0.06	0.24	0.13	0.01	0.03	0.05	-0.16	0.21	0.19	0.14	-0.08	-0.19	-0.08	-0.09
	Multi-Scale DistCorr	0.25	0.01	0.30	-0.09	0.05	0.11	-0.08	0.40	-0.08	0.30	0.00	-0.26	-0.04	-0.01
	AD-DistCorr	0.21	0.14	0.29	-0.09	0.03	0.08	-0.06	0.44	0.08	0.29	0.01	0.12	-0.13	-0.04

Table A.5: Language-domain ReSi results beyond the main table. Higher values are better.

	Dataset	MNLI									SST-2 additional columns				
		Acc.			JSD			Disagr.			Acc.	JSD	Disagr.		
	Test Architecture	BERT	ALBERT	SmolLM2	BERT	ALBERT	SmolLM2	BERT	ALBERT	SmolLM2	ALBERT	ALBERT	BERT	ALBERT	SmolLM2
CCA	PWCCA	0.01	-0.03	0.04	0.22	-0.17	0.53**	-0.37*	-0.36**	0.32**	–	–	-0.22**	–	–
	SVCCA	**0.32***	-0.00	0.17	**0.47****	0.12	0.06	0.00	0.33*	0.08	**0.66****	0.35*	0.49**	0.58**	0.47**
Alignment	AlignCos	0.25	0.00	0.19	0.37*	0.09	**0.66****	-0.16	-0.03	0.49**	0.13	0.77**	0.28**	0.51**	0.37*
	AngShape	0.28	0.12	0.14	0.26	-0.08	0.39**	-0.02	-0.01	0.23	0.34*	0.40**	0.48**	0.51**	0.24
	HardCorr	0.04	0.21	0.01	-0.27	-0.03	0.50**	-0.43**	0.00	0.30*	0.27	0.40**	0.34**	0.56**	0.23
	LinReg	0.18	0.05	-0.01	0.28**	0.01	0.62**	-0.03	-0.13	0.33**	0.23*	0.37**	0.32**	0.37**	0.31**
	OrthProc	0.28	0.12	0.14	0.26	-0.08	0.39**	-0.02	-0.01	0.23	0.34*	0.40**	0.48**	0.51**	0.24
	PermProc	0.09	-0.02	0.15	-0.06	0.04	0.45**	-0.30*	-0.04	0.31*	0.01	**0.79****	0.18*	0.44**	0.42**
	ProcDist	0.28	-0.04	0.17	0.07	-0.00	0.60**	-0.38*	-0.07	**0.43****	0.03	0.76**	0.38**	0.47**	0.52**
	SoftCorr	0.11	0.18	0.07	-0.23	-0.02	0.61**	-0.42**	0.01	0.37*	0.29	0.37*	0.36**	0.56**	0.31*
RSM	CKA	0.18	0.17	0.22	0.30*	0.28	0.08	-0.01	**0.57****	0.12	**0.66****	0.37*	0.51**	0.58**	0.50**
	RBF-CKA	0.07	0.03	0.09	0.27	-0.01	0.36	0.54	0.13	0.29	0.58	0.07	0.12	0.29	0.44
