



Stockholms
universitet

Scrutinizing the Training Dynamics of Neural Network Representations: An Approach from Graph-based Rep- resentation Dissimilarity

Isaac Zúñiga Arias

Scrutinizing the Training Dynamics of Neural Network Representations: An Approach from Graph-based Representation Dissimilarity

Isaac Zúñiga Arias*

September 2026

Abstract

Deep neural networks achieve remarkable predictive performance, yet understanding how their internal representations evolve throughout training remains a fundamental challenge. This thesis investigates theoretically and empirically the *Shape-Aware Graph Distance* (SAGD), a graph-based framework for comparing neural network representations through their intrinsic geometric structure. Representations are modeled as weighted graphs, whose pairwise Commute Time Distances are summarized by empirical cumulative distribution functions and compared using the 1-Wasserstein distance. This construction provides a global measure of representational dissimilarity that is naturally invariant to permutations of node labels and captures structural relationships beyond standard Euclidean comparisons.

The theoretical framework combines spectral graph theory, random walks, and 1-Wasserstein distance to establish the mathematical foundations of SAGD and to analyze its principal properties. Robustness is evaluated through controlled experiments involving synthetic Gaussian data, high-dimensional Gaussian mixtures, and Watts–Strogatz small-world networks, demonstrating that the proposed methodology is stable under moderate variations in graph construction and stochastic perturbations.

Finally, SAGD is applied to study the evolution of internal representations in a ResNet-18 model trained on the CIFAR-10 dataset. By combining SAGD with the graph-based visualization method; Shape-Aware Stochastic Neighbor Embedding (SASNE), the analysis reveals distinct geometric trajectories across network layers and training epochs. The results show that representations undergo rapid structural reorganization during the early stages of optimization, followed by gradual refinement, and continue to evolve even after predictive performance has largely converged. Overall, this work presents a graph-theoretic perspective on representation learning and provides a mathematically grounded framework for analyzing the geometry and dynamics of neural network representations.

*Postal address: Mathematical Statistics, Stockholm University, SE-106 91, Sweden.
E-mail: isaac.actuario@gmail.com. Supervisor: Chun-Biu Li.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Chun-Biu Li, for his guidance, support, and encouragement throughout this project. His supervision consistently emphasized both my academic development and personal growth, and he was always generous with his time for discussions and feedback. I would also like to thank Nik Tavakolian for the many insightful discussions and for providing the trained parameters of a CIFAR-10 classifier, which were used to construct the representations employed in the main experiments presented in this thesis.

Finally, I would like to acknowledge my family for their constant emotional support, patience, and encouragement throughout this process. Despite the distance and the time spent apart while pursuing my studies abroad, their support remained a continuous source of motivation and strength during the completion of this work.

I also note that, while the conceptual content and the original drafts of all figures are based on my own work and notes, AI-based tools were used to assist with coding and implementation in the generation of several figures presented in this thesis.

Contents

1	Introduction	3
2	Preliminaries	6
2.1	Neural Networks	6
2.1.1	Optimization	9
2.1.2	Convolutional Neural Networks	11
2.2	Representations	16
3	Theory and Methods	18
3.1	Spectral Graph Theory	18
3.2	The Commute Time Distance	20
3.2.1	Random Walk	20
3.2.2	The Embedding	21
3.3	Shape-Aware Graph Distance	25
3.3.1	1-Wasserstein Distance	25
3.3.2	Definition and Properties	27
3.3.3	Normalization	30
3.3.4	Representational Invariances	31
3.4	Shape-Aware Stochastic Neighbor Embedding	35
4	Results	39
4.1	Graph Construction	40
4.1.1	Synthetic Gaussian	43
4.1.2	High Dimensional Mixture	46
4.2	Small-World Model	49
4.3	Analysis of Training Dynamics	51
5	Conclusion	66
	Appendix	69
A	Proofs of Auxiliary Results	70
B	Additional Figures	76

Chapter 1

Introduction

The remarkable success of deep learning across a wide range of applications has motivated increasing interest in understanding not only why these models perform well, but also how they learn. Rather than viewing a neural network solely as a function that maps inputs to outputs, modern research increasingly studies the internal representations developed throughout training [1]. These representations describe how the network organizes data within successive feature spaces and provide a framework for investigating learning dynamics, comparing different architectures, and characterizing the evolution of internal geometry during optimization [2, 3].

Several mathematical approaches have been proposed to quantify representation similarity. Methods such as Singular Vector Canonical Correlation Analysis (SVCCA) have been successfully employed to study the evolution of deep neural networks throughout training [2], while other approaches have investigated the geometry of neural network error surfaces and the role of parameter symmetries [4]. Despite these developments, comparing representations remains a challenging problem. Internal representations are inherently high-dimensional. Furthermore, methods that rely strictly on Euclidean distance metrics within the high-dimensional activation space may fail to capture the geometry of the non-linear manifolds on which neural network representations naturally lie [5].

To address these limitations, this thesis adopts a graph-based perspective in which representations are viewed not simply as collections of data points but as geometric structures. Spectral graph theory provides a natural framework for studying such structures through the spectral properties of the graph Laplacian [6]. Intuitively, the graph Laplacian is a matrix that acts as a structural summary of the network, mathematically capturing how nodes are connected and how information diffuses across them. In particular, the Commute Time Distance (CTD), derived from random walks on weighted graphs, defines an intrinsic measure of proximity that incorporates the global connectivity of the underlying graph rather than only local Euclidean relationships [7, 8, 9].

Building upon these ideas, this thesis investigates and applies the Shape-Aware Graph Distance (SAGD), a graph-to-graph dissimilarity measure for comparing data representa-

tions. The employed methodology summarizes the collection of pairwise CTDs between all nodes in the graph—which correspond to the individual data points embedded in the layer’s activation space—through their empirical cumulative distribution functions (eCDFs) and measures the discrepancy between two representations using the 1-Wasserstein distance [10, 11]. Consequently, SAGD provides a statistical characterization of graph structure while remaining invariant to permutations of node labels on the graph of the underlying representation.

The motivation for adopting a graph-based approach is illustrated in Figure 1.1. Here, data points are assumed to lie on a non-linear manifold embedded in a higher-dimensional ambient space. Although two observations may appear close when measured by a straight Euclidean path, this distance may ignore the intrinsic geometry of the manifold connecting them. By instead constructing a graph representation and studying distances induced by its topology, one can capture structural relationships that better reflect the organization of the data.

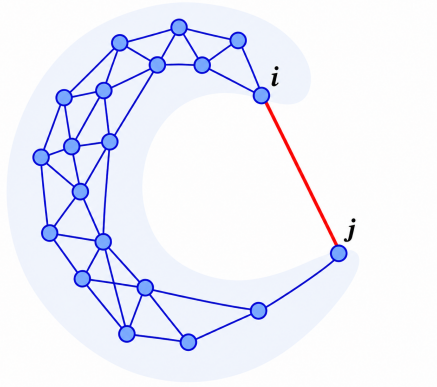


Figure 1.1: kNN Graph Embedding on a Low-Dimensional Manifold. The figure illustrates a graph structure where nodes (blue circles) and edges (blue lines) are embedded within a low-dimensional manifold, depicted as a curved, shaded blue region. A straight green line connects nodes i and j , traversing the ambient white space outside the manifold structure. Euclidean distance (the orange line) fails to capture the true geometric relationship between distant points on a non-linear manifold, highlighting the necessity of intrinsic graph-based distances that respect the data’s underlying topology.

Beyond describing a graph-to-graph dissimilarity function, an important objective of this thesis is to investigate its empirical properties. The mathematical framework underlying SAGD is developed through spectral graph theory, the properties of the CTD, and the construction of graph-level comparisons based on its intrinsic geometry. In addition, the method is analyzed from the perspective of representation similarity, emphasizing desirable invariance properties and its suitability for comparing neural network representations. Since any practical representation metric must also be robust to variations arising from graph construction and finite sampling, controlled experiments are conducted to evaluate the stability of SAGD under changes in graph hyperparameters, structural

randomness, and increasing ambient dimensionality.

After establishing these empirical properties, SAGD is applied to analyze the learning dynamics of a ResNet-18 architecture [12]. The objective is to study how internal representations evolve across layers and training epochs, and to explore whether these dynamics exhibit structured patterns discussed in the literature—such as the bottom-up learning hypothesis, whereby shallower layers are thought to converge before deeper ones [2]. To visualize these representational trajectories, the analysis employs Shape-Aware Stochastic Neighbor Embedding (SASNE) [13], a shape-aware extension of embedding methods such as SNE [14] and t-SNE [15] that replaces Euclidean distances with graph-based distances.

The remainder of this thesis is organized as follows. Chapter 2 introduces the necessary preliminaries on neural networks and representations before developing the theoretical framework underlying spectral graph theory, the Commute Time Distance, the Shape-Aware Graph Distance, its normalization and invariance properties, and the SASNE visualization methodology. Chapter 3 presents the empirical analysis, beginning with robustness experiments designed to evaluate the stability of SAGD under different sources of variability, followed by its application to the study of neural network representation dynamics throughout training. Finally, Chapter 4 discusses the main findings of the proposed framework.

Chapter 2

Preliminaries

2.1. Neural Networks

The main goal for machine learning can be broadly separated into two objectives: classification and regression. For either of these tasks practitioners study the consequential relationship between independent and dependent variables. Let x_i and y_i be the independent and dependent variables respectively, for $i \in \{1, \dots, n\}$. Denote by $\mathcal{X} \in \mathbb{R}^{n \times d}$ the matrix of samples with rows x_i and $\mathcal{Y} \in \mathbb{R}^{n \times D}$ the matrix with rows y_i . This is considered a consequential relationship since the goal is to understand y_i 's behavior given x_i . This “behavior” is expressed as an unknown function f^* such that $y_i = f^*(x_i)$.

Artificial neural networks provide a highly effective framework for capturing this underlying behavior. It is essentially a model that defines a mapping $f(\mathbf{x}, \theta)$ such that;

$$y_i = f^*(x_i) \approx f(x_i, \theta). \quad (2.1)$$

Here, θ represents the parameters to be learned in order for f to be a function approximation of f^* .

Neural networks are called “networks” because of their composing nature.

$$f(x_i, \theta) = f^{(H)}(f^{(H-1)}(\dots f^{(1)}(x_i) \dots)) \quad (2.2)$$

Each function $f^{(h)}$ represents a layer, where $f^{(1)}$ is the first layer and $f^{(H)}$ is the last layer of the network, specifically called the *output* layer. The space of independent variables, $\mathcal{X}$, is called the *input* layer. All layers $h \in \{1, \dots, H-1\}$ are called the *hidden* layers and the length of the chain (H) is referred to as the *depth* of the model.

On the other hand, a *neuron* is a function r at the most fundamental level of the network. For any input $x \in \mathbb{R}^d$, the neuron is parameterized by a weight vector $\mathbf{w} \in \mathbb{R}^d$, a scalar $b \in \mathbb{R}$ called a *bias*, and applies a non-linear *activation* function $\sigma(\cdot)$ such that;

$$r(x) = \sigma(\mathbf{w}^\top x + b). \quad (2.3)$$

Each layer is constituted by a certain amount of neurons called the *width*. Let us assume that the width for each layer h is the value $J^{(h)} \in \mathbb{N}_{>0}$. Then, each neuron in that layer is identified by the double index $r_j^{(h)}$ where $j \in \{1, \dots, J^{(h)}\}$. For any input $x \in \mathbb{R}^{J^{(h-1)}}$, the layer h outputs the vector;

$$f^{(h)}(x) = \left(r_1^{(h)}(x), \dots, r_{J^{(h)}}^{(h)}(x)\right) = \mathbf{r}_h \in \mathbb{R}^{J^{(h)}}. \quad (2.4)$$

The width is of great importance for the rest of the thesis since it establishes the dimension of the output vector $\mathbf{r}_h$. The vector $\mathbf{r}_h$ can be interpreted as a representation of the input x . The layer essentially has modified x in a non-linear way mapping its value to a space with a different dimensionality.

Another fundamental concept in a neural network is the set of *connections*. The connections determine how information is transmitted between neurons across different layers. Each connection is associated with a numerical weight $\mathbf{w}$, which controls the influence that one neuron has on another. When a neuron receives inputs from connected neurons, it combines these incoming signals through weighted sums before applying an activation function to produce an output. In this way, connections regulate the flow of information throughout the network and define how signals propagate from the input layer to the output layer.

In Figure 2.1 a Feedforward neural network is fully described for a classification task of the Iris dataset¹. The problem is computationally simple since it isn't necessary to increase the depth and width of the model to get a clean classification of the sample. Nonetheless, it exemplifies the structure of an implemented network and the dimensional expansion produced in the hidden layer. As part of its *architecture*, each single observation is originally in $\mathbb{R}^4$ then embedded in a $\mathbb{R}^8$ space before the classification.

Designing the architecture of a neural network essentially means deciding the depth, the width for each layer, the activation functions, and the connections. Different neural network architectures use different connection patterns:

- fully connected networks link every neuron between consecutive layers,
- as will be discussed in the following section, convolutional neural networks employ sparse local connections between neurons [17],
- and residual networks include additional pathways that allow signals to bypass certain layers, [12].

¹Classic dataset in statistics and machine learning, introduced by Ronald A. Fisher in 1936. It contains 150 samples of iris flowers from three species—Iris setosa, Iris versicolor, and Iris virginica. Each sample is described by four numerical features: sepal length, sepal width, petal length, and petal width [16]. For the rest of the thesis a fictional dataset of iris images will be used for explanations as an analogy.

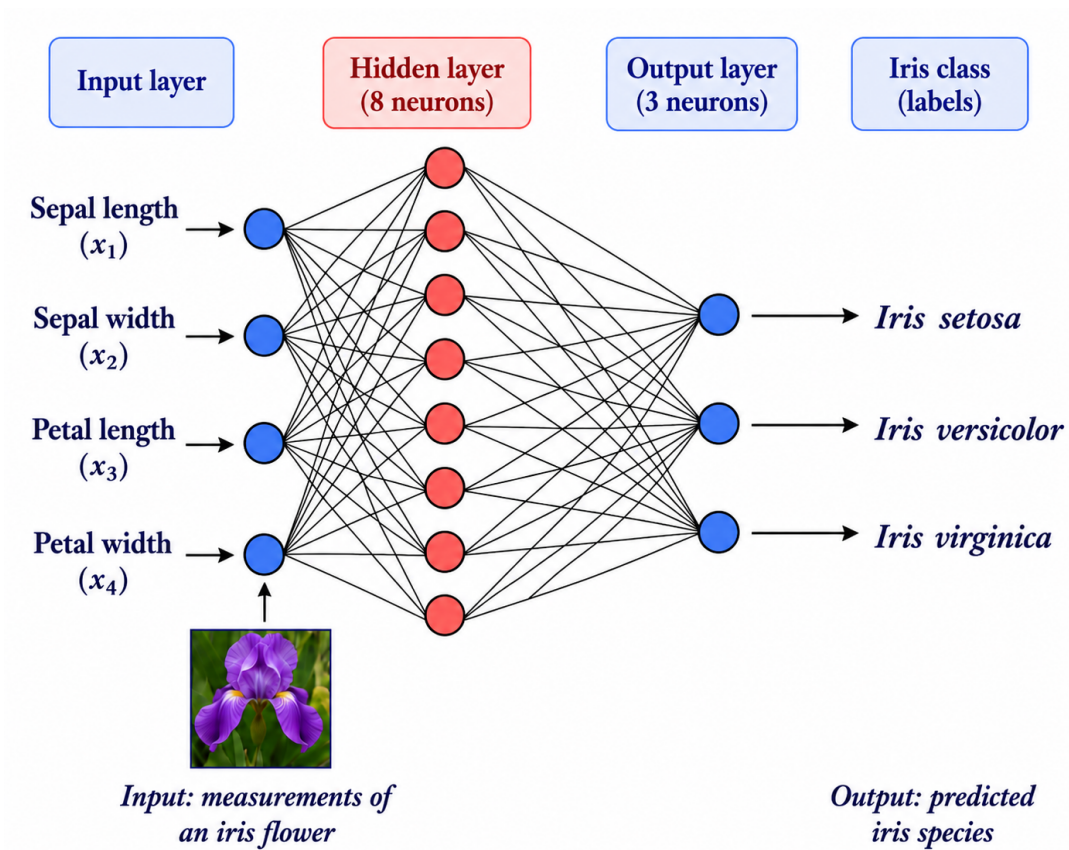


Figure 2.1: **Feedforward neural network architecture for Iris classification.** Illustration of a fully connected feedforward neural network with architecture $4 \rightarrow 8 \rightarrow 3$ used for the classification of samples from the Iris dataset. The input layer receives four botanical measurements of an iris flower: sepal length, sepal width, petal length, and petal width. These features are transformed by a hidden layer composed of eight neurons, generating an intermediate representation $\mathbf{r}_1 \in \mathbb{R}^8$. This hidden representation encodes non-linear combinations of the original measurements and can be interpreted as a learned feature space where the geometry of the data becomes more suitable for classification. Finally, the output layer maps the hidden representation into three neurons associated with the species labels *Iris setosa*, *Iris versicolor*, and *Iris virginica*. The fully connected structure illustrates how information propagates through weighted connections from the input space toward increasingly abstract internal representations before producing a prediction.

Consequently, the structure of these factors largely determines the computational behavior of the network. The output layer’s activation function is task-specific. Its value depends on the purpose of the network such as probability distributions over classes in classification tasks or continuous-valued predictions in regression problems.

In general terms, for a neural network $f(x, \theta)$ the parameter θ is then the set of all weights and biases in each neuron for every layer. In the next section, the necessary concepts for finding parameters θ that guarantee an approximation of f^* will be discussed.

2.1.1. Optimization

The performance of a neural network depends on finding parameters θ such that the model approximation $f(x, \theta)$ captures the behavior of the unknown function f^* . This process is known as *optimization* and consists of iteratively adjusting the parameters of the network in order to minimize a measure of prediction error. In practice, optimization is performed over a finite dataset of observations $\mathcal{X}$ and responses $\mathcal{Y}$ and relies on defining a numerical criterion that evaluates how well the network predictions agree with the target outputs, [17].

The discrepancy between the predicted output and the expected target is quantified through a *loss function*. Let $\hat{y}_i = f(x_i, \theta)$ be the prediction produced by the network for the input x_i . A loss function $\mathcal{L}(y_i, \hat{y}_i)$ assigns a numerical penalty to the prediction error. The optimization problem can then be expressed as minimizing the empirical risk over the dataset,

$$\mathcal{J}(\theta) = \frac{1}{n} \sum_{i=1}^n \mathcal{L}(y_i, f(x_i, \theta)). \quad (2.5)$$

The choice of the loss function depends on the task being solved. In regression problems, the mean squared error is commonly employed,

$$\mathcal{L}_{\text{MSE}}(y, \hat{y}) = |y - \hat{y}|_2^2, \quad (2.6)$$

while classification problems frequently use the cross-entropy loss due to its probabilistic interpretation. In both cases, minimizing the loss encourages the network outputs to approximate the target values more accurately.

In supervised learning, the dataset $\mathcal{X}$ is commonly partitioned into two disjoint subsets: a training set $\mathcal{X}_{\text{train}}$ and a test set $\mathcal{X}_{\text{test}}$. The training set is used during optimization to estimate the parameters θ of the neural network, while the test set remains unseen throughout training and is employed to evaluate the ability of the model to generalize to new observations. More precisely, the optimization procedure minimizes the empirical risk over $\mathcal{X}_{\text{train}}$, and the resulting parameter updates are sequentially computed. After training, the learned function $f(x, \theta)$ is evaluated on $\mathcal{X}_{\text{test}}$ in order to measure predictive

performance on unseen data.

The value of the loss during training is commonly monitored through a *training curve*, which represents the evolution of the objective function across optimization iterations or epochs. A decreasing loss curve generally indicates that the network is learning meaningful representations from the data. However, excessively low training loss may also indicate overfitting if the model fails to generalize to unseen data.

Training a neural network requires computing how changes in the parameters affect the loss function. This is achieved through the *backpropagation* algorithm, which efficiently evaluates the gradients of the objective function with respect to every parameter in the network.

Given the compositional structure of neural networks, backpropagation applies the chain rule of differential calculus recursively from the output layer toward the input layer. For a parameter $\theta_k \in \theta$, the gradient is expressed as,

$$\frac{\partial \mathcal{J}}{\partial \theta_k}. \quad (2.7)$$

These gradients quantify the sensitivity of the loss with respect to the parameters and determine the direction in which the parameters should be modified to reduce prediction error. Once the gradients are computed, the parameters are updated through an optimization algorithm. The most fundamental method is *gradient descent*, where the parameters are iteratively adjusted in the opposite direction of the gradient,

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla_{\theta} \mathcal{J}(\theta^{(t)}), \quad (2.8)$$

where $\eta > 0$ is the *learning rate*.

In practice, evaluating the gradient using the entire dataset at every iteration can be computationally expensive. Therefore, neural networks are commonly optimized using *stochastic gradient descent* (SGD), which approximates the gradient using small subsets of the data called *mini-batches*. An *epoch* corresponds to a complete pass of the entire training dataset through the network during the learning process. During one epoch, the optimization algorithm sequentially processes all mini-batches and updates the network parameters after each batch. Consequently, multiple parameter updates may occur within a single epoch when SGD is employed.

Deep neural networks often contain a large number of parameters, allowing them to represent highly complex functions. While this expressive power is advantageous, it also increases the risk of overfitting. To avoid this other methods were needed like; *regularization*.

Regularization refers to techniques designed to improve generalization by constraining the learning process. A common strategy is to penalize large parameter values by adding a regularization term to the objective function,

$$\mathcal{J}_{\text{reg}}(\theta) = \mathcal{J}(\theta) + \lambda\Omega(\theta), \quad (2.9)$$

where $\lambda > 0$ controls the strength of the penalty and $\Omega(\theta)$ is a regularization function. Typical choices include the ℓ_2 norm,

$$\Omega(\theta) = \|\theta\|_2^2, \quad (2.10)$$

which encourages smaller parameter magnitudes and promotes smoother solutions. Regularization methods help stabilize optimization and reduce the sensitivity of the network to noise in the training data.

A particularly effective regularization technique for deep learning is *dropout*. During training, dropout randomly deactivates a subset of neurons with a fixed probability p , temporarily removing them and their associated connections from the network. Consequently, the model cannot rely excessively on specific neurons and is forced to distribute learned representations across multiple units.

As the network parameters are updated, the distributions of intermediate activations may change significantly, making optimization unstable and slowing convergence. *Batch normalization* addresses this by normalizing the activations of a mini-batch so they approximately have zero mean and unit variance before passing them to the next layer. Intuitively, this prevents some neurons from producing excessively large or small values and allows information to propagate through the network in a more balanced way.

2.1.2. Convolutional Neural Networks

Although fully connected neural networks provide a general framework for approximating complex functions, their direct application to image data becomes computationally inefficient. Images possess an inherent spatial organization: neighboring pixels are strongly correlated and relevant visual patterns are typically local rather than global. Fully connected architectures ignore this geometric structure, leading to a large number of learnable parameters and poor scalability for high-dimensional inputs.

Convolutional Neural Networks (CNNs) address these limitations by introducing sparse local connectivity and parameter sharing [17]. Instead of connecting every neuron to all activations from the previous layer, CNNs apply local transformations through small learnable filters. This allows the network to efficiently capture spatial patterns such as edges, textures, and shapes while preserving the geometric structure of the data.

The fundamental operation of a CNN is the discrete convolution. Let $X \in \mathbb{R}^{H \times W}$ denote an input image with height H and width W , and let $K \in \mathbb{R}^{m \times m}$ denote a learnable kernel of size $m \times m$. The convolution between X and K is defined as

$$(X * K)(i, j) = \sum_{u=1}^m \sum_{v=1}^m X(i + u, j + v) K(u, v). \quad (2.11)$$

The kernel slides across the image and computes weighted local interactions between the filter and neighboring pixels. The resulting matrix is called a *feature map* as described in Figure 2.2. Each kernel specializes in detecting particular local structures in the data, and its parameters are optimized jointly with the rest of the network through backpropagation.

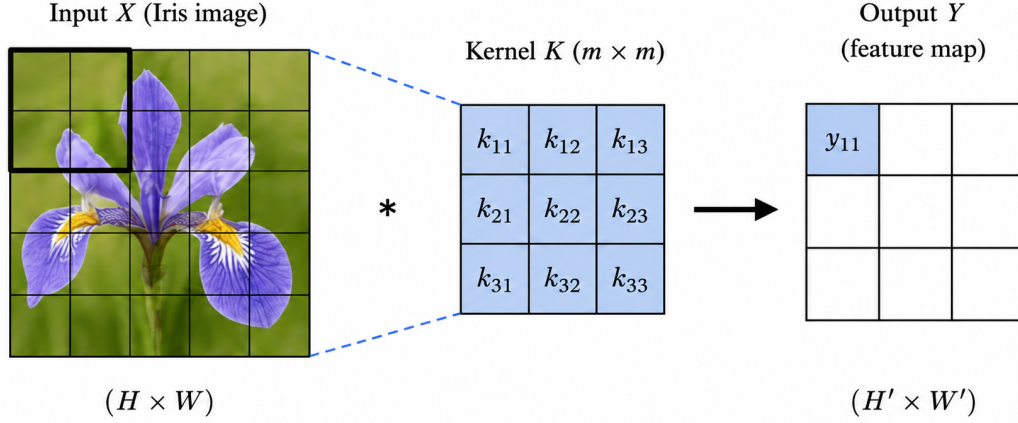


Figure 2.2: **Illustration of the discrete convolution operation in a convolutional neural network.** A local region of the input image $X \in \mathbb{R}^{H \times W}$ is combined elementwise with a learnable kernel $K \in \mathbb{R}^{m \times m}$ to compute a single activation in the output feature map $Y \in \mathbb{R}^{H' \times W'}$. The kernel slides across the spatial dimensions of the image, allowing the network to detect local visual patterns such as edges, textures, and shapes while preserving the geometric structure of the data.