	DistCorr	0.15	**0.25**	0.18	0.39**	**0.32***	0.13	0.12	**0.57****	0.17	0.56**	0.51**	0.53**	0.58**	0.51**
	EOS	0.03	-0.10	-0.11	0.36*	-0.10	-0.30	0.01	-0.16	-0.23	-0.06	0.13	-0.07	-0.05	0.06
	GULP	-0.01	-0.12	0.01	0.35*	-0.10	0.25	0.02	-0.16	0.13	-0.17	0.25	-0.12	-0.13	–
	RSA	0.00	0.18	**0.24**	0.27	0.23	-0.01	0.19	0.47**	0.00	0.43**	0.53**	**0.59****	0.58**	0.15
	RSMDiff	0.30*	-0.15	0.06	-0.18	-0.02	0.35*	-0.19	0.12	0.24	-0.05	0.43**	-0.10	0.28	0.15
Neighbors	2nd-Cos	-0.26	-0.25	0.03	0.16	0.04	-0.07	**0.55****	0.12	0.01	0.24	0.30*	0.15*	0.16	-0.20
	Jaccard	-0.21	-0.25	-0.03	0.13	0.02	0.09	0.17	-0.05	-0.03	0.23	0.32*	0.23**	0.12	0.06
	RankSim	-0.09	-0.27	-0.05	0.08	-0.06	0.08	0.05	-0.13	-0.01	0.12	0.26	0.24**	0.09	0.01
Topology	IMD	-0.26	-0.08	-0.13	-0.39**	-0.29*	0.12	-0.06	-0.16	0.09	0.02	0.22	0.21**	0.08	0.25
	RTD	0.09	-0.27	-0.02	0.04	-0.34*	0.33*	0.10	-0.18	0.12	-0.10	-0.08	0.13	0.03	0.06
Statistic	ConcDiff	-0.00	-0.07	0.01	0.02	-0.07	0.46**	-0.31*	0.07	0.30*	-0.20	0.59**	-0.26**	0.29	0.06
	MagDiff	0.22	-0.06	-0.20	0.01	0.01	-0.39**	-0.03	0.08	-0.46**	0.01	0.10	0.04	0.13	-0.17
	UnifDiff	0.14	-0.16	-0.12	-0.02	-0.30*	-0.05	-0.14	-0.24	0.03	0.27	0.20	-0.01	0.17	-0.01
Diffusion-based (Ours)	Multi-Scale CKA	0.27	0.14	0.22	0.33	0.26	0.07	-0.08	0.53	0.11	**0.66**	0.37	0.42	0.58	0.49
	AD-CKA	-0.09	-0.25	0.02	-0.02	0.24	-0.31	-0.06	0.34	-0.21	0.14	0.12	0.39	0.27	0.45
	Multi-Scale RBF-CKA	0.02	0.12	0.10	0.20	0.04	0.26	0.43	0.17	0.19	0.55	0.03	0.16	0.25	0.48
	AD-RBF-CKA	-0.14	-0.15	0.13	0.05	0.13	0.24	-0.05	0.30	0.24	0.19	0.04	0.37	0.25	0.39
	Multi-Scale DistCorr	0.22	**0.25**	0.16	0.40	0.27	0.06	-0.03	0.56	0.12	0.58	**0.51**	0.44	**0.60**	0.53
	AD-DistCorr	-0.11	-0.23	0.01	0.03	0.19	-0.34	-0.03	0.35	-0.19	0.18	0.11	0.41	0.31	**0.54**

Table A.6: Graph-domain ReSi Test 1 results: correlation to accuracy difference. Higher values are better.

	Dataset	Cora				Flickr			OGBN-Arxiv		
	Architecture	GCN	SAGE	GAT	PGNN	GCN	SAGE	GAT	GCN	SAGE	GAT
CCA	PWCCA	-0.05	0.07	-0.26*	-0.09	–	-0.05	-0.16	-0.09	0.06	-0.30**
	SVCCA	-0.02	-0.13	-0.19	-0.33*	0.01	0.01	-0.18	-0.27	0.10	-0.10
Alignment	AlignCos	-0.33*	0.13	-0.29	-0.08	0.35*	0.24	-0.07	-0.08	0.17	-0.17
	AngShape	0.15	-0.29	-0.12	-0.02	0.39**	0.28	-0.15	-0.04	0.09	-0.09
	HardCorr	0.11	-0.11	-0.14	-0.12	0.31*	0.35*	0.06	**0.37***	0.02	0.04
	LinReg	0.06	-0.21*	-0.11	-0.14	-0.04	0.17	-0.18	0.07	-0.01	-0.19
	OrthProc	0.15	-0.29	-0.12	-0.02	0.39**	0.28	-0.15	-0.04	0.09	-0.09
	PermProc	-0.12	**0.18**	-0.26	**0.29**	0.20	-0.19	0.15	-0.09	0.03	**0.43****
	ProcDist	0.08	0.01	-0.19	**0.29**	0.02	-0.06	0.11	-0.17	0.07	**0.43****
	SoftCorr	**0.18**	-0.05	0.03	-0.02	0.30*	0.33*	-0.07	0.35*	0.12	0.12
RSM	CKA	0.16	-0.17	0.02	-0.08	0.03	0.27	-0.16	-0.17	0.11	-0.05
	RBF-CKA	0.09	-0.23	0.02	–	0.52	0.10	-0.15	-0.14	0.19	-0.13
	DistCorr	0.01	-0.17	0.03	0.17	0.41**	**0.42****	-0.19	-0.10	0.15	-0.06
	EOS	-0.24	0.08	-0.05	-0.04	0.15	-0.27	**0.29**	-0.21	0.05	-0.32*
	GULP	-0.43**	0.08	-0.12	-0.02	0.04	-0.27	-0.27	-0.08	0.06	-0.34*
	RSA	-0.27	0.04	-0.28	0.21	0.53**	0.32*	-0.08	-0.07	**0.25**	0.32*
	RSMDiff	-0.19	0.07	-0.14	0.24	-0.18	-0.16	0.13	-0.05	-0.19	0.02
Neighbors	2nd-Cos	-0.10	0.04	-0.11	0.13	**0.54****	-0.19	0.01	-0.47**	0.22	-0.19
	Jaccard	0.02	-0.11	-0.16	0.07	0.32*	0.28	-0.18	-0.32*	-0.13	-0.14
	RankSim	-0.00	-0.10	0.32*	0.23	0.35*	0.31*	-0.20	-0.28	0.05	-0.09
Topology	IMD	-0.10	0.00	-0.03	0.04	-0.10	0.37*	-0.09	-0.21	-0.02	-0.15
	RTD	0.06	-0.26	**0.13**	-0.44**	0.19	0.13	-0.17	-0.23	0.15	-0.33*
Statistic	ConcDiff	0.15	-0.25	-0.20	0.13	-0.08	-0.29	-0.07	-0.07	-0.13	-0.12
	MagDiff	0.08	-0.13	-0.19	0.18	0.02	-0.17	0.14	-0.18	-0.20	0.11
	UnifDiff	-0.10	-0.04	-0.09	-0.13	-0.18	0.03	-0.18	-0.19	-0.20	-0.25
Diffusion-based (Ours)	Multi-Scale CKA	0.09	-0.20	-0.02	–	0.00	0.20	-0.17	-0.20	0.10	-0.04
	AD-CKA	0.06	-0.10	0.03	–	0.03	0.01	-0.12	-0.16	-0.00	-0.08
	Multi-Scale RBF-CKA	0.07	-0.29	0.02	–	0.52	0.03	-0.15	-0.09	0.15	-0.15
	AD-RBF-CKA	0.08	-0.25	0.02	–	0.42	-0.04	-0.19	-0.02	-0.04	-0.06
	Multi-Scale DistCorr	0.06	-0.20	-0.01	–	0.32	0.36	-0.21	-0.09	0.14	-0.00
	AD-DistCorr	0.04	-0.18	0.03	–	0.51	0.14	-0.14	-0.05	-0.07	0.01

A.1.3.5 Results for tests grounded by design

Table A.7 reports the columns for other ReSi tests (grounded by design). The accuracy and JSD columns from that table are omitted here because those prediction-grounded quantities are already represented in the main chapter or in the additional ReSi tables above. The test names follow the ReSi benchmark terminology; see the ReSi benchmark description for the detailed definitions of label-randomisation, shortcut, and augmentation tests [16].