A defining property of convolutional layers is *parameter sharing*. The same kernel weights are reused across all spatial locations of the image. Consequently, the number of learnable parameters is drastically reduced compared to fully connected architectures. More importantly, parameter sharing introduces translational equivariance: if a visual pattern changes position within the image, the convolutional layer can still detect it using the same filter.

Convolutional layers are commonly followed by non-linear activation functions and pooling operations. Pooling reduces the spatial dimensionality of the feature maps while preserving the most relevant information. A standard example is max-pooling, defined as

$$y_{ij} = \max_{(u,v) \in \mathcal{N}_{ij}} x_{uv}, \quad (2.12)$$

where $\mathcal{N}_{ij}$ denotes a local neighborhood around the spatial position (i, j) . Another common operation is average pooling, where the activations inside the neighborhood are averaged

instead of selecting the maximum value. Formally, average pooling is given by

$$y_{ij} = \frac{1}{|\mathcal{N}_{ij}|} \sum_{(u,v) \in \mathcal{N}_{ij}} x_{uv}, \quad (2.13)$$

where $|\mathcal{N}_{ij}|$ denotes the number of elements contained in the neighborhood. Both approaches are conveniently exemplified in Figure 2.3. While max-pooling emphasizes the strongest local activations, average pooling preserves smoother aggregate information across the feature map. Pooling operations improve computational efficiency, enlarge the effective receptive field of deeper neurons, and reduce sensitivity to small perturbations in the input.

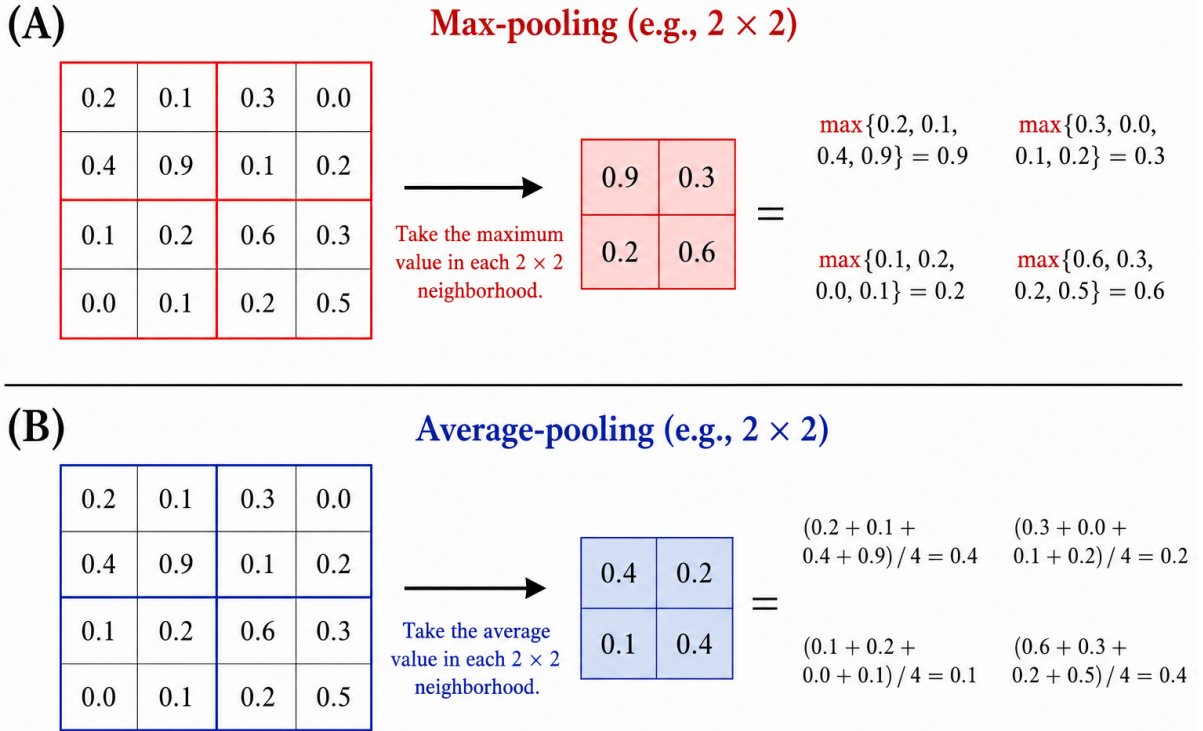


Figure 2.3: Illustration of max-pooling and average-pooling operations using 2×2 neighborhoods. In panel (A), max-pooling; the largest activation within each local region is selected, preserving the strongest feature responses. In average-pooling panel (B), activations within each neighborhood are averaged and rounded, producing a smoother aggregated representation. Both operations reduce the spatial dimensionality of the feature map while retaining relevant local information.

By composing multiple convolutional and pooling layers, CNNs progressively extract increasingly complex features from the data. Early layers typically detect simple geometric structures such as edges and textures, while deeper layers combine these local patterns into more abstract visual features. Consequently, CNNs construct a hierarchy of non-linear transformations adapted to the spatial structure of image data.

Convolutional neural networks have become the dominant paradigm for image recognition problems [17]. In this thesis, the analyzed architecture is a ResNet-18 model [12] trained on the CIFAR-10 dataset. CIFAR-10 consists of 60,000 color images of size 32×32 , distributed across ten object classes. The dataset provides a computationally tractable benchmark while retaining sufficient visual complexity to study the evolution of deep architectures during training.

Residual Networks

As convolutional neural networks become deeper, optimization difficulties begin to emerge. Although increasing network depth generally improves expressive capacity, very deep architectures may suffer from degradation phenomena in which training accuracy saturates or deteriorates despite the addition of new layers [12]. This behavior is closely related to unstable gradient propagation during backpropagation.

Residual Networks (ResNets) address this issue through the introduction of skip connections, also referred to as identity shortcuts [12]. Rather than learning a direct mapping $H(x)$, residual blocks are designed to learn a residual transformation $\mathcal{F}(x)$ relative to the input. Formally, the block computes

$$H(x) = x + \mathcal{F}(x). \quad (2.14)$$

More generally, a residual block can be written as

$$y = \sigma(x + \mathcal{F}(x, W)), \quad (2.15)$$

where $\mathcal{F}(x, W)$ denotes the transformation parameterized by weights W , and $\sigma(\cdot)$ represents a non-linear activation function. In practice, $\mathcal{F}(x, W)$ typically consists of convolutional layers, batch normalization, and ReLU activations.

The skip connection facilitates optimization by preserving information from earlier layers and improving gradient propagation across the network depth. Intuitively, residual learning allows the network to iteratively refine intermediate computations instead of reconstructing them entirely at each layer. This substantially stabilizes optimization and enables the successful training of deep convolutional architectures.

The ResNet-18 architecture employed in this thesis consists of multiple residual blocks organized hierarchically. In Figure 2.4, the skip connection is exposed, as data propagates through the network, the input undergoes a sequence of non-linear transformations while identity shortcuts preserve stable information flow between layers. This architectural structure provides an appropriate framework for studying how neural networks evolve throughout training.

In the following section, we formalize how the outputs generated by different layers of a neural network can be interpreted as internal representations of the data. These

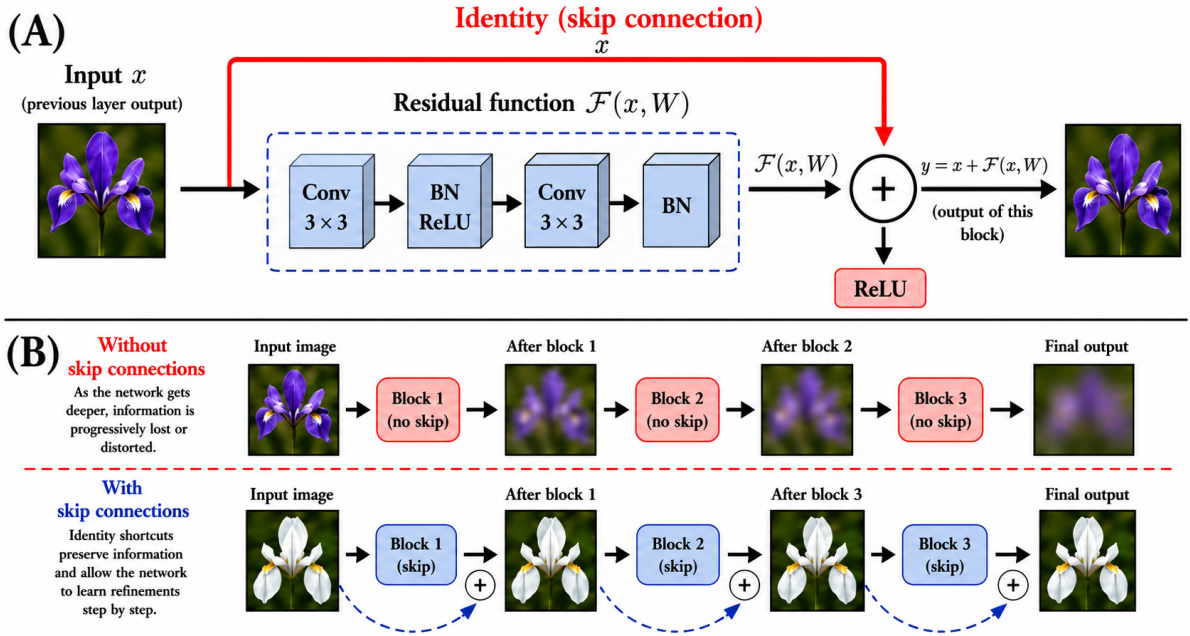


Figure 2.4: **Detailed illustration of the residual learning mechanism employed in Residual Networks (ResNets).** The panel (A) depicts the internal structure of a residual block. An input tensor x , corresponding to the output of a previous layer, is processed through a sequence of convolutional transformations represented by the residual function $\mathcal{F}(x, W)$. In this example, the residual branch consists of two 3×3 convolutional layers combined with batch normalization (BN) and ReLU activation. Simultaneously, the original input x is propagated directly through an identity shortcut (skip connection), shown by the red pathway. The outputs of both branches are combined through element-wise addition, producing $y = x + \mathcal{F}(x, W)$, after which a ReLU non-linearity is applied. This formulation allows the network to learn corrective refinements relative to the identity mapping instead of reconstructing the full transformation at each layer. Panel (B) compares the propagation of iris images through a deep architecture without and with skip connections. In the first row (without skip connections), each block applies successive non-linear transformations independently, causing progressive degradation and loss of local visual structure as depth increases. The resulting output becomes increasingly blurred, illustrating how information may deteriorate in very deep convolutional networks. In the second row (with skip connections), identity shortcuts preserve relevant information from earlier layers while residual blocks learn incremental modifications. As a consequence, the essential geometric and visual characteristics of the iris image remain stable throughout the network depth.

representations will later constitute the main objects of analysis throughout this thesis.

2.2. Representations

One of the central ideas in deep learning is that neural networks progressively transform the original data into increasingly abstract internal representations. As introduced in the previous sections, each hidden layer applies a non-linear transformation to its input through a composition of affine maps and activation functions. Consequently, the output vector produced at a given layer contains information about how the network internally encodes the data at a particular stage of computation.

For an input observation $x_i \in \mathbb{R}^d$, recall that the output of layer h is denoted by

$$\mathbf{r}_h(x_i) = f^{(h)}(f^{(h-1)}(\dots f^{(1)}(x_i) \dots)) \in \mathbb{R}^{J^{(h)}}, \quad (2.16)$$

where $J^{(h)}$ denotes the width of the h -th layer. The vector $\mathbf{r}_h(x_i)$ is referred to as the *representation* of the observation x_i at layer h . Geometrically, the layer maps the original input into a new feature space whose dimension is determined by the width of the layer.

Given a dataset $\mathcal{X} = \{x_1, \dots, x_n\}$, the representations generated by layer h can be collected into a matrix

$$\mathbf{R}_h = \begin{pmatrix} \mathbf{r}_h(x_1)^\top \\ \mathbf{r}_h(x_2)^\top \\ \vdots \\ \mathbf{r}_h(x_n)^\top \end{pmatrix} \in \mathbb{R}^{n \times J^{(h)}}. \quad (2.17)$$

Each row of $\mathbf{R}_h$ corresponds to the representation of a single observation, while each column corresponds to the activation of a neuron across the dataset. Therefore, the matrix $\mathbf{R}_h$ provides a complete description of how layer h organizes the dataset within its induced feature space.

The representations depend on the network parameters θ . During training, these parameters are iteratively updated through SGD, causing the geometry of the representation space to evolve over time. Let $\theta^{(p)}$ denote the parameters obtained after epoch p . Evaluating the test set $\mathcal{X}_{\text{test}}$ using $\theta^{(p)}$ produces a family of representations

$$\mathbf{R}_h^{(p)} = \left(\mathbf{r}_h^{(p)}(x_1), \dots, \mathbf{r}_h^{(p)}(x_n) \right)^\top, \quad (2.18)$$

where

$$\mathbf{r}_h^{(p)}(x_i) = f_{\theta^{(p)}}^{(h)}(f_{\theta^{(p)}}^{(h-1)}(\dots f_{\theta^{(p)}}^{(1)}(x_i) \dots)). \quad (2.19)$$

Thus, each epoch induces a new geometric organization of the same dataset. In this

sense, training can be interpreted as a dynamical process acting on the geometry of the representations. Early epochs typically produce unstable or weakly structured embeddings, while later epochs progressively organize the data into feature spaces that are more suitable for the learning objective.

Using the test set $\mathcal{X}_{\text{test}}$ for representation analysis is particularly important because it allows the study of representations independently of the optimization process itself. Since the parameters are updated directly using $\mathcal{X}_{\text{train}}$, representations extracted from the training set may partially reflect optimization artifacts specific to the observed samples. In contrast, evaluating the network on $\mathcal{X}_{\text{test}}$ provides a description of how the learned representations generalize to unseen data. Consequently, the geometric structure observed in the representation space more accurately reflects the features and invariances learned by the network rather than the particular samples used during optimization.

An important characteristic of these representations is their non-linear nature. Since activation functions are applied after affine transformations, the resulting feature spaces are not constrained to linear transformations of the original data. Instead, neural networks can progressively deform the geometry of the data manifold, separating classes, compressing redundant information, and constructing hierarchical features adapted to the task.

Understanding these internal representations has become a fundamental problem in deep learning research. As discussed in [2, 1, 3], studying representations provides insight into how neural networks learn, how different architectures organize information, and whether distinct models converge toward similar internal structures. Representation analysis has also been used to study redundancy across layers, the hierarchical organization of learned features, and the convergence properties of optimization procedures.

In particular, analyzing the evolution of representations throughout training offers a framework for studying learning dynamics. Rather than focusing exclusively on predictive accuracy or loss values, one may instead investigate how the geometry of the representation space changes across layers and epochs. This perspective is especially relevant for deep architectures, where learning emerges through a sequence of non-linear geometric transformations acting on the data manifold.

The central objective of this thesis is precisely to study these representational dynamics. To achieve this, representations will be modeled as weighted graphs whose geometry is analyzed through spectral graph theory and investigating a graph-based distance measures. By comparing the structural organization of representations across layers and training epochs, we aim to characterize how neural networks progressively construct meaningful internal representations of data.

Chapter 3

Theory and Methods

3.1. Spectral Graph Theory

Spectral graph theory is the study of the spectral decomposition of a matrix known as the graph Laplacian, which is constructed from the nodes (or vertices) of a weighted, undirected graph. Let $G = (V, E, W)$ be a weighted, undirected graph, where V denotes the set of nodes, $V = \{v_1, \dots, v_n\}$, and E denotes the set of edges, $E = \{\{i, j\} : \text{nodes } i \text{ and } j \text{ are connected}\}$. The edges are defined pairwise and can therefore be represented by a symmetric matrix $A = (a_{ij})_{i,j=1,\dots,n}$, called the adjacency matrix, where the entries are given by $a_{ij} = \mathbf{1}_{\{\{i,j\} \in E\}}$. In other words, the entry equals one if the edge exists and zero otherwise. Each pair of nodes v_i and v_j , with $i, j \in \{1, \dots, n\}$, is associated with a weight $w_{ij} \geq 0$. The matrix $W = (w_{ij})_{i,j=1,\dots,n}$ is called the *weight matrix*; it is symmetric and, for our purposes, it has no self-weights, i.e., $w_{ii} = 0$ for all i . Moreover, $w_{ij} = 0$ whenever nodes v_i and v_j are not connected.

A *connected graph* is a graph in which every pair of nodes is connected by a path. The degree of a node $v_i \in V$ is defined as

$$d_i = \sum_{j=1}^n w_{ij}. \quad (3.1)$$

The volume of G is defined as the sum of all node degrees,

$$\text{vol}(G) = \sum_{i=1}^n d_i. \quad (3.2)$$

We define the *degree matrix* as the diagonal matrix containing the node degrees, i.e.,

$$D = \text{diag}(d_1, \dots, d_n). \quad (3.3)$$

The graph Laplacian is a matrix that encapsulates the structural properties of a weighted, undirected graph. It provides a way to describe how each node differs from its neighbors.

For each node, the Laplacian captures the total “strength” of its connections and the “influence” of its neighboring nodes in terms of the weights.

Definition 3.1 (The graph Laplacian). *Given a weighted, undirected graph G with degree matrix D and weighted adjacency matrix W , define the Laplacian of G as*

$$L := D - W \quad (3.4)$$

As discussed in [6] and [7], several important properties are worth mentioning.

Theorem 3.1 (Properties of L). *The matrix L satisfies the following properties:*

1. L is symmetric and positive semi-definite.
2. The smallest eigenvalue of L is 0, and the corresponding eigenvector is the constant one vector $\mathbb{1}$.
3. L has n non-negative, real-valued eigenvalues

$$0 = \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n. \quad (3.5)$$

The proof is presented in Appendix A.1. These properties are important since they describe the foundational framework for studying connected graphs. In particular, the fact that the Laplacian is positive semi-definite will later allow us to study the CTD as a Euclidean distance measure.

Analogous definitions can be derived from the weights of a connected graph. In [6], it was mentioned how in the general literature the name “Laplacian matrix” can be interchangeable with different concepts. The *normalized laplacian* is defined as

$$L_{\text{sym}} := D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2} \quad (3.6)$$

and the *random walk Laplacian* is

$$L_{\text{rw}} := D^{-1} L = I - D^{-1} W. \quad (3.7)$$

Both are strongly related. Like in the Laplacian matrix, they both have zero as their smallest eigenvalue. For instance, $L_{\text{rw}} = D^{-1/2} L_{\text{sym}} D^{1/2}$ and they share the same eigenvalues, but not the same eigenvectors. L_{rw} gets its name from its relation with the stochastic process.

In the following sections, reasons for exclusively using L in this thesis will be discussed. For now, we will expose how the Laplacian matrix is related to a Random Walker defined over a connected graph and its CTD.

3.2. The Commute Time Distance

The notion of distance on a graph can be defined in several ways. This section follows the results of Fouss, Pirotte, Renders, and Saerens in [7]. Let us consider an undirected, weighted graph $G = (V, E, W)$. The CTD between two nodes v_i and v_j is defined as the expected time required for a random walker to travel from one node to the other and return, denoted by $c(i, j)$. More precisely, it corresponds to the *average first-passage time* from v_i to v_j and back. To formalize this notion, we first define the random walker on the graph.

3.2.1. Random Walk

The Markov chain describing the sequence of visited nodes by a random walker is referred to as a random walk. In this setting, we consider a first-order Markov chain whose state space is given by the node set V . The state at time t is denoted by $s(t)$; if the walker is at node v_i at time t , then $s(t) = v_i$. Transitions are restricted to adjacent nodes, meaning that if $w_{ij} = 0$, then $\mathbb{P}[s(t+1) = v_j \mid s(t) = v_i] = 0$ (i.e., no edge exists between i and j). In general, the transition probabilities are given by

$$\mathbf{p}_{ij} = \mathbb{P}[s(t+1) = v_j \mid s(t) = v_i] = \frac{w_{ij}}{\sum_{j=1}^n w_{ij}}. \quad (3.8)$$

Here, $\mathbf{P} = (\mathbf{p}_{ij})_{i,j=1,\dots,n}$ denotes the transition matrix of the Markov chain. Since the graph is connected, the Markov chain is irreducible; that is, every state can be reached from any other state. If this condition is not satisfied, the chain can be decomposed into disjoint closed communicating classes, each of which is irreducible, and the analysis can be applied independently to each component.

The relation between this stochastic process and the random walk Laplacian comes from the way the transition matrix is constructed, since

$$\mathbf{P} = D^{-1}W \quad (3.9)$$

then it is easy to check that $L_{\text{rw}} = I - \mathbf{P}$. In other words, the random walk Laplacian shares the same eigenvectors with the transition matrix of the random walker and λ is an eigenvalue of L_{rw} if and only if $1 - \lambda$ is an eigenvalue of $\mathbf{P}$.

Another important concept to consider is the *average first-passage time*. The average first-passage time $m(j \mid i)$ is defined as the expected number of steps required for a random walker, starting at state i , to reach state j for the first time ($i \neq j$). Formally, we define the stopping time as the first hitting time of state j starting from i :

$$T_{ij} = \min\{t \geq 0 \mid s(t) = j, s(0) = i\}, \quad (3.10)$$

for a given realization of the stochastic process. The average first-passage time is then given by

$$m(j \mid i) = \mathbb{E}[T_{ij} \mid s(0) = i]. \quad (3.11)$$

The average first-passage time satisfies a fundamental recurrence relation obtained via first-step analysis. Intuitively, to move from state i to state j , the process first transitions to some neighboring state k , and then continues from there:

$$m(j \mid i) = \begin{cases} 0, & i = j, \\ 1 + \sum_{k=1}^n p_{ik} m(j \mid k), & i \neq j. \end{cases} \quad (3.12)$$

The CTD is then defined as

$$c(i, j) = m(i \mid j) + m(j \mid i). \quad (3.13)$$

More importantly, $\sqrt{c(i, j)}$ can be interpreted as a Euclidean distance on the nodes of the graph. As expected, it has been shown in [9] that it satisfies the properties of a metric, namely, for any states i, j, k :

1. $\sqrt{c(i, j)} \geq 0$ (non-negativity),
2. $\sqrt{c(i, j)} = 0$ if and only if $i = j$ (definiteness),
3. $\sqrt{c(i, j)} = \sqrt{c(j, i)}$ (symmetry), and
4. $\sqrt{c(i, j)} \leq \sqrt{c(i, k)} + \sqrt{c(k, j)}$ (triangle inequality).

Although we already mentioned how is the random walk Laplacian matrix L_{rw} related to a random walker on a graph. We also want to introduce its relation with the Laplacian matrix L . The next section presents this relation between a random walker on a graph and L as one of the main theoretical tools used in this thesis. It also explains why only L is needed for the analysis of representations inside a neural network.

3.2.2. The Embedding

While the CTD provides a natural measure of proximity based on random walks, computing it directly via transition matrices may be computationally inconvenient. The graph Laplacian offers a powerful, structural alternative. By leveraging the Moore-Penrose pseudoinverse of the Laplacian, we can compute the exact CTD between any two nodes without explicitly simulating random walks. More importantly, this algebraic relationship reveals a profound geometric property: the pseudoinverse of the Laplacian acts as a kernel. This insight allows us to transition from a conceptual distance on a graph to a explicit

embedding in Euclidean space, where the CTD is preserved as the standard Euclidean distance. This is established by the following theorem, whose proof is given in [8].

Theorem 3.2 (Commute time distance). *Let $G = (V, E, W)$ be a connected, undirected graph and let $L^\dagger = (l_{ij}^\dagger)_{i,j=1,\dots,n}$ denote the Moore–Penrose pseudoinverse of the Laplacian L . Then the CTD between nodes i and j is given by,*

$$c(i, j) = \text{vol}(G) \left(l_{ii}^\dagger - 2l_{ij}^\dagger + l_{jj}^\dagger \right) = \text{vol}(G) (e_i - e_j)^\top L^\dagger (e_i - e_j). \quad (3.14)$$

This result is particularly significant because, using the eigenvalue decomposition $L = \mathcal{V}\Lambda\mathcal{V}^\top$, where $\mathcal{V}$ contains the eigenvectors of L and Λ is the diagonal matrix of corresponding eigenvalues, one can construct an embedding that maps each node i to a vector $z_i \in \mathbb{R}^{n-1}$ such that

$$\langle z_i, z_j \rangle = \text{vol}(G) e_i^\top L^\dagger e_j \quad \text{and} \quad \|z_i - z_j\|_2 = \sqrt{c(i, j)}. \quad (3.15)$$

In Figure 3.1 a toy example of this theorem will be presented. Now, it is possible to define a Euclidean geometry over the nodes of a graph induced by the Laplacian. It is important to note that in $\mathbb{R}^n$, $L^\dagger$ induces only a pseudo-distance, since $L^\dagger$ is not positive definite as a consequence of its smallest eigenvalue being zero.

A further implication concerns the explicit construction of the embedding z_i . Since the first eigenvector corresponds to the constant vector $\mathbb{1}$, let $\mathcal{V}'$ be the $(n-1) \times n$ matrix defined as $\mathcal{V}' = (\mathbf{v}_2, \dots, \mathbf{v}_n)$, and let Λ' be the $(n-1) \times (n-1)$ diagonal matrix satisfying $\text{diag}(\Lambda') = (\lambda_2, \lambda_3, \dots, \lambda_n)$. The embedding can then be written as

$$\Theta = \sqrt{\text{vol}(G)} (\Lambda')^{-1/2} (\mathcal{V}')^\top, \quad \Theta = (z_1, z_2, \dots, z_n). \quad (3.16)$$

For each node i , the corresponding vector in $\mathbb{R}^{n-1}$ is given by

$$z_i = \sqrt{\text{vol}(G)} \left(\frac{\mathbf{v}_{2i}}{\sqrt{\lambda_2}}, \frac{\mathbf{v}_{3i}}{\sqrt{\lambda_3}}, \dots, \frac{\mathbf{v}_{ni}}{\sqrt{\lambda_n}} \right), \quad (3.17)$$

and its α -th coordinate satisfies

$$z_{i\alpha} = \sqrt{\frac{\text{vol}(G)}{\lambda_\alpha}} \mathbf{v}_{\alpha i}. \quad (3.18)$$

The scaling factor $1/\sqrt{\lambda_\alpha}$ modifies the metric structure but preserves the linear independence of the eigenvectors. Consequently, the vectors z_i correspond to the coordinates of the vectors e_i projected onto the subspace spanned by $\mathbf{v}_2, \dots, \mathbf{v}_n$, that is,

$$\text{span}\{\mathbf{v}_2, \dots, \mathbf{v}_n\} = \mathbb{1}^\perp \subset \mathbb{R}^n. \quad (3.19)$$

This embedding does not span the entire space $\mathbb{R}^n$; rather, it spans precisely the $(n-1)$ -

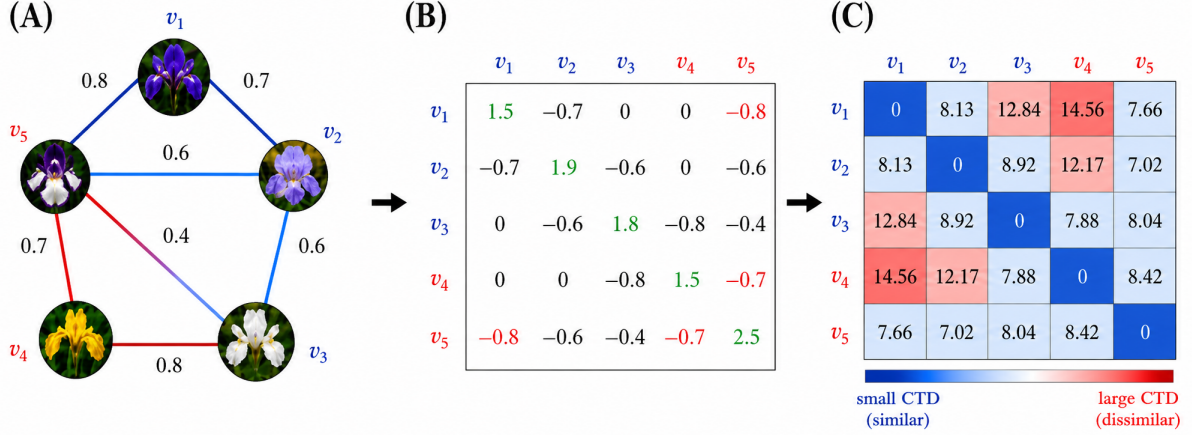


Figure 3.1: **Construction of the CTD from an artificial weighted graph of iris images.** (A) The left panel represents the dataset as an undirected weighted graph $G = (V, E, W)$, where each node v_i corresponds to an iris image and each edge weight w_{ij} encodes the similarity between two images. Larger weights indicate stronger similarity between flowers, while smaller weights correspond to weaker visual similarity. (B) The middle panel shows the associated graph Laplacian matrix $L = D - W$, where D is the degree matrix and W is the weighted adjacency matrix. The diagonal entries contain the node degrees, whereas the off-diagonal entries are the negative edge weights connecting neighboring nodes. The Laplacian summarizes the global connectivity structure of the graph and constitutes the central operator used to define random-walk dynamics on the network. (C) The right panel presents the resulting commute time distance matrix $c(i, j) = \text{vol}(G) (e_i - e_j)^\top L^\dagger (e_i - e_j)$, computed from the Moore–Penrose pseudoinverse of the Laplacian. Each entry measures the expected number of steps required for a random walker to travel from node v_i to node v_j and return. The color transition from blue to red indicates increasing graph dissimilarity: blue regions correspond to small CTD values and therefore highly connected or visually similar iris images, whereas red regions correspond to large CTD values, indicating weaker connectivity and larger distance across the graph. From a geometric perspective, the CTD captures the intrinsic relational structure of the dataset by incorporating all possible paths between nodes rather than only direct pairwise similarities.

dimensional subspace on which the Laplacian is invertible, which is isomorphic to $\mathbb{R}^{n-1}$. One may interpret $\mathbb{1}^\perp \subset \mathbb{R}^n$ as the ambient subspace where the intrinsic geometry of the graph is fully represented.

Now, why use L exclusively instead of L_{rw} or L_{sym} ? The answer not only comes from the simplicity of L but also from the properties involved. Using L_{rw} as a kernel for inducing a distance measure breaks the symmetry property required for a function to define a proper distance measure. This happens because L_{rw} itself is non-symmetric. On the other hand, L_{sym} is indeed symmetric and it appears to be more related to L . Then let us construct the embedding induced by the normalized Laplacian.

Let $L_{\text{sym}} = \mathcal{U} \Lambda^{\text{sym}} \mathcal{U}^\top$ be the eigendecomposition of L_{sym} , where $\mathcal{U} = (\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_n)$ contains the orthonormal eigenvectors of L_{sym} , and $\Lambda^{\text{sym}} = \text{diag}(0, \lambda_2^{\text{sym}}, \dots, \lambda_n^{\text{sym}})$ is the diagonal matrix of corresponding eigenvalues. Since the first eigenvalue satisfies $\lambda_1^{\text{sym}} = 0$, let $\mathcal{U}' = (\mathbf{u}_2, \dots, \mathbf{u}_n)$, and define $\Lambda^{\text{sym}'} = \text{diag}(\lambda_2^{\text{sym}}, \dots, \lambda_n^{\text{sym}})$. The normalized embedding is then given by

$$\Theta^{\text{sym}} = \sqrt{\text{vol}(G)} (\Lambda^{\text{sym}'})^{-1/2} (\mathcal{U}')^\top D^{-1/2}, \quad \Theta^{\text{sym}} = (\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_n). \quad (3.20)$$

For each node $v_i \in V$, the corresponding vector in $\mathbb{R}^{n-1}$ is

$$\mathbf{z}_i = \sqrt{\frac{\text{vol}(G)}{d_i}} \left(\frac{\mathbf{u}_{2i}}{\sqrt{\lambda_2^{\text{sym}}}}, \frac{\mathbf{u}_{3i}}{\sqrt{\lambda_3^{\text{sym}}}}, \dots, \frac{\mathbf{u}_{ni}}{\sqrt{\lambda_n^{\text{sym}}}} \right). \quad (3.21)$$

The similarities between equation 3.17 and 3.21 are clear from a structural point of view. But, there is no guarantee for the Laplacian to have the same spectrum as the normalized Laplacian. A second difference comes from the scaling. In equation 3.21 the scaling is not only using the volume of G it also divides by the square root of the degree of the node v_i i.e. d_i .

Surprisingly; using this new construction is not different from the CTD perspective since it defines an equivalent distance measure to the original construction, as the next theorem states.

Theorem 3.3 (Embedding equivalence). *Let $G = (V, E, W)$ be a connected, undirected graph with degree matrix $D = \text{diag}(d_1, \dots, d_n)$. Let $c(i, j)$ denote the commute time distance between nodes v_i and v_j , and let $L_{\text{sym}}^\dagger$ denote the Moore–Penrose pseudoinverse of the normalized Laplacian*

$$L_{\text{sym}} = D^{-1/2} L D^{-1/2}. \quad (3.22)$$

Then

$$c(i, j) = \text{vol}(G) (e_i - e_j)^\top D^{-1/2} L_{\text{sym}}^\dagger D^{-1/2} (e_i - e_j) = \|\mathbf{z}_i - \mathbf{z}_j\|_2^2. \quad (3.23)$$

For the interested reader, the full proof is given in Appendix A.2. This property effectively tells us that, for computing the CTD, choosing between the embeddings induced by L_{sym} and L is equivalent. For the rest of the thesis only L will be used for computations since its use results in a more parsimonious approach. This properties and the convinient CTD calculations sets the tools for a deeper analysis in the following construction of a graph-by-graph distance.

3.3. Shape-Aware Graph Distance

As established in the previous sections, we now possess a well-defined notion of distance at the level of individual nodes. However, the primary objective is to extend this framework in order to define a meaningful and robust distance between entire graphs. The CTD provides a natural foundation for this purpose. In particular, its square root encodes intrinsic geometric properties of the graph, reflecting how information propagates through its structure.

The key idea is to leverage this information by considering all pairwise CTDs $c(i, j)$ for $i < j$, and summarizing them through their eCDFs. Rather than comparing graphs at the level of individual node pairs, the approach consists of comparing these distributions. To this end, the 1-*Wasserstein* distance is employed to measure the discrepancy between the eCDFs associated with two graphs.

From an intuitive perspective, the eCDF serves as a compact representation of the structural information captured by the CTDs. Therefore, comparing two graphs reduces to comparing their corresponding distributions, allowing for a global and statistically meaningful notion of similarity.

3.3.1. 1-Wasserstein Distance

Conceptually, the p -Wasserstein distance (also known as the Kantorovich–Rubinstein metric) is defined as the minimal expected distance between two probability measures endowed with a joint coupling, under the assumption that their realizations are mutually dependent. Its original interpretation [18] is grounded in the notion of the minimal “cost” required to transform one distribution into another, formulated as an optimal transport problem; the hence called “Earth Mover’s distance”. More recently, it has been widely adopted in machine learning, particularly in the training of GANs [19].

Formally, assume that $(\mathcal{E}, d)$ is a separable metric space¹ and, as defined in [10], let

¹That is, d is a distance function defined on the set $\mathcal{E}$.

$\mathcal{Z}_p(\mathcal{E}, d)$ denote the collection of all Borel probability measures μ on $\mathcal{E}$ with finite p -th moment.² For $\mu, \nu \in \mathcal{Z}_p(\mathcal{E}, d)$, define the quantity $W_p(\mu, \nu) \geq 0$ by

$$W_p^p(\mu, \nu) = \inf_{\pi \in \Pi(\mu, \nu)} \int_{\mathcal{E} \times \mathcal{E}} d(x, y)^p d\pi(x, y), \quad (3.24)$$

where the infimum is taken over all probability measures π on the product space $\mathcal{E} \times \mathcal{E}$ (equipped with the product σ -algebra) with marginals μ and ν , in the sense that $\pi(A \times \mathcal{E}) = \mu(A)$, $\pi(\mathcal{E} \times B) = \nu(B)$, for all Borel sets $A, B \subset \mathcal{E}$. The value $W_p(\mu, \nu)$ is referred to as the *Wasserstein distance* of order p between μ and ν .

Why is the 1-Wasserstein distance a suitable candidate? Note that definition 3.24 is expressed in terms of probability measures rather than cumulative distribution functions. For this purpose, consider the following theorem by Dall'Aglio [11].

Theorem 3.4 (Representation for W_1). *Let μ and ν be probability measures in $\mathcal{Z}_1(\mathbb{R})$ with respective cumulative distribution functions F and $\tilde{F}$. Then*

$$W_1(\mu, \nu) = \int_{-\infty}^{\infty} |F(x) - \tilde{F}(x)| dx. \quad (3.25)$$

This theorem enables comparison between the eCDFs of the CTDs associated with two graphs. In other words, it enables the quantification of the cost of transforming one graph into another in a symmetric manner, since W_1 defines a distance on $\mathcal{Z}_1(\mathbb{R})$.

An equivalent representation is given by

$$W_1(\mu, \nu) = \int_0^1 |F^{-1}(t) - \tilde{F}^{-1}(t)| dt, \quad (3.26)$$

where F^{-1} and $\tilde{F}^{-1}$ denote the corresponding quantile functions. This formulation is extensively used in probability and statistics and is discussed in standard references [18].

Both formulations quantify equivalent discrepancies between distributions. However, the first integrates along the horizontal axis, while the second integrates along the vertical axis (differences of quantiles). Geometrically, both quantify the same area. This observation provides a convenient formulation for one-dimensional eCDFs, assuming both samples contain n ordered observations: $X_{(1)} \leq X_{(2)} \leq \dots \leq X_{(n)}$ and $Y_{(1)} \leq Y_{(2)} \leq \dots \leq Y_{(n)}$. The eCDFs are given by

$$F_n(x) = \frac{1}{n} \sum_{i=1}^n \mathbf{1}_{\{X_i \leq x\}}, \quad \tilde{F}_n(y) = \frac{1}{n} \sum_{i=1}^n \mathbf{1}_{\{Y_i \leq y\}}. \quad (3.27)$$

The inverse eCDFs at $\tau \in [0, 1]$ are given by:

$$F_n^{-1}(\tau) = X_{(\lceil n\tau \rceil)}, \quad \tilde{F}_n^{-1}(\tau) = Y_{(\lceil n\tau \rceil)}, \quad (3.28)$$

²In particular, $\mathcal{Z}_1(\mathbb{R})$ corresponds to the set of all Borel probability measures on $\mathbb{R}$ equipped with the Euclidean distance and finite expectation.

where $\lceil n\tau \rceil$ denotes the smallest integer greater than or equal to $n\tau$. Partitioning the interval $[(i-1)/n, i/n]$ for $i = 1, \dots, n$, we obtain the simplified expression:

$$\begin{aligned} W_1(F_n, \tilde{F}_n) &= \sum_{i=1}^n \int_{(i-1)/n}^{i/n} |X_{(i)} - Y_{(i)}| d\tau \\ &= \sum_{i=1}^n \frac{1}{n} |X_{(i)} - Y_{(i)}| \\ &= \frac{1}{n} \sum_{i=1}^n |X_{(i)} - Y_{(i)}|. \end{aligned} \tag{3.29}$$

The previous results establish the 1-Wasserstein distance as a natural and computationally tractable tool for comparing empirical distributions. In particular, the representation in equation 3.29 provides a direct way to quantify discrepancies between collections of CTDs through their ordered samples. This observation is fundamental for the present framework, since each graph induces a distribution of pairwise CTDs that encodes its global geometric structure. Figure 3.2 already introduces the notion of comparison of images as data points, we compared the dissimilarity between representation via the eCDFs of CTDs.

Consequently, instead of comparing graphs through individual node correspondences, we compare the distributions generated by their CTDs. This motivates the introduction of a graph-level distance based on the 1-Wasserstein distance between the corresponding eCDFs, leading to the definition of the Shape-Aware Graph Distance.

3.3.2. Definition and Properties

We now define a distance measure on a discrete space of weighted, undirected graphs. A key requirement is that all graphs in this space share the same number of nodes³, and that they are connected, i.e., they contain no independent components.

The requirement of a fixed number of nodes is motivated by the notion of representation similarity introduced in [1]. Since the central idea is to represent data as graphs, any transformations applied to the data—such as expansions or contractions of the feature space, or more generally non-linear transformations—can then be tracked through a corresponding family of representational graphs.

Definition 3.2 (Representation Graph Family). *Let $\mathbb{G}_{nm}$ be a set of weighted, undirected and connected graphs with the same amount of nodes;*

$$\mathbb{G}_{nm} := \{G^{(h)} = (V, E^{(h)}, W^{(h)}) : |V| = n, 1 \leq h \leq m\} \tag{3.30}$$

³This is a flexible restriction. The distance can be defined over an arbitrary collection of graphs, since the proposed construction does not fundamentally depend on comparing graphs with equal numbers of nodes. The only limitation arises when using formula 3.29 since it requires the same amount of CTD values derived by having the same number of nodes in each graph.

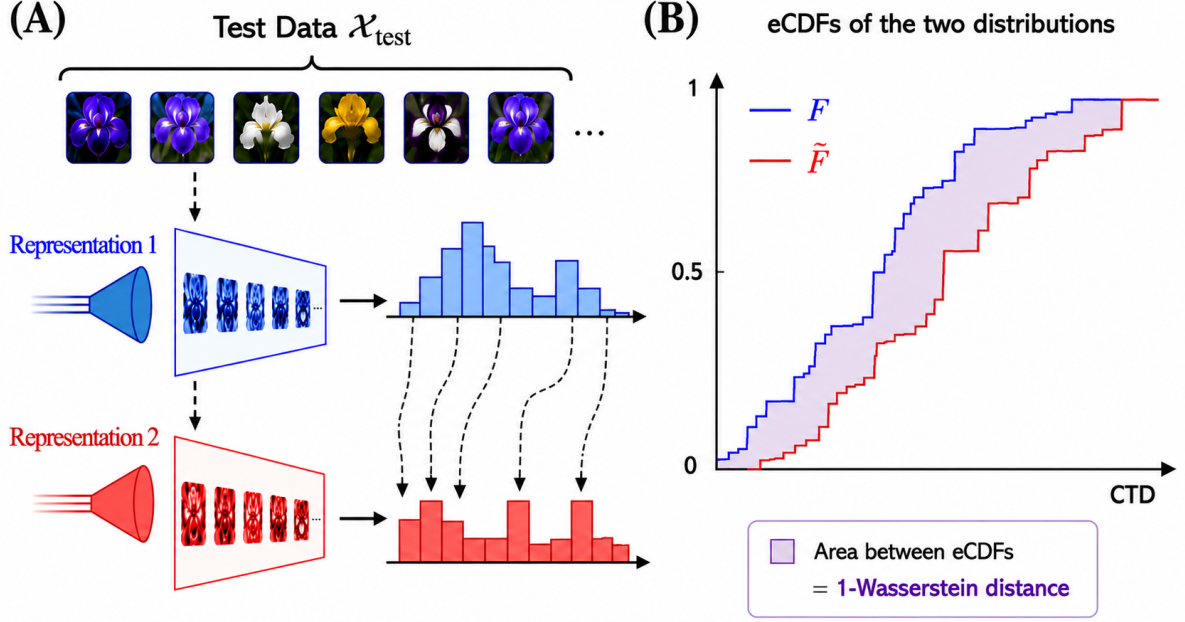


Figure 3.2: **Distributions of CTDs induced by different representations of the same dataset.** The diagram (A) illustrates how the same collection of iris images can generate different distributions of pairwise commute time distances under two distinct representations. The upper row represents the original dataset, where each flower image corresponds to a data point in the graph construction. The left panel shows two different representations of these data points (blue and red), each producing a distinct distribution of CTDs summarized by a histogram. These histograms encode the structural relationships between all image pairs in the corresponding graph representation. The right panel (B) displays the empirical cumulative distribution functions (eCDFs) associated with the two CTD distributions, denoted by F and $\tilde{F}$. The shaded region between the curves represents the discrepancy measured by the 1-Wasserstein distance, which quantifies the amount of “work” required to transform one CTD distribution into the other.

where n corresponds to the amount of nodes and m the number of elements.

In practice, representations derived from a dataset $\mathcal{X}$ tend to arise from a finite process. Nevertheless, the case $m = \infty$ remains a theoretical possibility, in which transformations may be described by an infinite procedure. A Representation Graph Family, by definition, will remain discrete, motivated by the representation similarity framework.

Definition 3.3 (Shape-Aware Graph Distance). *Let $\mathbb{G}_{nm}$ be a Representation Graph Family and $c(i, j)$ the CTD between nodes $v_i, v_j \in V$ for every $1 \leq i < j \leq n$, defined for the graph $G \in \mathbb{G}_{nm}$ ($\tilde{c}(i, j)$, respectively for $\tilde{G} \in \mathbb{G}_{nm}$). Let F be the eCDF of the CTDs $c(i, j)$, $1 \leq i < j \leq n$ (respectively $\tilde{F}$ for $\tilde{c}(i, j)$).*

Then the Shape-Aware Graph Distance corresponds to the distance measure $SAGD : \mathbb{G}_{nm} \times \mathbb{G}_{nm} \rightarrow \mathbb{R}_{\geq 0}$ such that;

$$SAGD(G, \tilde{G}) = W_1(F, \tilde{F}) \quad (3.31)$$

where, W_1 corresponds to the 1–Wasserstein distance.

A fundamental observation is that the 1-Wasserstein distance defines a metric on the space of probability measures with finite first moments. Consequently, provided that the mapping $G \mapsto F$ is well-defined, the metric properties of W_1 are naturally inherited by $SAGD$. In particular, non-negativity, symmetry, and the triangle inequality follow directly from the corresponding properties of the Wasserstein distance.

However, it is important to carefully interpret the definiteness condition. From the properties of W_1 , one has $W_1(F, \tilde{F}) = 0$ if and only if $F = \tilde{F}$. This equality indicates that the eCDFs of CTDs coincide, but it does not imply that the underlying graphs themselves are identical. Distinct graphs may induce identical distributions of pairwise CTDs. Therefore, the condition $SAGD(G, \tilde{G}) = 0$ does not necessarily entail that G and $\tilde{G}$ are identical. As a consequence, the pair $(\mathbb{G}_{nm}, SAGD)$ does not, in general, constitute a metric space; rather, it should be regarded as a pseudo-metric space.

An important advantage of $SAGD$ is its ability to encode global structural information of the graph. Since the CTD captures relationships between nodes through the connectivity of the entire network, the resulting distribution reflects the graph structure from a global perspective.

An additional strength lies in the fact that $SAGD$ incorporates both eigenvalues and eigenvectors of the Laplacian, without requiring any independence assumptions between them. This reduces sensitivity to co-spectral graphs, which may share identical eigenvalues but differ structurally. Furthermore, the scaling induced by the spectral decomposition ensures that higher-order eigenvectors—often associated with noise—do not dominate the representation, thereby improving robustness.

In addition, the formulation can be extended to compare graphs of different sizes. Nevertheless, for the purposes of this work, the Representation Graph Family has been

defined under the assumption that all graphs share the same number of nodes. While this restriction is not strictly necessary, it simplifies the exposition, and it will be maintained throughout the remainder of this thesis.

3.3.3. Normalization

To enhance the robustness of the SAGD, it is beneficial to normalize the samples of the CTDs prior to computing the 1-Wasserstein distance. When comparing multiple graphs, their edge weights may exist on entirely different numerical scales. If left unadjusted, the Wasserstein distance might capture arbitrary scaling differences rather than genuine structural variations. The primary objective of normalization is to map all graphs onto a comparable scale, thereby neutralizing global scaling effects and isolating the intrinsic geometric deformations of the representations. A natural approach to achieve this scale invariance is to normalize the CTDs by the total volume of the graph, $\text{vol}(G)$. Because scaling all edge weights in a graph by a constant factor proportionally scales its CTDs, normalizing by the volume effectively cancels out the “units” of the weights. Consequently, this transformation ensures that SAGD remains stable and focused exclusively on comparing the underlying topological structure of the graphs.

An alternative approach is to normalize the CTDs by their mean. This ensures the eCDFs are evaluated on a comparable scale, as both first moments are standardized to unity. Notably, this procedure implicitly performs a volume normalization, as demonstrated by the following relationship;

$$\frac{c(i, j)}{\bar{c}} = \frac{\text{vol}(G) \|z_i - z_j\|_2^2}{\text{vol}(G) \frac{1}{l} \sum_{h < k} \|z_h - z_k\|_2^2} = \frac{l \cdot \|z_i - z_j\|_2^2}{\sum_{h < k} \|z_h - z_k\|_2^2} \quad (3.32)$$

where $l = \frac{n(n-1)}{2}$.

Here, the normalization essentially rescales the size of norms in the embedding space.

Definition 3.4 (Mean-Normalized SAGD). *Let $G, \tilde{G} \in \mathbb{G}_{nm}$ and let*

$$\mathbf{c}(i, j) = \frac{c(i, j)}{\bar{c}}, \quad \tilde{\mathbf{c}}(i, j) = \frac{\tilde{c}(i, j)}{\tilde{\bar{c}}}, \quad (3.33)$$

be the normalized CTDs. Let $F_{\mathbf{c}}$ and $\tilde{F}_{\mathbf{c}}$ be their corresponding eCDFs. Where $\bar{c}$ and $\tilde{\bar{c}}$ are the mean CTDs of G and $\tilde{G}$ respectively. Define the mean-normalized SAGD as

$$\text{SAGD}(G, \tilde{G}) := W_1(F_{\mathbf{c}}, \tilde{F}_{\mathbf{c}}), \quad (3.34)$$

In all applications presented in the remainder of the thesis, the eCDFs are normalized by the mean CTD.

3.3.4. Representational Invariances

Even though $(\mathbb{G}_{nm}, \text{SAGD})$ defines only a pseudo-metric space, the proposed distance still encodes meaningful geometric information about the underlying structure of the data. In particular, SAGD reflects properties of the intrinsic manifold induced by the graph without requiring any explicit assumptions about its form. Rather than comparing graphs through their individual components, SAGD operates at a distributional level by quantifying differences between the eCDFs of CTDs. While SAGD is strictly a *distance* metric—where a smaller value indicates closer structural alignment—it serves the same diagnostic purpose as a representational similarity function by capturing the statistical divergence between representations.

Within the context of representation learning in neural networks, the question of how to evaluate representations has been extensively studied [2, 1, 4]. A key consideration is how an evaluation metric behaves under transformations that leave the functional behavior of a model unchanged. Consider a representation $\mathbf{R}$ followed by a fully-connected layer of the form $f(\mathbf{R}) = \sigma(\mathbf{R}\mathbf{W} + \beta)$. If one applies a transformation $\mathbf{R}' = \mathbf{R}A$, where A is an invertible matrix, and simultaneously transforms the weights to $\mathbf{W}' = A^{-1}\mathbf{W}$, the resulting function remains unchanged, meaning $f(\mathbf{R}) = f(\mathbf{R}')$. This observation implies that $\mathbf{R}$ and $\mathbf{R}'$ should be regarded as equivalent representations, as optimization procedures can absorb such linear transformations into the parameters [2, 1].

This functional equivalence motivates the requirement that any metric evaluating representations must be insensitive to certain classes of transformations. Initially, invariance under general invertible linear transformations was proposed as a desirable property for similarity indexes [2]. However, it has been argued that this requirement may be overly restrictive [1]. A more appropriate condition is invariance under orthogonal transformations of the feature space, which accommodates the inherent symmetries present in neural network parameterizations, such as arbitrary neuron permutations [4, 20, 1]. SAGD satisfies this requirement during the graph construction phase: because edge weights are derived from pairwise Euclidean distances between representations, any orthogonal transformation applied to the feature space leaves the resulting graph topology entirely unchanged.

Furthermore, any graph-based metric must also be invariant to permutations of the observations, meaning the evaluation should not depend on arbitrary node labeling. SAGD achieves this naturally through its statistical formulation. By evaluating the empirical distributions (eCDFs) of the Commute Time Distances rather than relying on node-to-node alignments, SAGD ensures that reordering the input dataset does not alter the measured structural dissimilarity.

Another relevant property is *invariance to isotropic scaling*. For a distance metric d , this implies $d(\mathbf{R}, \tilde{\mathbf{R}}) = d(\alpha\mathbf{R}, \beta\tilde{\mathbf{R}})$ for any $\alpha, \beta \in \mathbb{R}^+$. While SAGD does not explicitly

enforce this invariance at the level of the metric’s definition, it is effectively induced through the construction of the graph representation from the data. Following the perspective in [6], each node v_i can be interpreted as a representation of an observation $\mathbf{r}_h(x_i) \in \mathbb{R}^{J^{(h)}}$.

Graph Construction

Once this correspondence is established, a weighted undirected graph is constructed by defining edges and assigning weights between pairs of nodes. To ensure the construction of a sparse graph that accurately captures the underlying geometry of the data, the following procedure is adopted:

1. Initialize with $k = 5$, and connect each data point to its k -nearest neighbors (kNN) mutually.
2. If the resulting graph is connected, fix it as the graph representation. Otherwise, increment k by 1 and repeat step 1.

This process finds a proper adjacency matrix; nonetheless, the edge weights must still be assigned. A common approach is to define these weights by applying a similarity kernel to the pairwise distances between connected vertices in the latent space. Different kernel functions induce different weighting schemes, determining how rapidly the influence between neighboring vertices decreases with distance and, consequently, affecting the local and global properties of the resulting graph. Table 3.1 summarizes several commonly used similarity functions that can be employed to assign edge weights in connected sparse graphs.

Among the weighting schemes presented in Table 3.1, the Gaussian kernel is adopted throughout the experiments. This choice is motivated by both theoretical and practical considerations. First, the Gaussian kernel strongly emphasizes local neighborhoods by assigning rapidly decaying weights as the Euclidean distance between neighboring representations increases. Consequently, the resulting weighted graph reflects the local geometry of the representation while still preserving connectivity through the mutual kNN construction. The objective of this thesis is not to compare alternative graph weighting schemes but to evaluate the proposed SAGD under a standard and well-established graph construction procedure. Using the Gaussian kernel therefore isolates the behavior of SAGD from additional modeling choices while remaining consistent with common practice in graph-based machine learning.

The previous procedure guarantees that the graph is connected by increasing the neighborhood size until connectivity is achieved. However, in some applications it may be desirable to work with a smaller value of k to preserve a more local geometric structure, even if the resulting graph is disconnected. In such cases, the connected components can be linked through a simple post-processing step. Starting from the disconnected graph,

Table 3.1: Examples of similarity functions for assigning edge weights in connected sparse graphs.

Similarity	Weights	Characteristics
Gaussian (RBF)	$w_{ij} = \exp\left(-\frac{\ \mathbf{r}_h(x_i) - \mathbf{r}_h(x_j)\ _2^2}{2\sigma^2}\right)$	Produces highly localized edge weights, strongly emphasizing nearby neighbors while rapidly attenuating long-range connections. Commonly used in weighted kNN and manifold graphs.
Laplacian	$w_{ij} = \exp\left(-\frac{\ \mathbf{r}_h(x_i) - \mathbf{r}_h(x_j)\ _2}{\sigma}\right)$	Assigns larger weights to moderately distant neighbors than the Gaussian kernel, improving connectivity while preserving local structure.
Exponential	$w_{ij} = \exp(-\lambda\ \mathbf{r}_h(x_i) - \mathbf{r}_h(x_j)\ _2)$	Equivalent to the Laplacian under a different parameterization; provides a gradual decay of edge weights that balances locality and connectivity.
Inverse Distance	$w_{ij} = \frac{1}{1 + \ \mathbf{r}_h(x_i) - \mathbf{r}_h(x_j)\ _2}$	Maintains non-negligible weights over longer distances, yielding graphs with stronger global connectivity.
Inverse Quadratic	$w_{ij} = \frac{1}{1 + \frac{\ \mathbf{r}_h(x_i) - \mathbf{r}_h(x_j)\ _2^2}{c}}$	Assigns smoothly decaying weights with heavier tails than the Gaussian kernel, preserving influence among moderately distant vertices.
Rational Quadratic	$w_{ij} = \left(1 + \frac{\ \mathbf{r}_h(x_i) - \mathbf{r}_h(x_j)\ _2^2}{2\alpha l^2}\right)^{-\alpha}$	Acts as a multi-scale similarity function, allowing edge weights to capture both local and intermediate-range relationships.
Cauchy Kernel	$w_{ij} = \frac{1}{1 + \frac{\ \mathbf{r}_h(x_i) - \mathbf{r}_h(x_j)\ _2^2}{\sigma^2}}$	Heavy-tailed weighting function that preserves relatively strong connections between distant neighbors, increasing graph robustness.
Student- t Kernel	$w_{ij} = \left(1 + \frac{\ \mathbf{r}_h(x_i) - \mathbf{r}_h(x_j)\ _2^2}{\nu}\right)^{-\frac{\nu+1}{2}}$	Heavy-tailed kernel that assigns appreciable weights to distant vertices while maintaining local neighborhood structure.

the pair of observations belonging to different connected components with the smallest Euclidean distance is identified, and an edge is added between them. This operation is repeated iteratively, each time merging two connected components, until a single connected graph is obtained. The resulting graph contains the minimum number of additional edges required to ensure connectivity while preserving the local neighborhood relationships induced by the chosen value of k .

The effect of isotropic scaling is controlled by the choice of kernel hyperparameters. In practice, these parameters are often selected in a data-dependent manner. For instance, in the case of the Cauchy kernel; for example, setting σ as the median of the kNN distances provides an automatic normalization that compensates for global scaling of the data. A similar mechanism applies to the Gaussian kernel.

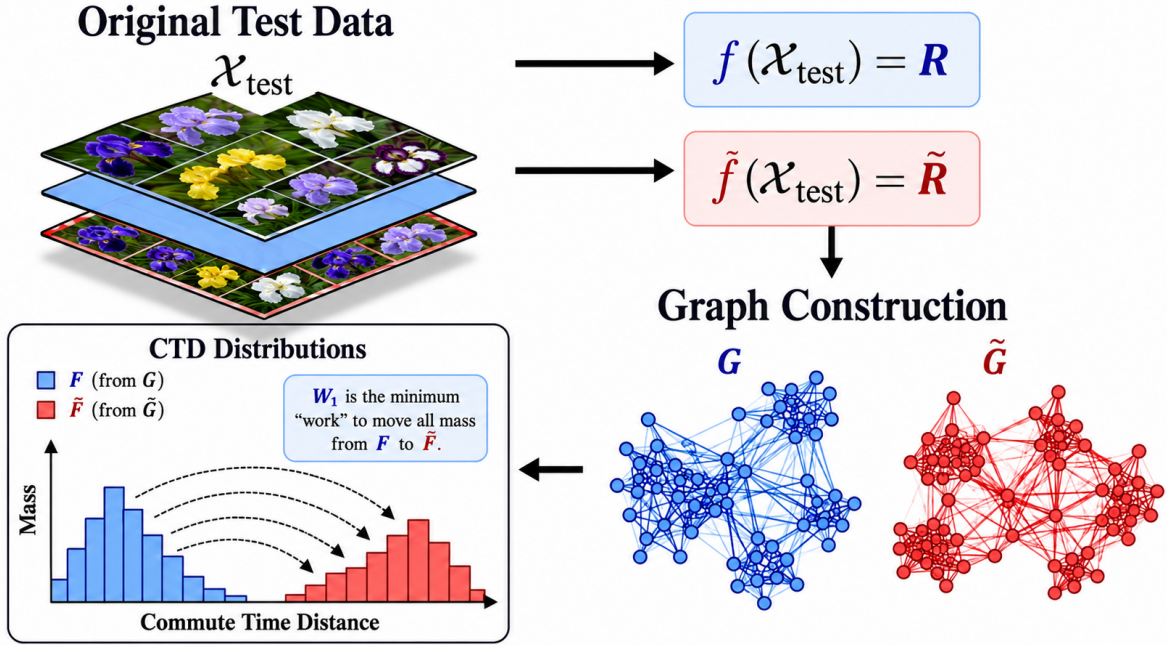


Figure 3.3: **Overview of the SAGD Pipeline.** The diagram illustrates the analytical procedure beginning with the test dataset $\mathcal{X}_{\text{test}}$ (top left). Two non-linear transformations of the same dataset are evaluated, producing the representations $\mathbf{R}$ (blue box) and $\tilde{\mathbf{R}}$ (red box) through the mappings $f(\mathcal{X}_{\text{test}}) = \mathbf{R}$ and $\tilde{f}(\mathcal{X}_{\text{test}}) = \tilde{\mathbf{R}}$, respectively. These representations are then used to construct two weighted graph structures, denoted by G (blue) and $\tilde{G}$ (red). The lower-left panel illustrates the corresponding distributions of CTDs associated with each graph, represented by the empirical cumulative distribution functions F and $\tilde{F}$. The dashed arrows visualize the Earth Mover’s interpretation of the 1-Wasserstein distance (W_1), corresponding to the minimum “work” required to transport the probability mass from F to $\tilde{F}$. In this way, SAGD quantifies the structural divergence between the two graph representations by measuring the optimal transportation cost between their CTD distributions.

Consequently, although SAGD is not inherently scale-invariant, this property is effectively achieved through the adaptive behavior of the kernel used in the graph construction.

Ultimately, while SAGD operates as a measure of distance rather than a similarity score, it successfully fulfills the invariance criteria required to evaluate neural network representations. Following the transformations

$$\mathcal{X} \mapsto \mathbf{R} \mapsto G \mapsto F \quad (3.35)$$

(original data, representation, graph representation, eCDF representation), for two different representations of $\mathcal{X}$, $\mathbf{R}$ and $\tilde{\mathbf{R}}$, the following evaluation holds:

$$\text{SAGD}(G, \tilde{G}) = W_1(F, \tilde{F}) \propto d(\mathbf{R}, \tilde{\mathbf{R}}). \quad (3.36)$$

By strictly quantifying structural divergence rather than simple correlation, SAGD establishes a robust foundation for evaluating the non-linear representations of $\mathcal{X}_{\text{test}}$. Yet, evaluating these pairwise distances is only part of the analytical challenge. In this thesis the *robustness* of the SAGD will be evaluated and to fully map the learning procedures of a neural network, these high-dimensional divergences must be projected into a comprehensible, low-dimensional visual space. Standard embedding algorithms often fail in this regard because they force Euclidean assumptions onto curved data manifolds. Therefore, to accurately visualize the topological shape preserved by SAGD, we require a visualization method that natively understands graph-based distances.

3.4. Shape-Aware Stochastic Neighbor Embedding

The visualization method employed in this thesis is the Shape-Aware Stochastic Neighbor Embedding (SASNE), [13]. The method is derived as a generalization of t-SNE [15] using graph distances. Furthermore, to understand t-SNE and SASNE, it is important to mention their predecessor SNE, [14].

Embedding methods aim to represent high-dimensional data $\{x_i\}_{i=1}^n \subset \mathbb{R}^D$ in a low-dimensional space $\{y_i\}_{i=1}^n \subset \mathbb{R}^d$ (with $d \ll D$), while preserving meaningful structural relationships. In particular, they attempt to maintain neighborhood similarities so that nearby points in the original space remain close in the embedding.

SNE defines similarities in the high-dimensional space using conditional probabilities:

$$p_{j|i} = \frac{\exp\left(\frac{-\|x_i - x_j\|_2^2}{2\sigma_i^2}\right)}{\sum_{k \neq i} \exp\left(\frac{-\|x_i - x_k\|_2^2}{2\sigma_i^2}\right)}. \quad (3.37)$$

Similarly, in the low-dimensional space;

$$q_{j|i} = \frac{\exp\left(-\|y_i - y_j\|_2^2\right)}{\sum_{k \neq i} \exp\left(-\|y_i - y_k\|_2^2\right)}. \quad (3.38)$$

The embedding is obtained by minimizing the sum of Kullback–Leibler (KL) divergences:

$$C_{\text{SNE}} = \sum_i \text{KL}(P_i \| Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}. \quad (3.39)$$

Intuitively speaking, the KL divergence measures how different two probability distributions are, and its key intuition as a cost function is that it penalizes using one distribution to approximate another. SNE successfully captures local structure, but it has notable drawbacks. Initially, the cost function is asymmetric, optimization is difficult, and it suffers from the *crowding problem* [15], where the low-dimensional embedding becomes excessively concentrated. The t-SNE addresses these issues with two main modifications. First, it symmetrizes the probabilities by;

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}. \quad (3.40)$$

Second, it replaces the Gaussian kernel in the low-dimensional space with a heavy-tailed Student- t distribution, hence its name;

$$q_{ij} = \frac{(1 + \|y_i - y_j\|_2^2)^{-1}}{\sum_{k \neq l} (1 + \|y_k - y_l\|_2^2)^{-1}}. \quad (3.41)$$

The cost function becomes,

$$C_{\text{t-SNE}} = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}. \quad (3.42)$$

The heavy tails of the Student- t distribution alleviate the crowding problem by mapping moderate-to-large high-dimensional distances to significantly larger spatial distances in the low-dimensional embedding. This expansion pushes dissimilar points further apart, preventing distinct clusters from collapsing into a single dense region and thereby preserving their relative separation. This makes t-SNE highly effective at revealing cluster structure while maintaining local neighborhoods.

The hyperparameter σ_i is of great importance for this method, it establishes the *perplexity*. The perplexity controls the effective size of the local neighborhood around each point. It determines how the Gaussian bandwidth σ_i is chosen in the high-dimensional similarity distribution. A small σ_i introduces more importance to nearby points, while a large σ_i sets more consideration to wider neighborhoods. Any particular value of σ_i induces a probability distribution, P_i , over all of the other datapoints.

Instead of choosing σ_i directly, t-SNE specifies a desired perplexity. It is defined from the *Shannon entropy* of the conditional distribution P_i . The entropy is

$$\mathcal{H}(P_i) = - \sum_j p_{j|i} \log_2 p_{j|i}, \quad (3.43)$$

and the perplexity⁴ is then

$$\text{Perp}(P_i) = 2^{\mathcal{H}(P_i)}. \quad (3.44)$$

Choosing the perplexity can be a challenge depending on the geometry of the data. In practice, a value of 30 is commonly used. Nonetheless, it is recommended to increase depending on the sample size n .

Some shortcomings can appear when a small perplexity is chosen. In [21], it was pointed out that small perplexity can cause the low-dimensional embedding to fragment into artificial clusters that are not actually present in the original high-dimensional data. At the same time, broader relationships between groups of points may be distorted or disappear entirely, making it difficult to preserve large-scale or hierarchical organization in the embedding. On the other hand, both SNE and t-SNE rely on Euclidean distances $\|x_i - x_j\|_2$ in the original space. However, when data lies on a non-linear manifold—as occurs in our setting for neural network activations—Euclidean distances may fail to capture the intrinsic geometry. Points that are close along the manifold may appear far in Euclidean space, and vice versa.

SASNE mitigates some of these issues by extending t-SNE replacing Euclidean distance with a graph-based distance; the *Biharmonic distance* (BHD) [22] for the high-dimensional data.

A kNN graph is first constructed, and distances $b(i, j)$ are computed as a node-by-node distance. Having a lot of similar features with the CTD. The BHD also captures the structural geometry of the data without Euclidean assumptions over the latent space.

Definition 3.5 (Biharmonic Distance). *For a graph G and its respective Laplacian matrix L . The Biharmonic Distance $b(i, j)$ between two nodes v_i and v_j is defined by*

$$b(i, j) = (L^\dagger)_{ii}^2 + (L^\dagger)_{jj}^2 - 2(L^\dagger)_{ij}^2, \quad (3.45)$$

where $(L^\dagger)_{ij}^2$ means the (i, j) -entry of the squared Moore-Penrose pseudoinverse of the Laplacian matrix; $(L^\dagger)^2$.

One analogous description between the BHD and the CTD is how they both can be written using the eigenvalues and eigenvectors of L ; a spectral representation [23]. Following the same notation of section 3.2.2 one gets;

$$b(i, j) = \sum_{\alpha=2}^n \frac{(\mathbf{v}_{\alpha i} - \mathbf{v}_{\alpha j})^2}{\lambda_\alpha^2}. \quad (3.46)$$

⁴Perplexity can be interpreted as the effective number of neighbors. If one neighbor dominates: $p_{j|i} \approx 1$, then entropy is small and perplexity is near 1. If probabilities are spread uniformly across k neighbors: $p_{j|i} = \frac{1}{k}$, then $\mathcal{H}(P_i) = \log_2 k$, giving $\text{Perp}(P_i) = k$.

Here the BHD is the sum of squared subtractions between the eigenvalues which entries are corresponding to specific nodes. The weightes are the square of the inversed non-zero eigenvalues. In an analogous way, the CTD scale the sum by the volume of the graph and ignores the square of the weights;

$$c(i, j) = \text{vol}(G) \sum_{\alpha=2}^n \frac{(\mathbf{v}_{\alpha i} - \mathbf{v}_{\alpha j})^2}{\lambda_{\alpha}}. \quad (3.47)$$

The full proof for this two representations are deferred to Appendix A.3 and A.4. The BHD is then used to define probabilities;

$$p_{j|i} = \frac{\exp\left(\frac{-b(x_i, x_j)^2}{2\sigma_i^2}\right)}{\sum_{k \neq i} \exp\left(\frac{-b(x_i, x_k)^2}{2\sigma_i^2}\right)}. \quad (3.48)$$

Since the distance itself is shape-aware and meaningful at large scales, SASNE can then safely use a large neighborhood size, being automatically set to 90% of the sample size N . The rest of the formulation follows t-SNE, but with these shape-aware distances.

This shape-awareness is of particular importance for neural network representations, where after applying a graph construction to all representations the correspondent Representation graph family ($\mathbb{G}_{nm}$) is in a more complex space. SAGD captures shape information between individual networks, where each node corresponds to a single data point. However, it does not account entirely for the geometric structure of the broader Representation graph family, in which each node represents an entire network or representation. To capture this higher-level organization, SASNE is applied to learn the underlying shape of the Representation graph family. In such cases, SASNE can preserve both local and global structure more faithfully than t-SNE, leading to more meaningful visualizations.

Chapter 4

Results

The objective of this chapter is twofold. First, we investigate the robustness of the proposed SAGD under controlled experimental settings. Since SAGD is computed from weighted graph representations rather than directly from the data, it is essential to understand how sensitive the method is to the graph construction procedure and to stochastic variations in graph topology before applying it to the analysis of neural network representations. Second, after establishing these robustness properties, we employ SAGD to study the evolution of internal representations throughout the training of a deep neural network. The chapter is organized into three parts.

The first part focuses on the *parameter selection* involved in the graph construction procedure. Recall that constructing the representation graph requires selecting both the neighborhood size used in the mutual k -nearest-neighbor graph and the kernel hyperparameter used to assign edge weights. Since these quantities directly determine the resulting graph topology and its associated CTD, an important practical question is whether moderate perturbations of these parameters lead to substantial changes in the resulting SAGD values.

To address this question, the analysis is developed progressively through three complementary experiments. We begin with a simple illustrative example based on the Iris dataset, where the neighborhood size k and the Gaussian kernel bandwidth are varied to provide an intuitive visualization of the stability of the graph construction procedure. We then consider a synthetic Gaussian dataset ($n = 500$, $d = 3$) and systematically evaluate the effect of parameter perturbations under a controlled setting where the underlying data distribution is known. Finally, we extend the analysis to a substantially more challenging synthetic setting consisting of a high-dimensional mixture of five Gaussian distributions ($n = 500$, $d = 10000$). This final experiment provides a more realistic assessment of the robustness of the automatic parameter selection procedure in the presence of complex geometric structure and high ambient dimensionality.

The second part of the chapter investigates the robustness of SAGD with respect to stochastic perturbations of graph topology. For this purpose, random graphs generated

from the Watts–Strogatz small-world model [24] are considered. By continuously varying the rewiring probability, we evaluate whether SAGD remains capable of identifying the underlying structural regime of a graph despite the stochastic variability inherent in its generation.

Having established the robustness of the proposed methodology, the final part of the chapter presents the primary application of this thesis: the analysis of neural network representation dynamics. Representations extracted from different layers and training epochs of a ResNet-18 model are compared using SAGD, and the resulting pairwise distances are visualized through SASNE embeddings. These experiments provide a geometric characterization of how internal representations evolve, stabilize, and progressively organize throughout the optimization process.

4.1. Graph Construction

The first experiment investigates the sensitivity of SAGD to the two parameters involved in the construction of the weighted graph representation: the neighborhood size k used to define the mutual kNN graph and the Gaussian kernel bandwidth σ employed to assign edge weights. Since both parameters influence the resulting graph topology and, consequently, the CTD, it is important to evaluate whether moderate perturbations around their automatically selected values produce substantial changes in the resulting SAGD.

For this purpose, the classical Iris dataset [16] is considered as a simple and well-understood benchmark. The dataset consists of 150 observations described by four numerical features corresponding to sepal and petal measurements from three different Iris species. Owing to its low dimensionality and interpretable geometric structure, the dataset provides a convenient setting for visualizing the influence of the graph construction parameters before extending the analysis to more challenging synthetic datasets.

The automatic graph construction procedure described in Section 3.3.4 is first applied to the dataset in order to determine the reference graph. This procedure identifies the smallest neighborhood size that produces a connected mutual kNN graph and subsequently estimates the Gaussian kernel bandwidth as the median of the corresponding kNN-distance distribution. For the Iris dataset, the resulting reference parameters are

$$k^* = 12, \quad \sigma^* = 0.374, \quad (4.1)$$

where σ^* corresponds to the median edge kNN distance of the reference graph.

The sensitivity analysis is divided into two complementary experiments. In the first experiment, the Gaussian bandwidth is kept fixed at its reference value σ^* while the neighborhood size is varied from $k^* - 1$ to $k^* + 9$. For each value of k , a new weighted

graph is constructed and its SAGD with respect to the reference graph is computed.

Since k^* is defined as the smallest neighborhood size producing a connected mutual kNN graph, the graph obtained with $k = 11$ is no longer connected. As discussed in Chapter 3, the computation of the CTD requires the graph to consist of a single connected component. Therefore, whenever a tested value of k produces a disconnected graph, the connected components are merged by iteratively adding an edge between the pair of nodes belonging to different components with the smallest Euclidean distance. This procedure is repeated until a single connected graph is obtained while introducing the minimum number of additional edges. Consequently, every graph considered in the sensitivity analysis admits a valid CTD embedding while preserving, as much as possible, the topology induced by the selected neighborhood size.

In the second experiment, the neighborhood size is fixed at the reference value k^* while the Gaussian bandwidth is varied. The explored interval is defined by the 5th and 95th percentiles of the edge-distance distribution obtained from the reference graph, thereby excluding extreme edge lengths that occur only rarely. Twenty-five equally spaced values are evaluated within this interval, including the reference bandwidth itself. As in the previous experiment, each modified graph is compared with the reference graph using SAGD.

Figure 4.1 summarizes the results. Panel (A) shows the empirical distribution of kNN distances in the reference graph. The shaded region indicates the interval between the 5th and 95th percentiles, while the red vertical line denotes the median kNN distance selected as the reference bandwidth σ^* . Panel (B) presents the evolution of SAGD as the Gaussian bandwidth is varied while keeping the neighborhood size fixed at k^* . Panel (C) shows the corresponding sensitivity with respect to the neighborhood size while fixing the Gaussian bandwidth at σ^* . In both cases, the dashed vertical line indicates the automatically selected reference parameter.

The results presented in Figure 4.1 indicate that the variations in SAGD induced by changes in either the neighborhood size or the Gaussian bandwidth are relatively small in magnitude. In particular, the sensitivity with respect to the neighborhood size is minimal, while moderate perturbations of the Gaussian bandwidth produce only gradual changes around the automatically selected reference value. Nevertheless, these observations remain qualitative, as SAGD has no intrinsic scale that allows its numerical values to be interpreted in absolute terms. Consequently, although the measured variations appear small, it is not yet possible to determine whether they are negligible or substantial without an external point of reference. To address this limitation, the following experiments introduce controlled synthetic settings in which meaningful reference distances can be established, providing a more principled interpretation of the magnitude and robustness of SAGD.

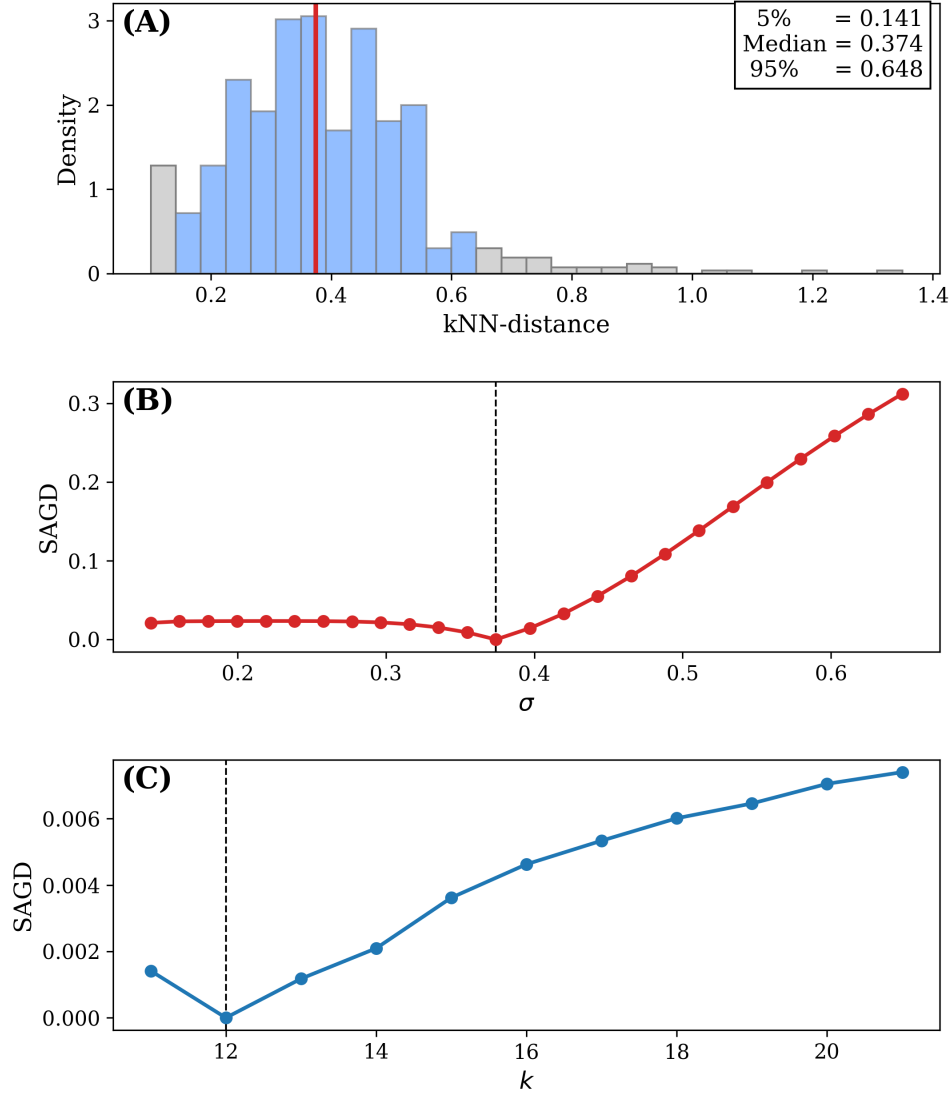


Figure 4.1: **Sensitivity of SAGD to graph construction parameters on the Iris dataset.** (A) Empirical distribution of kNN distances in the reference mutual kNN graph. The blue shaded region corresponds to the interval between the 5th and 95th percentiles of the edge-distance distribution, while the red vertical line indicates the median kNN distance used as the reference Gaussian kernel bandwidth σ^* . (B) SAGD between the reference graph and graphs constructed with fixed neighborhood size $k = k^*$ while varying the Gaussian kernel bandwidth σ over the interval shown in panel (A). (C) SAGD between the reference graph and graphs constructed with fixed Gaussian bandwidth $\sigma = \sigma^*$ while varying the neighborhood size from $k^* - 1$ to $k^* + 9$. In both experiments, the dashed vertical line indicates the automatically selected reference parameter used to construct the reference graph.

4.1.1. Synthetic Gaussian

The previous experiment on the Iris dataset suggested that moderate perturbations of the graph construction parameters produce only small variations in SAGD. However, since SAGD has no intrinsic numerical scale, these observations remain qualitative. A value that appears numerically small may nevertheless correspond to a meaningful structural modification. Consequently, before considering more challenging datasets, it is necessary to establish a reference framework that allows the magnitude of the observed SAGD values to be interpreted.

To this end, we consider a controlled synthetic experiment in which the underlying probability distribution is completely known. The objective is to evaluate the robustness of the graph construction procedure with respect to its hyperparameters while simultaneously providing a statistical reference for the expected variability of SAGD.

Before performing the robustness analysis, the graph construction parameters are calibrated using independent realizations of the Gaussian model. Specifically, 500 datasets

$$X_j = \{x_1, \dots, x_n\}, \quad x_i \sim \mathcal{N}(0, I_3), \quad j = 1, \dots, 500, \quad (4.2)$$

are generated independently, each containing $n = 500$ observations. For every realization, the graph construction procedure described in Section 3.3.4 is applied to determine

- the smallest neighborhood size producing a connected mutual kNN graph, and
- the Gaussian kernel bandwidth obtained from the median mutual kNN edge distance.

The resulting empirical distributions provide estimates of the graph construction parameters. Throughout the remainder of the experiment, the reference parameters

$$(k^*, \sigma^*) \quad (4.3)$$

are defined as the empirical means of these calibration distributions. Using the reference parameters, an empirical distribution describing the natural variability of SAGD is constructed. For each Monte Carlo replication, two independent datasets

$$X_i, Y_i \stackrel{\text{i.i.d.}}{\sim} \mathcal{N}(0, I_3), \quad i = 1, \dots, m, \quad (4.4)$$

are generated independently, in this case $m = 500$. Graphs are constructed from both datasets using the fixed parameters (k^*, σ^*) , and the corresponding SAGD values

$$D_i = \text{SAGD}(G(X_i; k^*, \sigma^*), G(Y_i; k^*, \sigma^*)) \quad (4.5)$$

form an empirical reference distribution. The 5th and 95th percentiles of this distribution define the reference interval representing the structural variability expected between inde-

pendent realizations of the same probability distribution under identical graph construction parameters.

The robustness analysis is then performed by perturbing the graph construction parameters. For every candidate pair (k, σ) , independent Gaussian datasets

$$X_i, Y_i \stackrel{\text{i.i.d.}}{\sim} \mathcal{N}(0, I_3) \quad (4.6)$$

are again generated. The graph associated with X_i is always constructed using the reference parameters (k^*, σ^*) , whereas the graph associated with Y_i is constructed using the perturbed parameters (k, σ) . The corresponding structural discrepancy is therefore

$$S_i(k, \sigma) = \text{SAGD}(G(X_i; k^*, \sigma^*), G(Y_i; k, \sigma)), \quad (4.7)$$

and its empirical expectation is estimated by

$$\widehat{\mathbb{E}}[S(k, \sigma)] = \frac{1}{m} \sum_{i=1}^m S_i(k, \sigma). \quad (4.8)$$

This expectation quantifies the average structural deviation introduced solely by modifying the graph construction parameters. The resulting value is interpreted relative to the previously constructed reference distribution. Specifically, if

$$\widehat{\mathbb{E}}[S(k, \sigma)] \notin [Q_{0.05}(D), Q_{0.95}(D)], \quad (4.9)$$

where $Q_{0.05}(D)$ and $Q_{0.95}(D)$ denote the empirical 5th and 95th percentiles of the reference distribution, then the perturbation is considered to produce a structural discrepancy exceeding the variability naturally observed between independent samples from the same Gaussian distribution.

Figure 4.2 summarizes the calibration stage of the experiment. The graph construction procedure was applied to 500 independent realizations of the Gaussian model in order to estimate the reference graph construction parameters. Panel (A) shows the empirical distribution of the smallest neighborhood size required to obtain a connected mutual kNN graph, while panel (B) shows the corresponding distribution of Gaussian kernel bandwidths estimated from the median mutual kNN edge distance. The dashed vertical line indicates the empirical mean adopted as the reference parameter in each case. The calibration produced the reference values

$$k^* = 9, \quad \sigma^* = 0.422. \quad (4.10)$$

Using these reference parameters, the empirical reference distribution described above was obtained from independent Gaussian samples generated according to the same probability model. The resulting distribution defines the range of SAGD values expected solely

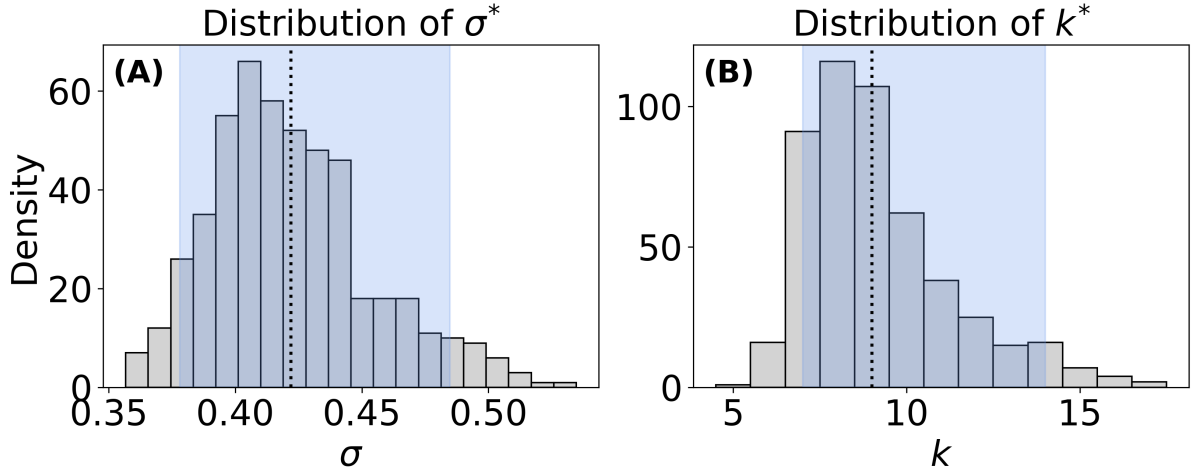


Figure 4.2: **Calibration of the graph construction parameters for the synthetic Gaussian experiment.** (A) Empirical distribution of the Gaussian kernel bandwidth obtained from the median mutual kNN edge distance. (B) Empirical distribution of the smallest neighborhood size producing a connected mutual kNN graph over 500 independent Gaussian realizations. In both panels, the dashed vertical line indicates the empirical mean adopted as the reference graph construction parameter, while the shaded region corresponds to the empirical 5th and 95th percentiles.

from sampling variability when identical graph construction parameters are employed. Consequently, the empirical 5th and 95th percentiles provide a natural reference interval against which the perturbation experiments can be interpreted.

The results of the perturbation analysis are presented in Figure 4.3. In both panels, the black curve corresponds to the empirical expectation $\hat{\mathbb{E}}[S(k, \sigma)]$, while the shaded region represents the empirical reference interval obtained from the reference distribution. The horizontal dashed line indicates the mean SAGD under the reference model, and the red marker identifies the calibrated graph construction parameter.

Panel (A) shows the effect of varying the Gaussian kernel bandwidth while keeping the neighborhood size fixed at k^* . The expected SAGD is smallest in the vicinity of the calibrated bandwidth and increases as the bandwidth is reduced. In contrast, increasing the bandwidth beyond the calibrated value produces only a moderate decrease before the curve stabilizes. Throughout the explored range, the expected SAGD remains within the empirical reference interval, indicating that the structural differences introduced by these perturbations are comparable to those naturally observed between independent Gaussian realizations.

Panel (B) presents the corresponding analysis for the neighborhood size while fixing the Gaussian kernel bandwidth at σ^* . As additional neighbors are incorporated into the mutual kNN graph, the expected SAGD increases gradually before reaching an approximately stable level. As in the previous experiment, the entire curve remains inside the empirical reference interval, suggesting that moderate changes in the neighborhood size do not

introduce structural discrepancies beyond the variability expected from independent samples generated by the same distribution.

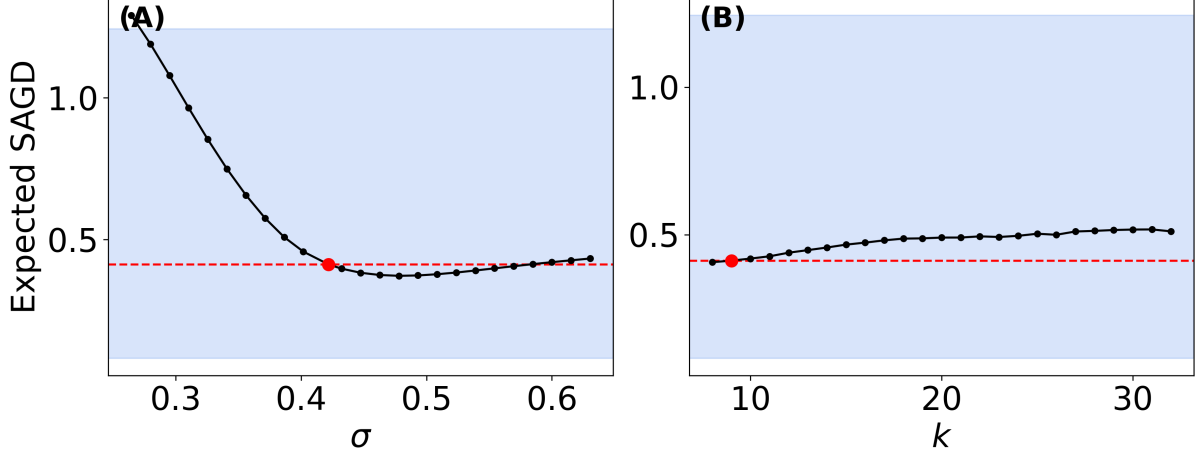


Figure 4.3: **Sensitivity of SAGD to graph construction parameters for the synthetic Gaussian experiment.** (A) Expected SAGD obtained by varying the Gaussian kernel bandwidth while fixing the neighborhood size at the calibrated value k^* . (B) Expected SAGD obtained by varying the neighborhood size while fixing the Gaussian kernel bandwidth at σ^* . In both panels, the black curve represents the empirical expectation over the Monte Carlo replications, the shaded region denotes the empirical 5th–95th percentile interval of the reference SAGD distribution, the horizontal dashed line indicates its mean, and the red marker identifies the calibrated reference parameter.

Overall, the synthetic Gaussian experiment provides a statistical interpretation of the magnitude of SAGD values. Unlike the Iris experiment, where only qualitative conclusions could be drawn, the reference distribution obtained here allows the effect of graph construction perturbations to be evaluated relative to the variability naturally expected between independent samples from the same probability distribution. Within the range of perturbations considered, the expected SAGD remains inside this reference interval, providing empirical evidence that the proposed graph construction procedure is robust to moderate changes in both the neighborhood size and the Gaussian kernel bandwidth.

4.1.2. High Dimensional Mixture

The previous experiment considered observations sampled from a single low-dimensional Gaussian distribution, corresponding to a single approximately spherical cloud of points. Although this setting provides a useful baseline for studying the behavior of the graph construction procedure, many practical datasets exhibit multiple groups or clusters rather than a single homogeneous population. To investigate the robustness of the calibration procedure under a more structured data distribution, the experiment is repeated using a balanced mixture of five Gaussian components.

Specifically, observations are generated from the mixture

$$X \sim \frac{1}{5} \sum_{r=1}^5 \mathcal{N}(\mu_r, I_d),$$

where the five component means are given by

$$\mu_1 = \mu(1, \dots, 1), \quad \mu_2 = -\mu(1, \dots, 1),$$

$$\mu_3 = \mu(1, \dots, 1, -1, \dots, -1),$$

$$\mu_4 = -\mu(1, \dots, 1, -1, \dots, -1), \text{ and } \mu_5 = \{0, 0, \dots, 0\},$$

with $\mu = 2$. Each component has covariance matrix I_d , yielding five well-separated clusters in a $d = 10\,000$ dimensional ambient space. As in the previous experiment, each simulated dataset contains $n = 500$ observations, and the calibration procedure is repeated over 500 independent Monte Carlo realizations.

Figure 4.4 summarizes the resulting distributions of the automatically selected graph parameters. The reference neighborhood size is centered around $k^* \approx 47$, while the reference bandwidth is highly concentrated around $\sigma^* \approx 140.5$. Although the underlying distribution is considerably more complex than the single Gaussian considered previously, the bandwidth parameters exhibit relatively small variability across Monte Carlo replications in comparison with the previous experiment.

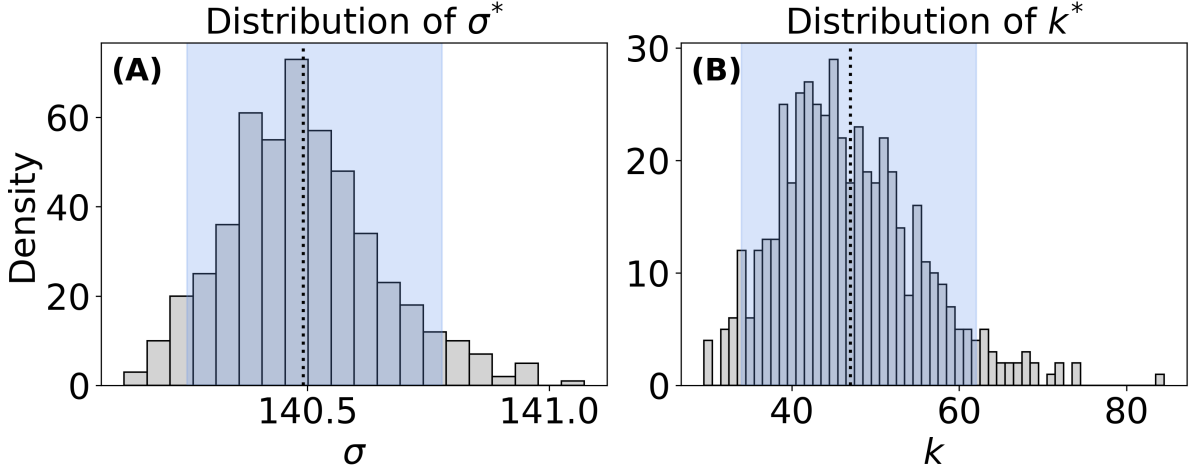


Figure 4.4: **Empirical distributions of the automatically calibrated graph parameters.** Over 500 independent realizations of the five-component Gaussian mixture in dimension $d = 10\,000$. Panel (A) displays the corresponding kernel bandwidth σ^* , while (B) shows the distribution of the selected neighborhood size k^* . The shaded regions indicate the empirical 5%–95% intervals, and the dashed vertical lines denote the corresponding sample means used as reference parameters in the subsequent perturbation analysis.

The corresponding perturbation experiment is shown in Figure 4.5. Panel (A) displays the expected SAGD as a function of the kernel bandwidth while keeping the neighborhood size fixed at k^* . As in the previous experiment, the curve is smooth and convex, attaining

its minimum at the calibrated bandwidth σ^* . Furthermore, the expected SAGD remains almost entirely within the empirical 5%–95% reference interval for all considered perturbations, demonstrating that the method is relatively insensitive to moderate changes in the kernel bandwidth.

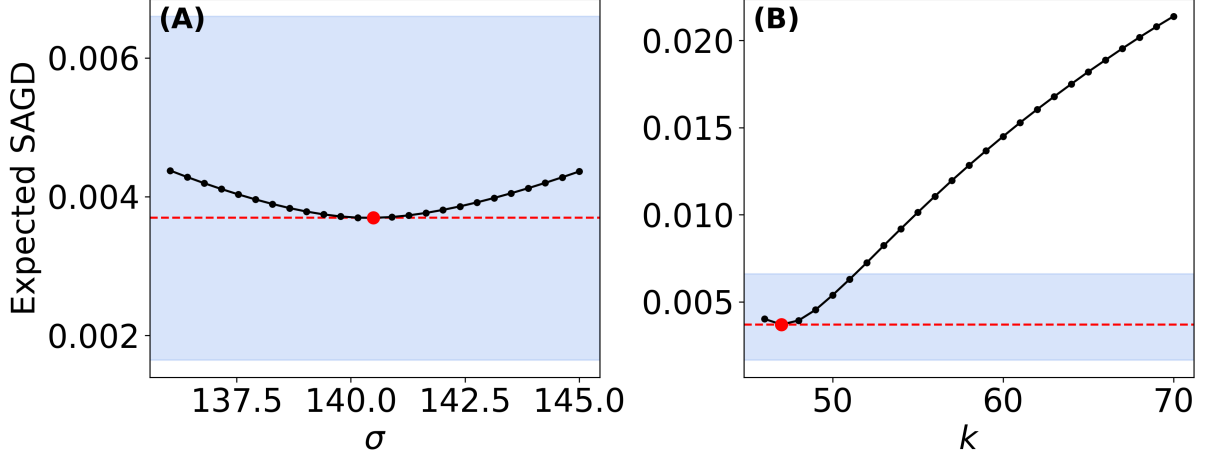


Figure 4.5: Expected SAGD under perturbations of the graph construction parameters for the five-component Gaussian mixture. Panel (A) varies the kernel bandwidth while fixing the neighborhood size at k^* , whereas Panel (B) varies the neighborhood size while fixing the bandwidth at σ^* . The red dashed line represents the mean reference SAGD, the shaded region denotes the empirical 5%–95% reference interval, and the red marker identifies the calibrated parameter selected during the automatic graph construction procedure.

In contrast, Panel (B) reveals a substantially stronger dependence on the neighborhood parameter. While the minimum again occurs at the calibrated value k^* , the expected SAGD increases rapidly as larger values of k are considered, leaving the empirical reference interval after only a small perturbation. As additional neighbors are incorporated, the graph progressively introduces connections between distinct Gaussian components, producing more pronounced changes in the graph Laplacian and therefore in the corresponding CTD representation.

Compared with the single Gaussian experiment, several important observations emerge. First, the calibration procedure remains equally stable despite the significantly more complex data-generating distribution, demonstrating that the automatic selection of (k^*, σ^*) is robust to multimodal data. Second, the expected SAGD exhibits essentially the same behavior with respect to the kernel bandwidth, with the calibrated value corresponding to the minimum of the perturbation curve. Finally, the dependence on the neighborhood size is considerably stronger than in the single Gaussian setting. Whereas increasing k in a single approximately spherical cloud primarily adds nearby edges, larger neighborhood sizes in the Gaussian mixture create an increasing number of connections between distinct clusters, substantially altering the graph geometry. Consequently, the choice of k becomes considerably more influential in preserving the intrinsic structure of

multimodal high-dimensional data.

4.2. Small-World Model

To evaluate the robustness of SAGD under stochastic perturbations of graph topology, we consider random graphs generated according to the Watts–Strogatz small-world model [24]. Also used in [25] for evaluating graph-by-graph distances; this family of graphs provides a convenient framework for studying the effect of structural randomness because it continuously interpolates between highly regular and highly random networks through a single rewiring probability parameter.

The generation procedure begins with a regular ring lattice in which every node is connected to its k nearest neighbors. Each edge is then independently rewired with probability p , while preserving the total number of edges in the graph. As the rewiring probability increases from 0 to 1, the resulting topology undergoes a gradual transition:

- I. $p = 0$ (*Fully Regular*): The graph remains a regular lattice, exhibiting strong local clustering but relatively long path lengths between distant nodes.
- II. *Small p (Small-World Regime)*: A small-world network is characterized by the simultaneous presence of a high clustering coefficient and a small characteristic path length. The clustering coefficient measures the tendency of neighboring vertices to also be connected to one another, forming tightly knit local communities, while the characteristic path length measures the typical number of intermediate nodes required to connect pairs of vertices. Consequently, a small-world network combines the strong local organization of a regular lattice with the efficient global connectivity of a random graph [24, 26].
- III. $p = 1$ (*Fully Random*): The graph becomes essentially random, maintaining short path lengths but losing the highly structured local neighborhoods present in the regular regime.

Let p_0 denote a reference rewiring probability. For each value

$$p_0 \in \{0.1, 0.3, 0.5, 0.9\}, \quad (4.11)$$

the experiment evaluates a collection of candidate probabilities $p \in [0, 1]$.

For every candidate probability p , m independent Monte Carlo replications are performed. During the i -th replication, two independent graphs are generated,

$$G_i^{(\text{ref})} \sim WS(n, k, p_0), \quad \text{and} \quad G_i \sim WS(n, k, p), \quad (4.12)$$

where $WS(n, k, p)$ denotes the Watts–Strogatz model with n vertices, neighborhood size k , and rewiring probability p . Connectivity of both graphs is verified, and disconnected realizations are discarded and regenerated to ensure that all comparisons are well defined.

Each graph is subsequently represent as its adjacency matrix. The corresponding CTD embeddings are computed, and the SAGD between the two graphs is evaluated, producing the quantity

$$S_i(p) = \text{SAGD} \left(G_i^{(\text{ref})}, G_i \right). \quad (4.13)$$

For every candidate probability, the expected SAGD is estimated by the Monte Carlo average

$$\hat{\mu}(p) = \frac{1}{m} \sum_{i=1}^m S_i(p). \quad (4.14)$$

In addition to the estimated mean, the empirical 5th and 95th percentiles of the Monte Carlo samples, $Q_{0.05}(p)$ and $Q_{0.95}(p)$, are computed and displayed as a shaded region in the corresponding figure. This interval characterizes the intrinsic variability of SAGD arising from the stochastic nature of the graph generation process.

If SAGD successfully captures the structural characteristics associated with the Watts–Strogatz model, then the expected distance should attain its minimum when the candidate probability coincides with the reference probability,

$$\hat{p} = \arg \min_p \hat{\mu}(p) \approx p_0. \quad (4.15)$$

Therefore, the experiment evaluates whether the minimum expected SAGD correctly recovers the true underlying rewiring probability despite the randomness inherent in the graph generation process.

Throughout this experiment, the number of vertices was fixed at ($n = 300$), while each vertex was initially connected to its ($k = 6$) nearest neighbors in the regular ring lattice from which the Watts–Strogatz graph is constructed. This parameters were held constant for all simulations so that the only varying factor was the rewiring probability (p). For every candidate probability, ($m = 500$) independent Monte Carlo simulations were performed. In each simulation, a new reference graph and a new comparison graph were independently generated, their SAGD was computed, and the resulting value was recorded, the results are presented in Figure 4.6 .

The results suggest that SAGD is robust to the structural variability inherent in the Watts–Strogatz graph generation process. Although two graphs generated with the same rewiring probability are never identical, the expected SAGD consistently reaches its minimum when the candidate probability matches the true underlying probability. In other words, the random fluctuations introduced during graph generation do not obscure the

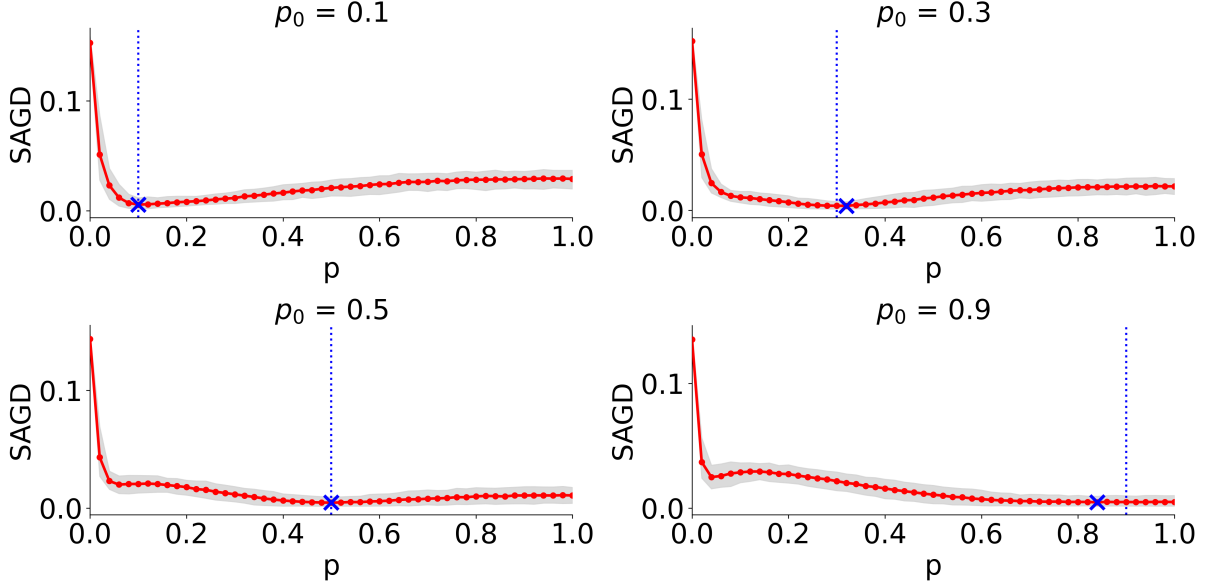


Figure 4.6: **SAGD evaluation on Watts–Strogatz small-world networks.** Each panel corresponds to a reference rewiring probability $p_0 \in \{0.1, 0.3, 0.5, 0.9\}$ with $n = 300$ and $k = 6$. For each candidate probability p , 500 independent graph pairs were sampled from $WS(n, k, p_0)$ and $WS(n, k, p)$, and their mean SAGD is shown by the red curve. The gray region denotes the empirical 5th–95th percentile interval, the blue dotted line indicates the true reference probability, and the blue cross marks the probability minimizing the expected SAGD.

structural information captured by the metric. Across all four scenarios, the minimum of the expected SAGD occurs very close to the true reference probability, with the negligible gap between the blue cross and the vertical blue dotted line indicating a high degree of agreement between the estimated and true values. The difference is negligible relative to the observed variability of SAGD, as the estimated minimum remains well within the corresponding 5th–95th percentile interval. Consequently, SAGD is able to distinguish the underlying topological regime while remaining stable across independent graph realizations, indicating that it effectively handles the stochastic variability associated with random graph construction.

4.3. Analysis of Training Dynamics

The previous sections established the robustness of SAGD with respect to the graph construction procedure and to stochastic perturbations of graph topology. Having verified that the proposed distance provides stable comparisons under these sources of variability, we now consider the primary application of this thesis: the analysis of representation dynamics throughout the training of a deep neural network.

As discussed in Section 2.2, the representations extracted from a neural network at different layers and optimization stages naturally define a Representation Graph Family,

where each graph corresponds to the geometric organization of the same test dataset under a particular layer and training epoch. Comparing these graphs using SAGD makes it possible to study statistical learning from a geometric perspective by monitoring how the structure of the representation space evolves during optimization.

The experiments are performed using a standard ResNet-18 architecture trained on the CIFAR-10 image classification dataset for 250 epochs. CIFAR-10 consists of 60,000 color images distributed across ten object categories, of which the complete 10,000 observation test set is used throughout the analysis. Each layer generates a temporal sequence of graphs belonging to the same Representation Graph Family. Throughout the remainder of this chapter, these representation extraction locations are referred to as *checkpoints*.

To characterize the evolution of these graph representations, pairwise SAGD values are computed between all representations within the family. The resulting distance matrix quantifies the structural dissimilarity between representations across both layers and training epochs. These distances are subsequently embedded into a low-dimensional space using SASNE, allowing the representational trajectories to be visualized while preserving the shape-aware geometry encoded by SAGD.

Rather than focusing solely on predictive performance, this analysis investigates how the internal geometry of the network develops throughout optimization. In particular, the experiments address several questions. How do representations evolve as training progresses? Do different layers stabilize at different stages of optimization? Is the evolution consistent with the bottom-up learning hypothesis [2], whereby earlier layers converge before deeper ones? Are representational changes gradual or characterized by distinct phases of reorganization? Finally, we investigate whether the graph-based perspective provided by SAGD reveals aspects of the learning process that are not apparent from conventional optimization metrics such as the training loss or classification accuracy.

To capture the hierarchical organization of the network, representations are extracted at six locations along the forward pass:

- **Input:** the raw CIFAR-10 images before entering the network.
- **Checkpoint 1:** the output after the first residual block group.
- **Checkpoint 2:** the output after the second residual block group.
- **Checkpoint 3:** the output after the third residual block group.
- **Checkpoint 4:** the output after the fourth residual block group.
- **Output:** the representation after the global average pooling layer.

Together, these checkpoints span the complete representational hierarchy learned by ResNet-18, from the original input space to the final feature representation used

for classification. Studying their evolution throughout training provides a geometric characterization of how increasingly abstract representations emerge and stabilize during optimization.

Learning Curves

Before analyzing the evolution of the learned representations, it is useful to briefly examine the optimization process of the underlying classifier. Although the primary objective of this thesis is not to study predictive performance, the learning curves provide important context for the subsequent representational analysis by verifying that the network undergoes a stable optimization process and converges to a high-performing solution. Consequently, any structural changes observed in the Representation Graph Family can be interpreted as genuine representational phenomena rather than artifacts arising from optimization instability.

Throughout training, the classification accuracy and cross-entropy loss were recorded on both the training and test sets. Simultaneously, representations of the complete CIFAR-10 test set were extracted every five epochs from the five trainable checkpoints described above. The input representation, being independent of the network parameters, was extracted only once. Consequently, the complete experiment produced a total of 251 representations, which together define the Representation Graph Family analyzed throughout the remainder of this chapter.

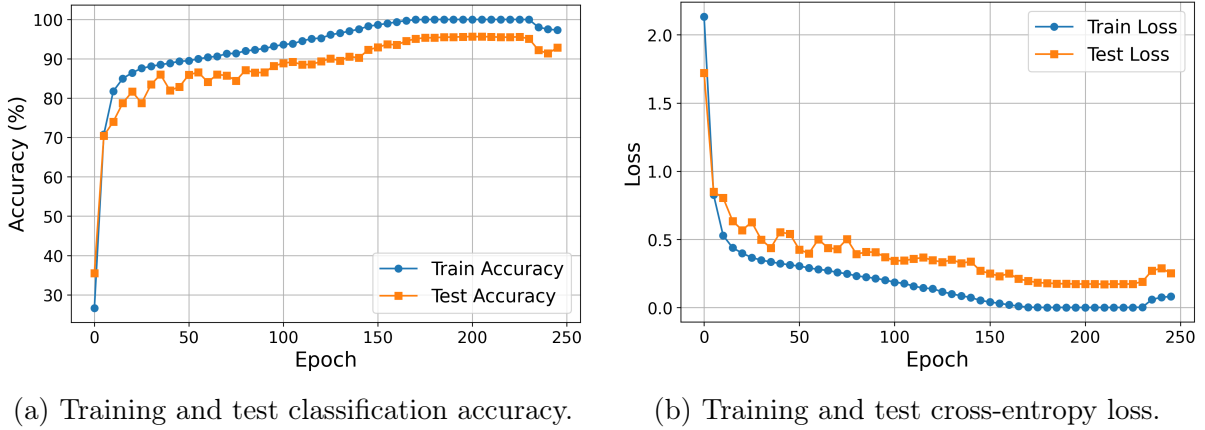


Figure 4.7: **Learning curves of the ResNet-18 model trained on CIFAR-10 for 250 epochs.** Panel (A) presents the evolution of the training and test classification accuracy, while panel (B) shows the corresponding cross-entropy losses throughout optimization. Representations were extracted every five epochs from the five trainable checkpoints, while the input representation was included once, producing a total of 251 representations that constitute the Representation Graph Family analyzed in the following sections. Overall, the learning curves indicate a stable optimization process that converges to a high-performing classifier, providing a reliable foundation for studying the evolution of its internal representations.

The optimization dynamics are summarized in Figure 4.7. During the initial stage of training, the network exhibits rapid improvement in both predictive performance and optimization objective. Within the first ten epochs, the training accuracy increases from approximately 27% to over 80%, while the test accuracy follows a similar trajectory, reaching roughly 70% after only five epochs. This rapid increase is accompanied by a sharp decrease in both training and test losses, indicating that the network quickly learns a set of discriminative low-level features and escapes the random initialization regime.

Following this initial stage, the optimization enters a considerably longer refinement phase. Between approximately epochs 10 and 170, both training and test accuracies continue to increase steadily, while the corresponding losses decrease smoothly. Unlike the rapid changes observed during the first epochs, the improvements during this interval become progressively smaller, suggesting that the network has already acquired a useful internal representation and is now refining it through incremental parameter updates. The absence of pronounced oscillations in either the loss or accuracy curves further indicates that the optimization procedure remains stable throughout training.

Around epoch 170, the classifier reaches an apparent convergence with respect to the optimization objective. The training accuracy approaches 100%, while the test accuracy stabilizes near 95–96%. Simultaneously, the training loss becomes nearly zero and the test loss attains an approximately constant minimum. Although optimization continues for several additional epochs, the predictive performance exhibits only marginal improvements, indicating that the network has largely converged from the perspective of the classification task.

During the final stage of training, after approximately epoch 230, a slight increase in the test loss accompanied by a modest decrease in test accuracy is observed while the training performance remains essentially unchanged. This slight increase is consistent with mild overfitting, as the network continues adapting to the training data without producing further improvements on previously unseen observations. Nevertheless, the magnitude of this effect remains relatively small, and the final model retains strong generalization performance on the CIFAR-10 test set.

From the perspective of this thesis, the principal role of Figure 4.7 is to establish that the optimization process is well behaved. More importantly, the learning curves provide a useful baseline against which the representational analysis can be interpreted. Conventional optimization metrics describe the evolution of predictive performance, whereas the Representation Graph Family captures the evolution of the geometry of the learned feature spaces.

All-checkpoint Embedding

Having established that the optimization process converges smoothly, we now turn to the evolution of the Representation Graph Family itself. Rather than examining each checkpoint separately, we begin by considering the complete family of representations simultaneously. This global view provides a geometric overview of the learning process and serves as the common reference frame for all subsequent analyses.

To visualize these relationships, the complete SAGD distance matrix is embedded into three dimensions using SASNE, producing a low-dimensional representation in which nearby points correspond to geometrically similar representation graphs.

Since each point simultaneously represents both a network checkpoint and a stage of training, the visualization employs a color scheme designed to encode these two sources of information. Each checkpoint is assigned a distinct color: the input representation is shown in gray and is shaped as a square, while Checkpoints 1–4 and the output representation are colored blue, red, green, purple, and yellow, respectively. Within each checkpoint, progressively lighter shades indicate increasing training epochs, while consecutive representations are connected by line segments to emphasize the temporal trajectory followed throughout optimization. To facilitate comparisons across checkpoints, the earliest trainable representation of every trajectory is shown in black, whereas the final representation is shown in gray, independently of the checkpoint color. This encoding allows both the architectural depth and the temporal evolution of the network to be interpreted simultaneously.

The resulting manifold immediately reveals several remarkable properties of the learning process. First, the trajectories associated with the different checkpoints exhibit distinct geometric patterns throughout training. Rather than following a common trajectory, each checkpoint develops along its own distinct path through the representation manifold. Interestingly, while the shapes of these trajectories differ noticeably between the shallow and deep stages—with Checkpoint 1 exhibiting a highly elongated structure compared to the compact dynamics of the deeper layers—their spatial locations do not strictly follow the architectural depth. Most notably, the all-checkpoint embedding reveals that Checkpoint 1 is positioned closely to the final output checkpoint within the space.

A second observation concerns the temporal evolution of these trajectories. Every shallower checkpoint (checkpoints 1, 2 and 3) exhibits a pronounced initial displacement from its earliest representation, visible as the long segment connecting the black marker to the first colored snapshots. This rapid geometric transition closely mirrors the behavior observed in the learning curves (Figure 4.7), where the largest improvements in classification accuracy and the steepest reduction in loss also occur during the first epochs of optimization. Thus, the period of fastest optimization coincides with the largest structural reorganization of the learned representation space.

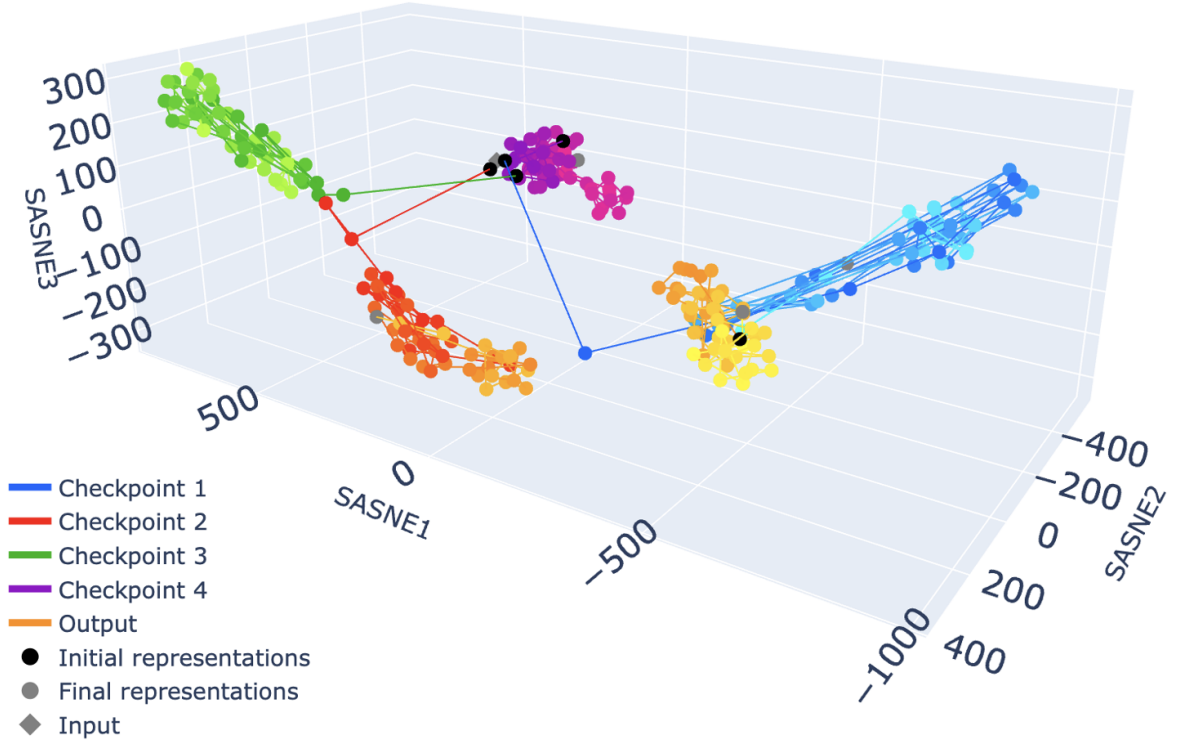


Figure 4.8: **All-checkpoint SASNE embedding of the complete Representation Graph Family.** Each point represents one graph corresponding to the representation of the CIFAR-10 test set extracted from a particular checkpoint and training epoch. Each checkpoint is assigned a distinct color: the input representation is shown in gray and is shaped as a square, while Checkpoints 1–4 and the output representation are colored blue, red, green, purple, and yellow, respectively. Within each checkpoint, progressively lighter shades indicate increasing training epochs, while consecutive representations are connected by line segments to emphasize the temporal trajectory followed throughout optimization. Black markers denote the earliest trainable representation and gray markers denote the final epoch for every checkpoint. Since the embedding is computed from the complete SAGD distance matrix, all trajectories are represented within a common geometric space, enabling direct comparison across both layers and training stages.

Following this initial transition, the trajectories become considerably denser and successive representations move through the manifold by progressively smaller amounts. This behavior parallels the long refinement phase observed in the learning curves, during which both accuracy and loss continue to improve, albeit at a much slower rate. From the representational perspective, learning appears to proceed through smooth geometric refinements.

An equally interesting observation emerges near the end of training. Although the learning curves indicate that predictive performance has essentially converged after approximately epoch 170, specific representation trajectories—namely those of Checkpoint 2, Checkpoint 4, and the output representation—continue evolving toward their final gray endpoints. Consequently, while conventional optimization metrics suggest that learning has largely saturated, the Representation Graph Family reveals that the internal geometry of these particular layers continues to reorganize even after predictive performance has stabilized. This observation illustrates one of the principal advantages of the proposed graph-based analysis: whereas loss and accuracy quantify the global evolution of the classifier, SAGD directly characterizes the localized evolution of the learned representations themselves.

The all-checkpoint embedding reveals that not all checkpoints evolve in the same manner. The input representation and the first three checkpoints collectively form a continuous, elongated, and curved macro-structure—resembling an upward-facing arc. Within this shared formation, Checkpoint 1 expands toward the right extremity, Checkpoint 3 develops toward the left, and Checkpoint 2 bridges the center. In contrast, Checkpoint 4 and the output representation are situated above this arc and follow a distinctly different geometric pattern. Rather than forming extended trajectories, they manifest as compact, localized clouds that exhibit pronounced temporal segregation, with early training epochs clustered on one side of the formation and late epochs grouped on the opposite side. While these two broad structural regimes are clearly distinguishable in the global view, viewing the complete 3D embedding as a static projection obscures the finer epoch-by-epoch dynamics inside each individual checkpoint. Therefore, to better visualize these internal evolutionary details, the following sections examine the shallow and deep layers separately while preserving the common geometric reference frame established by the all-checkpoint SASNE embedding.

Shallow-Layer Dynamics

The all-checkpoint embedding shown in Figure 4.8 suggests that the input representation together with Checkpoints 1–3 exhibit a similar large-scale organization. To examine this behavior in greater detail, we now focus exclusively on these shallow checkpoints.

Importantly, the following visualizations are not obtained by recomputing SASNE

from a SAGD matrix restricted to the selected checkpoints. Instead, the complete Representation Graph Family is first used to compute the BHD matrix, thereby encoding the global manifold structure induced by all 251 representations. Only after this step are the checkpoints of interest selected, and the final SASNE embedding is computed from the corresponding submatrix of the BHD matrix. Consequently, although only a subset of representations is embedded, their pairwise distances already incorporate information from the complete manifold, allowing the resulting visualization to preserve the global geometric context while providing a clearer view of the selected trajectories.

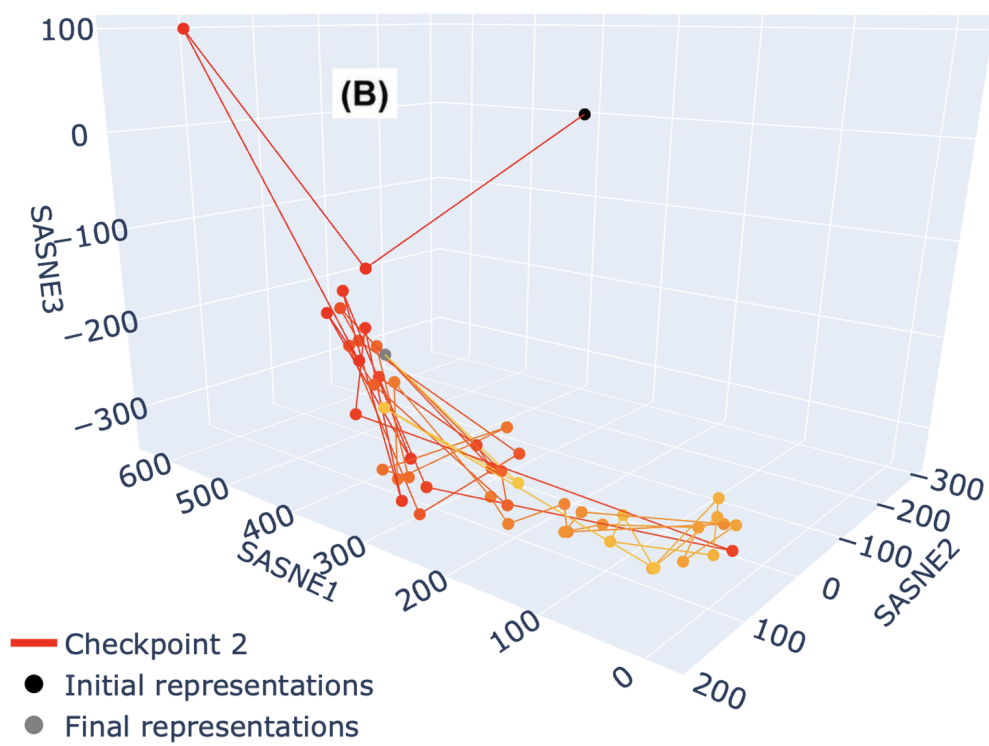
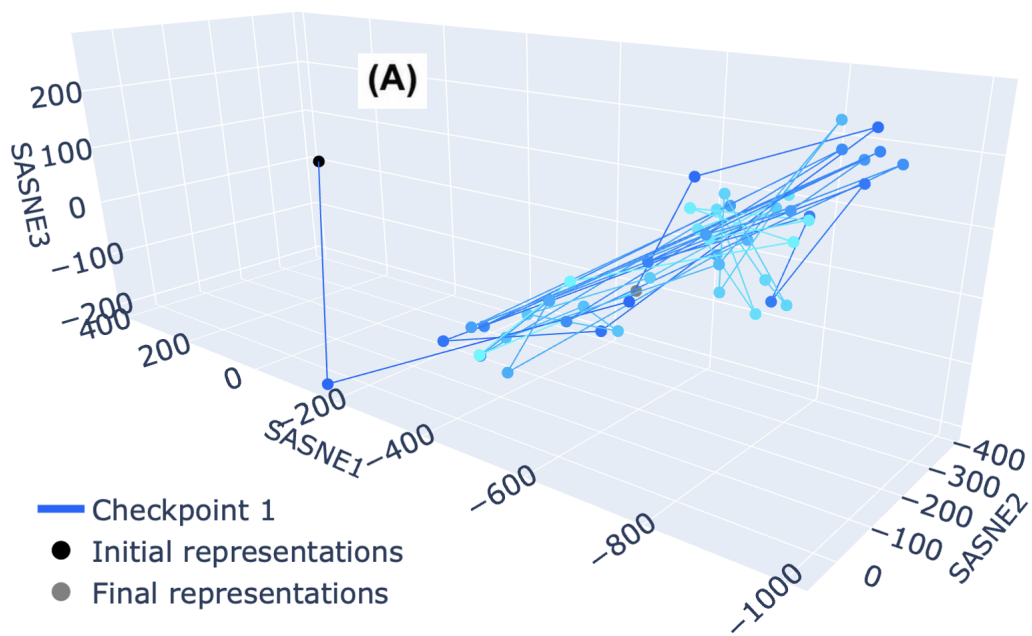
Figure 4.9 shows the trajectories corresponding to Checkpoints 1–3 individually, allowing the evolution of each layer to be examined without the visual complexity introduced by the remaining checkpoints.

Several common characteristics emerge immediately from Figure 4.9. The most prominent is the exceptionally large geometric displacement occurring at the beginning of optimization. For all three checkpoints, the transition from the initial trainable representation to the first few training epochs is substantially larger than any subsequent movement observed throughout training. This pronounced initial transition closely coincides with the rapid increase in classification accuracy and the steep decrease in loss observed in Figure 4.7, indicating that the period of fastest optimization is also accompanied by the largest structural reorganization of the representation space.

Beyond this initial reorganization, however, the trajectories reveal a geometric property that is not immediately apparent from the all-checkpoint embedding. Rather than continuing to explore the embedding freely, the successive representations remain confined within a narrow, curved region of the SASNE space. After entering this region during the first epochs of training, optimization proceeds almost exclusively inside it, with each new representation corresponding to a local displacement along the same curved structure. This behavior suggests that the early layers rapidly establish a stable representational regime, after which learning consists primarily of refining the geometry within an already well-defined portion of the manifold rather than repeatedly reorganizing it.

The temporal organization of the trajectories further reinforces this interpretation. Instead of accumulating around a single location after the initial transition, the representations are distributed along an elongated curved path, with early, intermediate, and late epochs occupying progressively different regions of this structure. In this sense, the representation space appears to evolve through directed geometric refinement rather than through successive independent adjustments.

Another noteworthy characteristic is the absence of abrupt reorganizations after the initial transition. Once the trajectories enter their respective representational regimes, the magnitude of successive displacements decreases progressively and no evidence of additional large-scale geometric rearrangements is observed. Instead, optimization proceeds through continuous adaptation, indicating that the evolution of the early representations is smooth



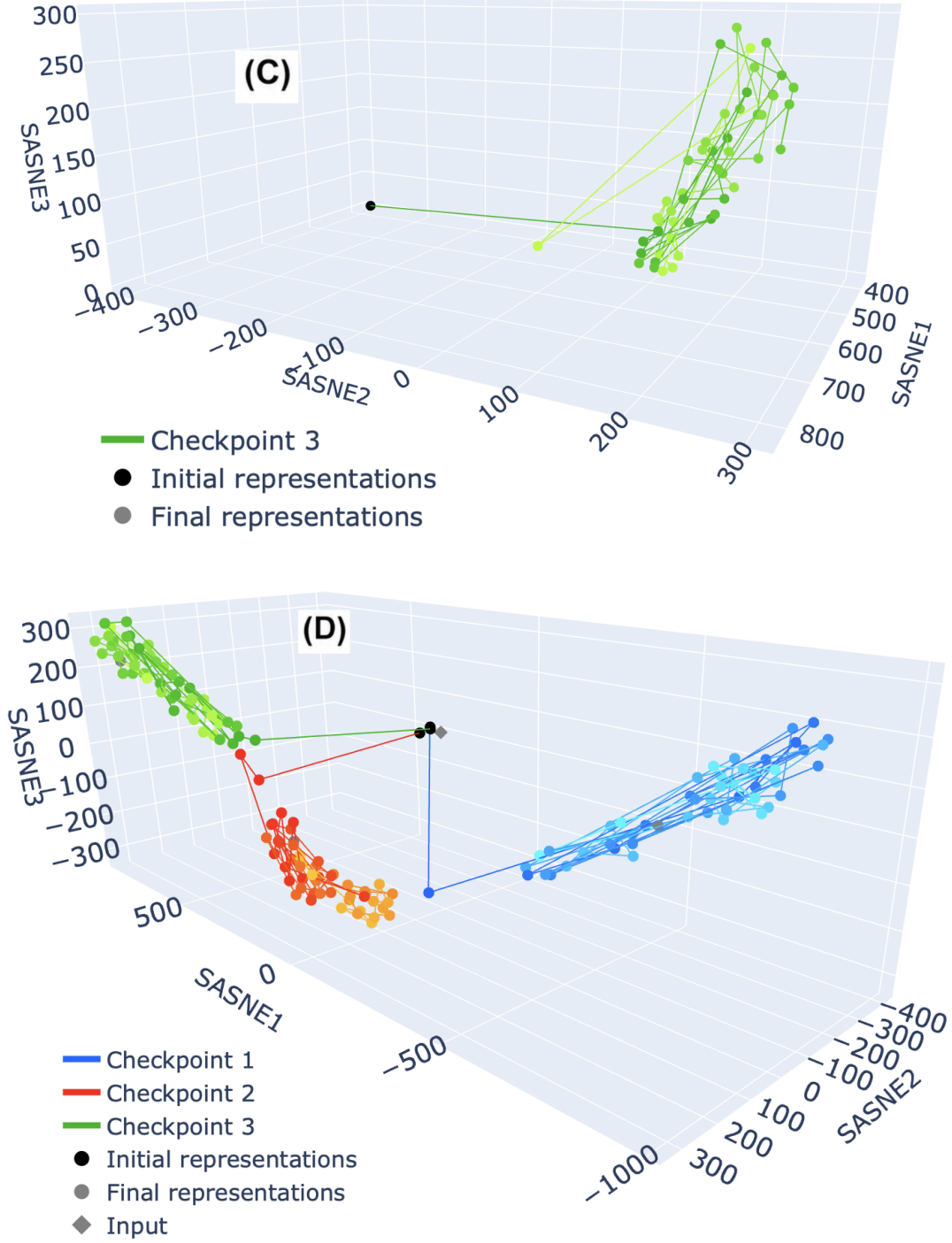


Figure 4.9: **Evolution of the early checkpoints within the global SASNE embedding.** Panels (A)–(C) display the trajectories of Checkpoints 1, 2, and 3, respectively, extracted from the common SASNE embedding shown in Figure 4.8. Colors indicate training progression, with darker shades corresponding to earlier epochs and lighter shades to later epochs. Black markers denote the earliest trainable representation, gray markers indicate the final epoch, and consecutive representations are connected by line segments to illustrate the temporal evolution of each checkpoint. Since these trajectories are selected from the all-checkpoint embedding rather than recomputed independently, their geometry remains directly comparable with both the complete manifold and the analyses presented in subsequent sections.

rather than phase-like.

The trajectories continue advancing toward their final gray endpoints well after the learning curves have largely stabilized. Although the remaining displacements become considerably smaller than those observed during the earliest epochs, they persist throughout the remainder of training. Thus, while conventional optimization metrics suggest that learning has essentially converged, the graph-based analysis reveals that the internal geometry of the early representations continues to evolve through subtle but consistent refinements. This observation further illustrates the complementary information provided by SAGD, which captures ongoing changes in the organization of the learned representation space even after predictive performance exhibits only marginal improvements.

Deep-Layer Dynamics

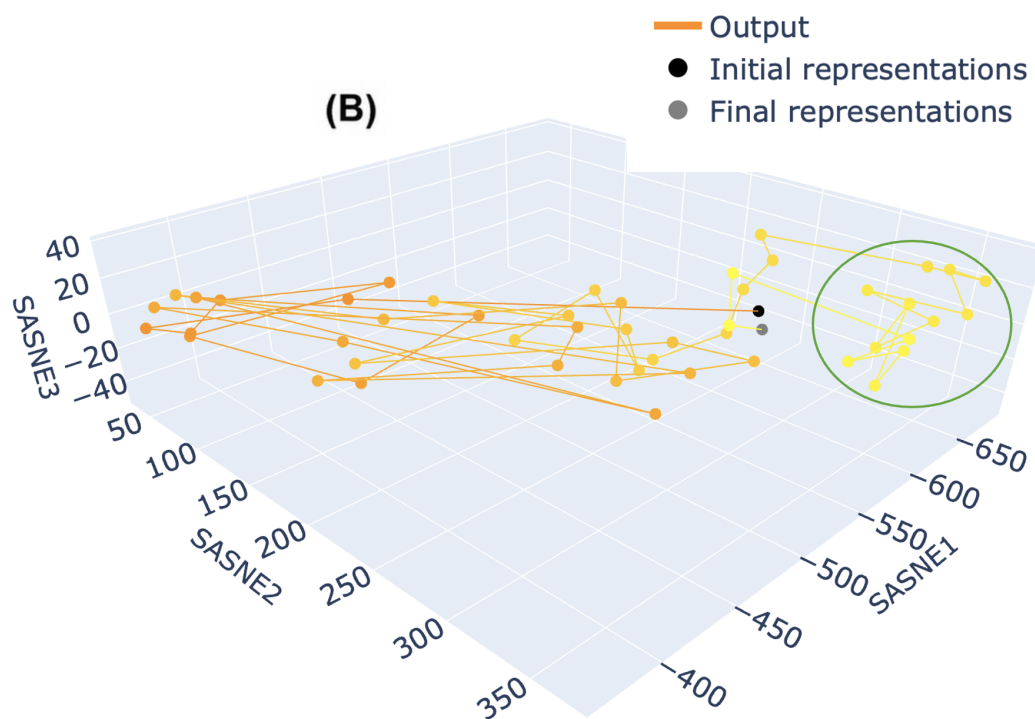
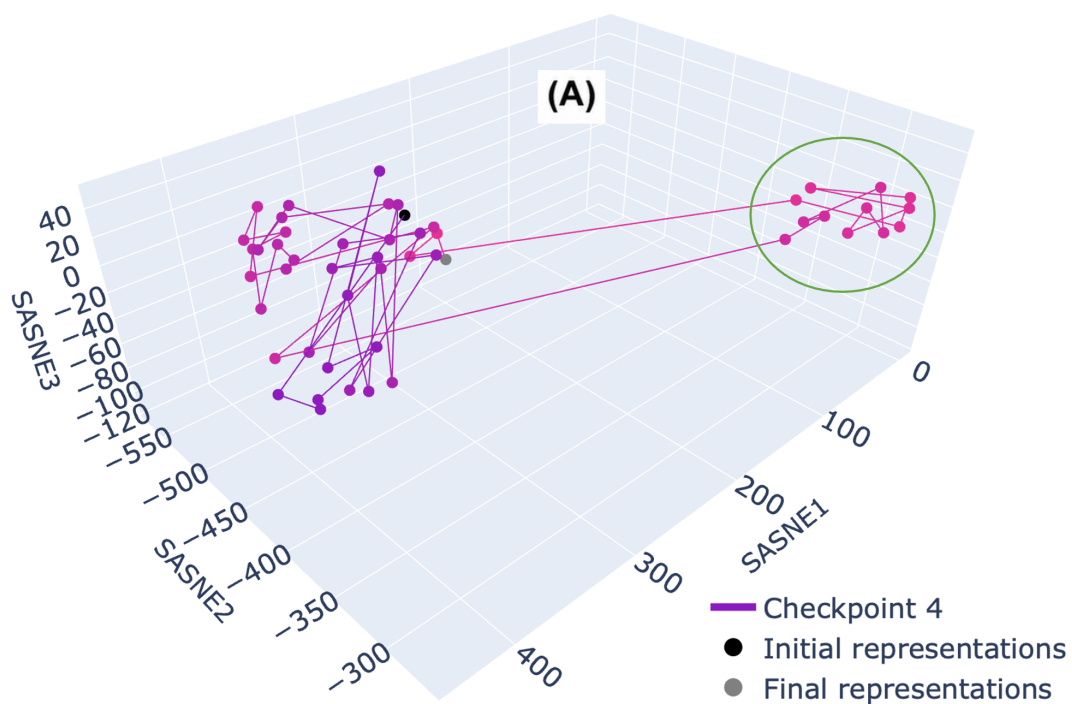
The all-checkpoint embedding also suggests that Checkpoint 4 and the output representation behave differently from the preceding layers while exhibiting sufficient similarity to one another. To better understand this common behavior, we now examine these two checkpoints separately from the shallow stages of the network.

As in the previous subsection, the visualizations are obtained by selecting the corresponding representations from the BHD matrix computed using the complete Representation Graph Family. Consequently, although only two checkpoints are displayed, their geometric relationships remain conditioned by the complete manifold and therefore preserve the common geometric reference frame established by the all-checkpoint SASNE embedding.

Figure 4.10 presents the trajectories of Checkpoint 4 and the output representation individually, together with a joint visualization that facilitates their direct comparison.

Several characteristics distinguish the deep checkpoints from the representational behavior observed in the first three residual stages. The most evident difference is the geometry of the trajectories themselves. Rather than extending along elongated curved regions of the embedding, both Checkpoint 4 and the output representation remain confined to relatively compact regions of the manifold. Although optimization continues throughout training, the representations explore only a limited neighborhood of the SASNE space. This indicates that—even during the rapid optimization phase of the first 10 to 15 epochs—the deepest layers undergo substantially less large-scale geometric displacement than the shallower checkpoints.

This compact organization suggests a qualitative change in the nature of the optimization process. Whereas the shallower checkpoints continuously traverse an extended representational path, the deepest layers appear to perform predominantly local geometric refinements within an already established representational structure. In other words, the optimization no longer consists of broad exploration of the manifold but of progressively



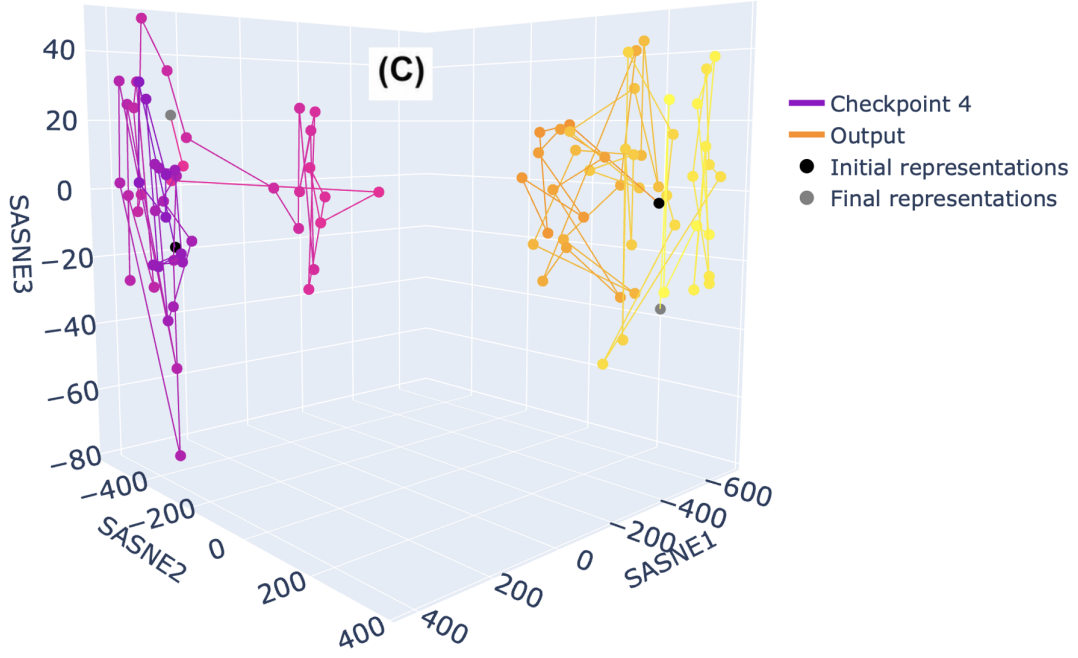


Figure 4.10: **Evolution of the deep checkpoints within the global SASNE embedding.** Panel (A) shows the trajectory of Checkpoint 4, panel (B) the trajectory of the output representation, and panel (C) both checkpoints embedded together within the common SASNE space. Colors encode training progression from darker to lighter shades, black markers denote the earliest trainable representation, gray markers indicate the final epoch, and consecutive representations are connected by line segments to illustrate the temporal evolution. Since these embeddings are obtained from the BHD matrix computed over the complete Representation Graph Family, the trajectories preserve the global geometric context while allowing a detailed comparison of the late-stage representational dynamics. In panel (A) and (B) the green circle denotes epochs 175 to 230. Epochs associated with the convergence stage of the classifier loss and accuracy.

adjusting representations inside a stable geometric neighborhood.

Despite this compactness, the trajectories are far from temporally homogeneous. In both Checkpoint 4 and the output representation, the embedding separates into two neighboring temporal regions. The darker representations corresponding to the earliest stages of optimization occupy one side of the manifold, whereas progressively lighter representations migrate toward a second region containing the final stages of training. This temporal organization is consistently observed in both checkpoints and becomes particularly evident in the joint visualization shown in Figure 4.10(C), indicating that the deepest representations evolve according to a common geometric pattern.

Remarkably, this partition closely reflects the optimization dynamics observed in the learning curves. The transition toward the second region occurs approximately during the stage at which the classification accuracy and loss begin to stabilize (around epoch 175), suggesting that the onset of convergence; for the learning curve, is accompanied by a reorganization of the deepest representation space. Furthermore, the final representations corresponding to the last training epochs exhibit a subtle drift back toward the region occupied by the earlier representations. This behavior coincides with the slight increase in test loss and the modest decrease in test accuracy observed during the final stage of training; observed in Figure 4.7, where mild overfitting becomes apparent. Although the geometric changes remain continuous and no abrupt transition is observed, the existence of these two temporally organized regions suggests that the deepest representations capture distinct phases of the optimization process that are not evident from the learning curves alone. From the perspective of SAGD, the deep layers therefore appear to encode not only the progression of learning but also the onset of the final optimization regime.

Another notable observation is the similarity between Checkpoint 4 and the output representation. Although they correspond to different stages of the network, both exhibit alike geometric characteristics: a pronounced initial displacement, compact trajectories and similar temporal partition between early and late epochs. Their parallel behavior strongly suggests that these two checkpoints belong to a common representational regime, distinct from that characterizing the shallower layers of the network.

Finally, the global organization of the Representation Graph Family provides little evidence of representational redundancy across the architecture. Throughout training, each checkpoint occupies a distinct region of the representation manifold and follows its own continuous trajectory, with only limited overlap between neighboring checkpoints. Although the deepest checkpoints exhibit similar optimization dynamics, they nevertheless remain geometrically distinguishable within the embedding. Likewise, the shallower checkpoints evolve along neighboring but separate manifolds, indicating that successive stages of the network learn complementary rather than duplicated representations. From the perspective of SAGD, the residual architecture therefore appears to preserve a progressive hierarchy of representations instead of repeatedly encoding the same geometric structure

across multiple layers. This observation is consistent with the role of residual networks as hierarchical feature extractors, where each stage contributes additional representational structure while building upon the information learned by the preceding layers.

Chapter 5

Conclusion

This thesis investigated a graph-based framework for comparing neural network representations by modeling them as weighted graphs and analyzing their intrinsic geometry through spectral graph theory. The central contribution is the investigation of the empirical properties of the Shape-Aware Graph Distance (SAGD), a graph-to-graph dissimilarity measure that combines the Commute Time Distance (CTD) with the 1-Wasserstein distance between the empirical distributions of pairwise graph distances. Rather than comparing representations through their coordinates in a latent Euclidean space, SAGD compares the global structural organization induced by the relationships among data points.

One of the principal advantages of the studied methodology is that it naturally addresses several limitations of existing representation similarity measures [2]. Since SAGD is based on graph topology rather than individual feature coordinates, it captures the intrinsic geometry of representations instead of relying exclusively on Euclidean relationships in high-dimensional spaces. By incorporating the CTD, the proposed framework accounts for the global connectivity of the representation graph, allowing information from multiple paths through the graph to contribute to the comparison rather than only local pairwise distances. Furthermore, summarizing the collection of CTDs through their empirical cumulative distribution functions produces a statistical characterization of the representation that is naturally invariant to permutations of the observations. The use of the 1-Wasserstein distance then provides a principled and interpretable measure of structural discrepancy between representations.

From a theoretical perspective, the thesis established the mathematical framework underlying the proposed methodology. The spectral properties of the graph Laplacian provided the foundation for constructing the CTD embedding, while the connection between spectral graph theory and the 1-Wasserstein distance motivated the definition of SAGD. The resulting distance inherits desirable properties from the Wasserstein metric, including symmetry, non-negativity, and the triangle inequality, while the normalization procedure further improves robustness by removing scaling effects of the graphs volume.

Together, these results provide a mathematical framework for studying neural network representations from the perspective of graph geometry.

The empirical experiments demonstrated that the proposed methodology is sufficiently robust for practical applications. The sensitivity analyses showed that the automatic graph construction procedure remains stable under moderate perturbations of both the neighborhood size and the kernel bandwidth. The synthetic Gaussian experiments provided a statistical calibration for interpreting SAGD values and showed that the structural variation induced by these perturbations is generally comparable to the variability naturally observed between independent samples from the same probability distribution. The experiments based on Watts–Strogatz small-world networks further demonstrated that SAGD successfully identifies the underlying structural regime of a graph despite the stochastic variability inherent in random graph generation. Collectively, these experiments indicate that the proposed framework is not sensitive to the graph construction procedure and therefore provides a reliable basis for representation analysis.

The primary contribution of this thesis, however, lies in the analysis of neural network representation dynamics. By representing every checkpoint of a ResNet-18 throughout training as an element of a Representation Graph Family, SAGD provided a geometric description of how the internal organization of the network evolves during optimization. Unlike conventional learning curves, which describe only predictive performance, the proposed framework allowed the evolution of the representations themselves to be studied directly.

The results revealed that representational learning is highly structured throughout optimization. The largest geometric changes occurred during the earliest epochs of training, coinciding with the period of fastest improvement in classification accuracy and loss. This indicates that the initial optimization stage is characterized by a substantial reorganization of the representation space, during which the network rapidly establishes meaningful geometric relationships among the data. After this initial phase, the evolution of the representations became considerably smoother, with successive checkpoints undergoing progressively smaller structural changes. Rather than repeatedly reorganizing the representation space, the network entered a regime of gradual geometric refinement.

An important observation is that representational evolution continued even after the learning curves stabilize. Although the classifier reached near-optimal predictive performance relatively early during optimization, the Representation Graph Family continued to evolve throughout the remaining training epochs. This suggests that optimization does not terminate once classification performance saturates. Instead, the internal geometry of the learned feature spaces continues to be refined through subtle structural modifications that are largely invisible when monitoring only loss or accuracy. In this sense, SAGD provides complementary information to standard performance evaluation methods—specifically the monitoring of cross-entropy loss and classification accuracy—by revealing aspects of

learning that are not directly reflected in predictive performance.

The analysis also demonstrated that different checkpoints of the network exhibit distinct representational dynamics. The shallow residual blocks followed extended trajectories through the representation manifold, indicating continuous geometric refinement over a relatively broad region of the embedding space. Their evolution was characterized by a large initial reorganization followed by a smooth progression along elongated manifold structures, suggesting that these layers continuously refine their representations throughout optimization. In contrast, the deepest layers evolved within much more localized regions of the manifold while maintaining a clear temporal progression throughout training. Rather than traversing extended trajectories, these representations remained confined to compact neighborhoods and exhibited a consistent partition into two temporally organized regions associated with the early and late stages of optimization. This geometric organization closely paralleled the convergence behavior observed in the learning curves and even reflected the onset of the mild overfitting regime during the final training epochs. Consequently, the deepest representations appear to encode distinct optimization phases while remaining within an already established representational structure. Together, these observations suggest that the nature of learning changes with network depth, with shallow layers undergoing broad geometric refinement and deeper layers progressively reorganizing increasingly structured feature spaces through localized adjustments.

Several limitations remain. Since SAGD compares empirical distributions of CTDs, distinct graphs may produce identical distributions, implying that the method defines a pseudo-metric rather than a true metric on the space of graphs. Moreover, the present work focused on representations extracted from a single architecture trained on a single image classification dataset. Although this provided a suitable framework for evaluating the proposed methodology, additional experiments involving different architectures, learning paradigms, and datasets would provide a broader assessment of its applicability.

Future work may proceed in several directions. From a theoretical perspective, it would be valuable to investigate the statistical properties of SAGD, including upper-lower bounds, convergence, and asymptotic behavior under random graph models. Alternative graph constructions and other spectral distance measures could also be explored within the proposed framework. From an applied perspective, SAGD could be employed to compare representations across different neural network architectures, optimization algorithms, initialization strategies, or transfer learning settings. Since the methodology depends only on graph representations, it also has potential applications beyond deep learning wherever complex data can be naturally represented as weighted graphs.

In summary, this thesis demonstrates that studying neural network representations from a graph-theoretic perspective provides meaningful insights into the learning process that complement traditional optimization metrics. By combining spectral graph theory, commute time distances, 1-Wasserstein distances, and shape-aware visualization, SAGD

offers a mathematically grounded and practically robust framework for comparing representations and analyzing their evolution throughout training. The results suggest that understanding how neural networks learn requires not only evaluating their predictive performance, but also examining the geometric structure that emerges within their internal representations as optimization progresses.

Appendix A

Proofs of Auxiliary Results

Theorem A.1 (Properties of L). *The matrix L satisfies the following properties:*

1. L is symmetric and positive semi-definite.
2. The smallest eigenvalue of L is 0, and the corresponding eigenvector is the constant one vector $\mathbb{1}$.
3. L has n non-negative, real-valued eigenvalues

$$0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n. \quad (\text{A.1})$$

Proof. We prove each property separately.

1. L is symmetric and positive semi-definite.

Since the graph is undirected, the weight matrix W is symmetric. The degree matrix D is diagonal, hence symmetric as well. Therefore,

$$L^\top = (D - W)^\top = D^\top - W^\top = D - W = L, \quad (\text{A.2})$$

which proves that L is symmetric.

To show positive semi-definiteness, let $x \in \mathbb{R}^n$. Then

$$\begin{aligned} x^\top Lx &= x^\top (D - W)x \\ &= \sum_{i=1}^n d_i x_i^2 - \sum_{i,j=1}^n w_{ij} x_i x_j. \end{aligned} \quad (\text{A.3})$$

Using the identity $d_i = \sum_{j=1}^n w_{ij}$, we obtain

$$x^\top Lx = \frac{1}{2} \sum_{i,j=1}^n w_{ij} (x_i - x_j)^2. \quad (\text{A.4})$$

Since $w_{ij} \geq 0$ and $(x_i - x_j)^2 \geq 0$, it follows that

$$x^\top Lx \geq 0. \quad (\text{A.5})$$

Therefore, L is positive semi-definite.

2. *The smallest eigenvalue of L is 0, and the corresponding eigenvector is $\mathbb{1}$.*

Let $\mathbb{1} = (1, \dots, 1)^\top$. Then

$$\begin{aligned} L\mathbb{1} &= (D - W)\mathbb{1} \\ &= D\mathbb{1} - W\mathbb{1}. \end{aligned} \quad (\text{A.6})$$

The i -th component of $D\mathbb{1}$ is d_i , while

$$(W\mathbb{1})_i = \sum_{j=1}^n w_{ij} = d_i. \quad (\text{A.7})$$

Hence,

$$L\mathbb{1} = 0. \quad (\text{A.8})$$

Therefore, $\mathbb{1}$ is an eigenvector associated with eigenvalue 0. Since L is positive semi-definite, all eigenvalues are non-negative, implying that 0 is the smallest eigenvalue.

3. *L has n non-negative, real-valued eigenvalues.*

Since L is symmetric, the spectral theorem implies that all eigenvalues of L are real-valued.

Let v be an eigenvector associated with eigenvalue λ . Then

$$Lv = \lambda v. \quad (\text{A.9})$$

Multiplying on the left by $v^\top$ gives

$$v^\top Lv = \lambda v^\top v. \quad (\text{A.10})$$

Because L is positive semi-definite,

$$v^\top Lv \geq 0, \quad (\text{A.11})$$

and since $v^\top v > 0$ for any non-zero eigenvector, it follows that

$$\lambda \geq 0. \quad (\text{A.12})$$

Thus, all eigenvalues are real and non-negative. Ordering them increasingly gives

$$0 = \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n. \quad (\text{A.13})$$

□

Theorem A.2 (Embedding equivalence). *Let $G = (V, E, W)$ be a connected, undirected graph with degree matrix*

$$D = \text{diag}(d_1, \dots, d_n). \quad (\text{A.14})$$

Let $c(i, j)$ denote the commute time distance between nodes v_i and v_j , and let $L_{\text{sym}}^\dagger$ denote the Moore–Penrose pseudoinverse of the normalized Laplacian

$$L_{\text{sym}} = D^{-1/2} L D^{-1/2}. \quad (\text{A.15})$$

Then

$$c(i, j) = \text{vol}(G) (e_i - e_j)^\top D^{-1/2} L_{\text{sym}}^\dagger D^{-1/2} (e_i - e_j) = \|\mathbf{z}_i - \mathbf{z}_j\|_2^2. \quad (\text{A.16})$$

Proof. Let

$$L_{\text{sym}} = \mathcal{U} \Lambda^{\text{sym}} \mathcal{U}^\top \quad (\text{A.17})$$

be the eigendecomposition of the normalized Laplacian. Define

$$\mathcal{U}' = (\mathbf{u}_2, \dots, \mathbf{u}_n), \quad (\text{A.18})$$

and let

$$\Lambda^{\text{sym}'} = \text{diag}(\lambda_2, \dots, \lambda_n). \quad (\text{A.19})$$

The normalized embedding is given by

$$\Theta^{\text{sym}} = \sqrt{\text{vol}(G)} (\Lambda^{\text{sym}'})^{-1/2} (\mathcal{U}')^\top D^{-1/2}, \quad (\text{A.20})$$

with

$$\Theta^{\text{sym}} = (\mathbf{z}_1, \dots, \mathbf{z}_n). \quad (\text{A.21})$$

Hence, for two nodes v_i and v_j ,

$$\begin{aligned} \langle \mathbf{z}_i, \mathbf{z}_j \rangle &= (\Theta_i^{\text{sym}})^\top \Theta_j^{\text{sym}} \\ &= \text{vol}(G) e_i^\top D^{-1/2} \mathcal{U}' (\Lambda^{\text{sym}'})^{-1} (\mathcal{U}')^\top D^{-1/2} e_j. \end{aligned} \quad (\text{A.22})$$

Since

$$L_{\text{sym}}^\dagger = \mathcal{U}' (\Lambda^{\text{sym}'})^{-1} (\mathcal{U}')^\top, \quad (\text{A.23})$$

it follows that

$$\langle \mathbf{z}_i, \mathbf{z}_j \rangle = \text{vol}(G) e_i^\top D^{-1/2} L_{\text{sym}}^\dagger D^{-1/2} e_j. \quad (\text{A.24})$$

Therefore,

$$\begin{aligned} \|\mathbf{z}_i - \mathbf{z}_j\|_2^2 &= \langle \mathbf{z}_i, \mathbf{z}_i \rangle - 2\langle \mathbf{z}_i, \mathbf{z}_j \rangle + \langle \mathbf{z}_j, \mathbf{z}_j \rangle \\ &= \text{vol}(G) (e_i - e_j)^\top D^{-1/2} L_{\text{sym}}^\dagger D^{-1/2} (e_i - e_j). \end{aligned} \quad (\text{A.25})$$

To relate this expression to the unnormalized Laplacian L , consider the matrix $M = D^{-1/2} L_{\text{sym}}^\dagger D^{-1/2}$. Although the Moore-Penrose pseudoinverse does not distribute over matrix multiplication (meaning $M \neq L^\dagger$), M is nevertheless a symmetric generalized inverse of L . To verify this, we confirm the property $LM L = L$:

$$\begin{aligned} L M L &= L (D^{-1/2} L_{\text{sym}}^\dagger D^{-1/2}) L \\ &= (D^{1/2} L_{\text{sym}} D^{1/2}) (D^{-1/2} L_{\text{sym}}^\dagger D^{-1/2}) (D^{1/2} L_{\text{sym}} D^{1/2}) \\ &= D^{1/2} L_{\text{sym}} L_{\text{sym}}^\dagger L_{\text{sym}} D^{1/2} \\ &= D^{1/2} L_{\text{sym}} D^{1/2} \\ &= L. \end{aligned} \quad (\text{A.26})$$

A fundamental property of generalized inverses is that the quadratic form $u^\top M u$ is invariant for all vectors u in the column space of L . Since L is the Laplacian of a connected graph, its column space consists of all vectors orthogonal to $\mathbb{1}$. Letting $u = e_i - e_j$, we observe that $u^\top \mathbb{1} = 1 - 1 = 0$, meaning $(e_i - e_j) \in \text{Range}(L)$.

Consequently, evaluating the quadratic form using M yields the identical result as using the true Moore-Penrose pseudoinverse $L^\dagger$:

$$(e_i - e_j)^\top (D^{-1/2} L_{\text{sym}}^\dagger D^{-1/2}) (e_i - e_j) = (e_i - e_j)^\top L^\dagger (e_i - e_j). \quad (\text{A.27})$$

Substituting this into our previous expression, we obtain:

$$\begin{aligned} \|\mathbf{z}_i - \mathbf{z}_j\|_2^2 &= \text{vol}(G) (e_i - e_j)^\top L^\dagger (e_i - e_j) \\ &= c(i, j), \end{aligned} \quad (\text{A.28})$$

where the last equality follows directly from Theorem 3.2. $\square$

Theorem A.3 (Spectral representation of the Biharmonic Distance). *Let $G = (V, E, W)$ be a connected, undirected graph with Laplacian matrix L . Let*

$$L = \mathcal{V} \Lambda \mathcal{V}^\top \quad (\text{A.29})$$

be the eigendecomposition of L , where

$$\Lambda = \text{diag}(0, \lambda_2, \dots, \lambda_n), \quad (\text{A.30})$$

and let $\mathbf{v}_\alpha$ denote the eigenvector associated with the eigenvalue λ_α . Then the Biharmonic Distance between nodes v_i and v_j can be written as

$$b(i, j) = \sum_{\alpha=2}^n \frac{(\mathbf{v}_{\alpha i} - \mathbf{v}_{\alpha j})^2}{\lambda_\alpha^2}. \quad (\text{A.31})$$

Proof. Since L is symmetric and positive semi-definite, its Moore–Penrose pseudoinverse admits the spectral decomposition

$$L^\dagger = \sum_{\alpha=2}^n \frac{1}{\lambda_\alpha} \mathbf{v}_\alpha \mathbf{v}_\alpha^\top. \quad (\text{A.32})$$

Squaring the pseudoinverse yields

$$(L^\dagger)^2 = \sum_{\alpha=2}^n \frac{1}{\lambda_\alpha^2} \mathbf{v}_\alpha \mathbf{v}_\alpha^\top, \quad (\text{A.33})$$

where orthonormality of the eigenvectors has been used.

Therefore, the (i, j) -entry of $(L^\dagger)^2$ is

$$(L^\dagger)^2_{ij} = \sum_{\alpha=2}^n \frac{\mathbf{v}_{\alpha i} \mathbf{v}_{\alpha j}}{\lambda_\alpha^2}. \quad (\text{A.34})$$

Using the definition of the Biharmonic Distance,

$$b(i, j) = (L^\dagger)^2_{ii} - 2(L^\dagger)^2_{ij} + (L^\dagger)^2_{jj}, \quad (\text{A.35})$$

we obtain

$$\begin{aligned} b(i, j) &= \sum_{\alpha=2}^n \frac{\mathbf{v}_{\alpha i}^2}{\lambda_\alpha^2} - 2 \sum_{\alpha=2}^n \frac{\mathbf{v}_{\alpha i} \mathbf{v}_{\alpha j}}{\lambda_\alpha^2} + \sum_{\alpha=2}^n \frac{\mathbf{v}_{\alpha j}^2}{\lambda_\alpha^2} \\ &= \sum_{\alpha=2}^n \frac{\mathbf{v}_{\alpha i}^2 - 2\mathbf{v}_{\alpha i} \mathbf{v}_{\alpha j} + \mathbf{v}_{\alpha j}^2}{\lambda_\alpha^2}. \end{aligned} \quad (\text{A.36})$$

Finally, using the identity

$$a^2 - 2ab + b^2 = (a - b)^2, \quad (\text{A.37})$$

it follows that

$$b(i, j) = \sum_{\alpha=2}^n \frac{(\mathbf{v}_{\alpha i} - \mathbf{v}_{\alpha j})^2}{\lambda_\alpha^2}. \quad (\text{A.38})$$

□

Theorem A.4 (Spectral representation of the Commute Time Distance). *Let $G = (V, E, W)$ be a connected, undirected graph with Laplacian matrix L . Let*

$$L = \mathcal{V}\Lambda\mathcal{V}^\top \quad (\text{A.39})$$

be the eigendecomposition of L , where

$$\Lambda = \text{diag}(0, \lambda_2, \dots, \lambda_n), \quad (\text{A.40})$$

and let $\mathbf{v}_\alpha$ denote the eigenvector associated with the eigenvalue λ_α . Then the Commute Time Distance between nodes v_i and v_j satisfies

$$c(i, j) = \text{vol}(G) \sum_{\alpha=2}^n \frac{(\mathbf{v}_{\alpha i} - \mathbf{v}_{\alpha j})^2}{\lambda_\alpha}. \quad (\text{A.41})$$

Proof. From Theorem 3.2, the CTD is given by

$$c(i, j) = \text{vol}(G) \left(l_{ii}^\dagger - 2l_{ij}^\dagger + l_{jj}^\dagger \right). \quad (\text{A.42})$$

Using the spectral decomposition of the pseudoinverse,

$$L^\dagger = \sum_{\alpha=2}^n \frac{1}{\lambda_\alpha} \mathbf{v}_\alpha \mathbf{v}_\alpha^\top, \quad (\text{A.43})$$

the entries of $L^\dagger$ satisfy

$$l_{ij}^\dagger = \sum_{\alpha=2}^n \frac{\mathbf{v}_{\alpha i} \mathbf{v}_{\alpha j}}{\lambda_\alpha}. \quad (\text{A.44})$$

Substituting into the expression for $c(i, j)$ gives

$$\begin{aligned} c(i, j) &= \text{vol}(G) \left(\sum_{\alpha=2}^n \frac{\mathbf{v}_{\alpha i}^2}{\lambda_\alpha} - 2 \sum_{\alpha=2}^n \frac{\mathbf{v}_{\alpha i} \mathbf{v}_{\alpha j}}{\lambda_\alpha} + \sum_{\alpha=2}^n \frac{\mathbf{v}_{\alpha j}^2}{\lambda_\alpha} \right) \\ &= \text{vol}(G) \sum_{\alpha=2}^n \frac{\mathbf{v}_{\alpha i}^2 - 2\mathbf{v}_{\alpha i} \mathbf{v}_{\alpha j} + \mathbf{v}_{\alpha j}^2}{\lambda_\alpha}. \end{aligned} \quad (\text{A.45})$$

Using the identity

$$a^2 - 2ab + b^2 = (a - b)^2, \quad (\text{A.46})$$

we conclude that

$$c(i, j) = \text{vol}(G) \sum_{\alpha=2}^n \frac{(\mathbf{v}_{\alpha i} - \mathbf{v}_{\alpha j})^2}{\lambda_\alpha}. \quad (\text{A.47})$$

□

Appendix B

Additional Figures

This appendix presents additional visualizations of the SASNE embeddings analyzed in Chapter 4. Each experiment displays the same embedding without temporal connections between consecutive epochs. Removing the connecting lines facilitates inspection of the spatial organization of the representations. Unless otherwise stated, all embeddings preserve the same color encoding adopted throughout the chapter, where color identifies the network checkpoint and progressively lighter shades indicate increasing training epochs.

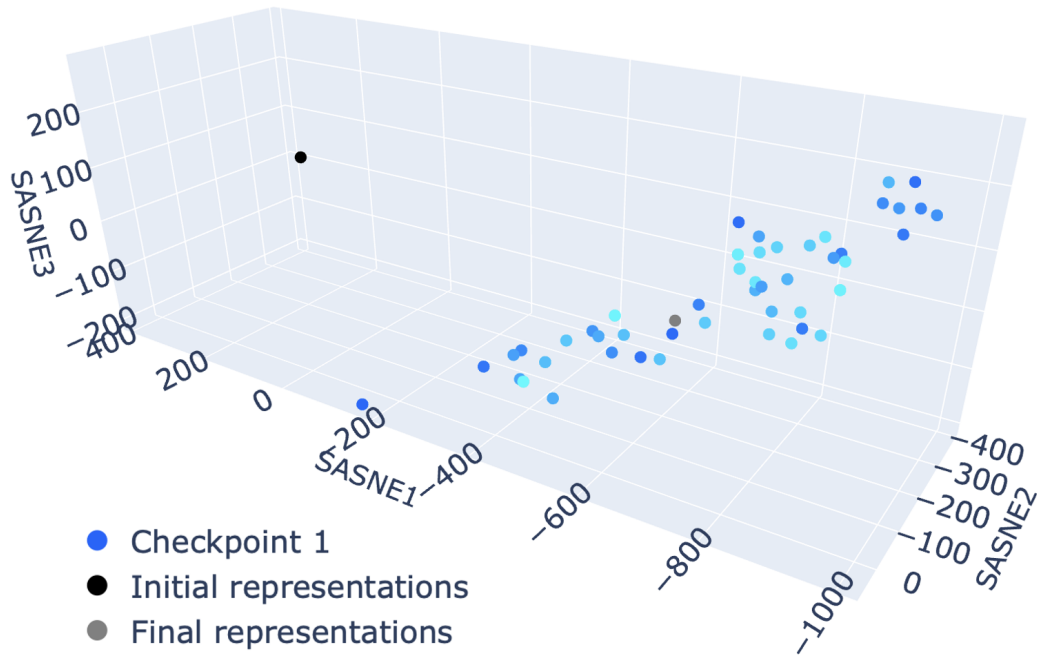


Figure B.1: **SASNE embedding of Checkpoint 1 without temporal trajectories.** Identical embedding after removing the temporal connections between consecutive epochs. The point-only visualization highlights the elongated shape organization followed by the representations throughout optimization.

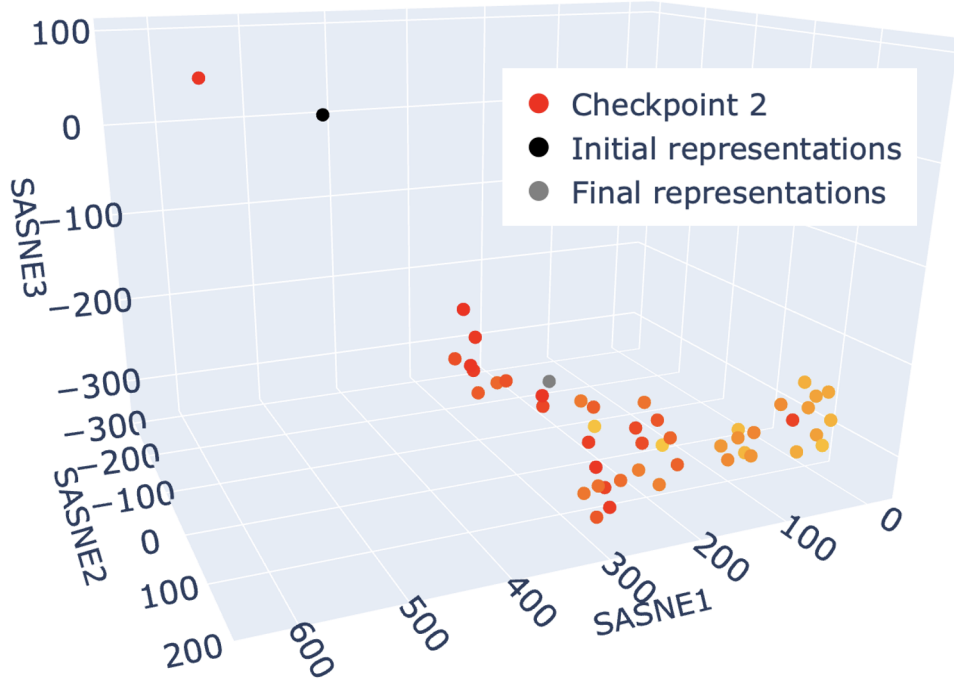


Figure B.2: **SASNE embedding of Checkpoint 2 without temporal trajectories.** Corresponding point-only embedding. The representations remain distributed along the same curved region identified in the main analysis, illustrating the directional refinement that characterizes the optimization of the second residual stage.

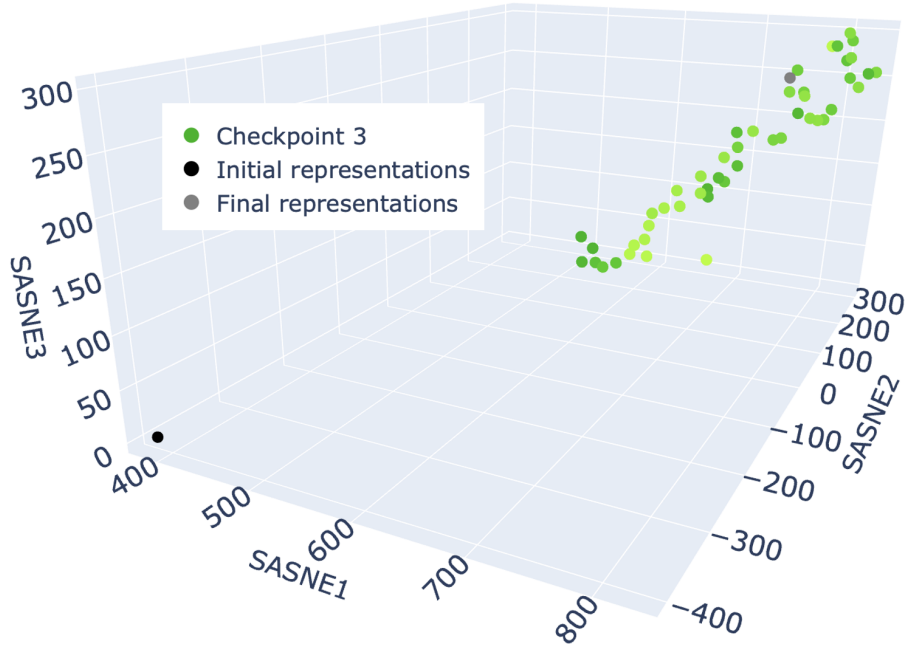


Figure B.3: **SASNE embedding of Checkpoint 3 without temporal trajectories.** Corresponding point-only embedding. The spatial organization confirms that the third residual stage follows the same elongated representational corridor observed for the preceding checkpoints. It is clear how there are two clusters generated. The smaller cluster belongs to the late epochs of the optimization procedure.

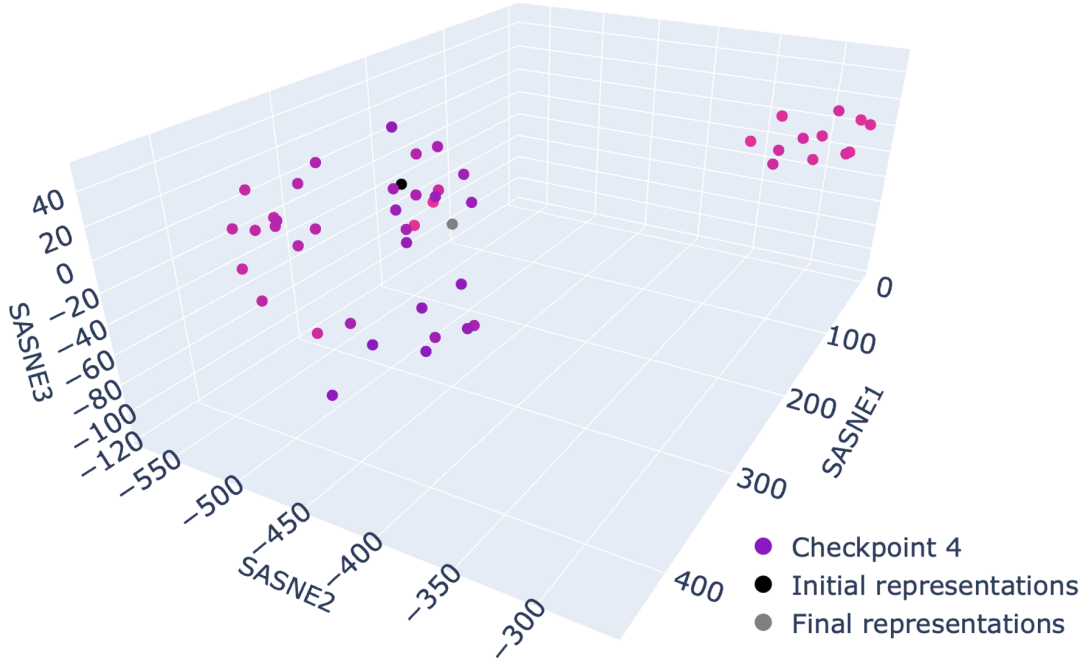


Figure B.4: **SASNE embedding of Checkpoint 4 without temporal trajectories.** Corresponding point-only embedding. Unlike the shallow checkpoints, the representations occupy a considerably more compact region of the manifold while remaining partitioned into early and late optimization stages.

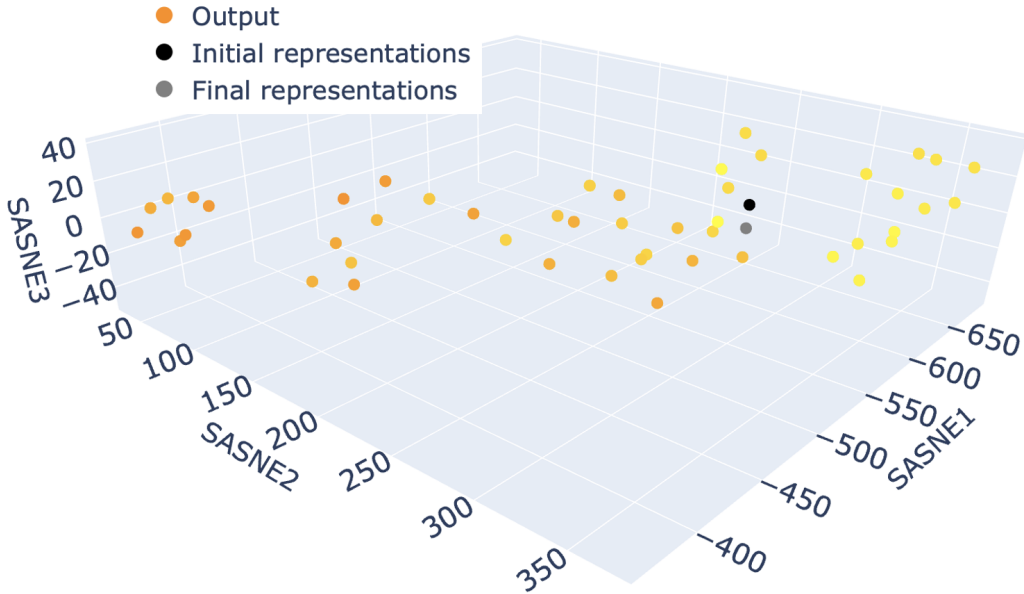


Figure B.5: **SASNE embedding of the output representation without temporal trajectories.** Corresponding point-only embedding. The clear separation between early and late epochs mirrors the behavior observed in Checkpoint 4, providing additional evidence that the deepest representations share a common optimization dynamic distinct from that of the earlier network stages.