A.1.4 Compute Resources

All experiments were run on a compute cluster, primarily using a single NVIDIA A40 or a single NVIDIA V100 GPU at a time. The computational requirements are modest, and the experiments can also be executed on a single T4 GPU. Thus, the SoTA gains reported above do not rely on unusually large computational budgets; they arise from changing the geometric object used for comparison.

Table A.7: Design-grounded ReSi results. Higher values indicate better adherence to the corresponding similarity grounding.

	Domain	Graphs			Language		Vision	
	Setting	Flickr, GraphSAGE			SST-2, BERT		IN100, ResNet18	
	Test	Label rand.	Shortcut	Aug.	Label rand.	Shortcut	Shortcut	Aug.
CCA	PWCCA	0.44	0.43	0.57	0.27	0.32	0.99	0.90
	SVCCA	0.80	0.93	0.67	0.64	0.36	0.55	0.40
Alignment	AlignCos	0.42	**1.00**	0.70	**0.99**	0.54	**1.00**	0.71
	AngShape	0.43	**1.00**	0.76	0.49	0.43	**1.00**	0.71
	HardCorr	0.46	**1.00**	0.72	0.44	0.36	0.97	0.46
	LinReg	0.45	0.61	0.81	0.40	0.46	0.99	0.94
	OrthProc	0.43	**1.00**	0.76	0.49	0.43	**1.00**	0.71
	PermProc	0.90	**1.00**	0.69	0.45	0.55	0.72	0.41
	ProcDist	0.62	**1.00**	0.81	0.86	0.52	**1.00**	0.58
	SoftCorr	0.45	**1.00**	0.58	0.48	0.34	0.97	0.45
RSM	CKA	0.66	**1.00**	0.75	0.59	0.38	**1.00**	0.90
	RBF-CKA	0.46	**1.00**	0.96	0.53	**1.00**	**1.00**	0.65
	DistCorr	0.43	**1.00**	0.79	0.59	0.39	**1.00**	0.83
	EOS	0.42	0.43	0.53	0.36	0.33	**1.00**	0.93
	GULP	0.42	0.43	0.54	0.28	0.30	**1.00**	0.92
	RSA	0.42	**1.00**	0.98	0.48	0.47	**1.00**	**0.98**
	RSMDiff	0.92	0.92	0.93	0.91	0.37	0.57	0.45
Neighbors	2nd-Cos	0.42	**1.00**	0.92	0.37	**0.64**	**1.00**	0.78
	Jaccard	0.43	0.83	0.88	0.35	**0.64**	**1.00**	0.79
	RankSim	0.43	0.77	0.88	0.34	**0.64**	0.99	0.71
Topology	IMD	0.37	0.97	0.57	0.47	0.34	0.67	0.56
	RTD	0.59	**1.00**	**1.00**	0.27	0.39	**1.00**	0.57
Statistic	ConcDiff	0.57	0.18	0.35	0.96	0.27	0.53	0.43
	MagDiff	0.72	0.78	0.17	0.38	0.35	0.37	0.37
	UnifDiff	0.90	0.50	0.24	0.60	0.37	0.75	0.17
Diffusion-based (Ours)	Multi-Scale CKA	0.74	**1.00**	0.76	0.61	**1.00**	**1.00**	0.74
	AD-CKA	**0.95**	0.96	0.75	0.62	0.99	0.91	0.66
	Multi-Scale RBF-CKA	0.51	**1.00**	0.96	0.53	0.99	0.97	0.65
	AD-RBF-CKA	0.65	**1.00**	**1.00**	0.58	0.96	0.96	0.64
	Multi-Scale DistCorr	0.43	**1.00**	0.78	0.60	**1.00**	0.99	0.60
	AD-DistCorr	0.72	**1.00**	0.77	0.62	**1.00**	0.92	0.64

DEPARTMENT OF MATHEMATICAL SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY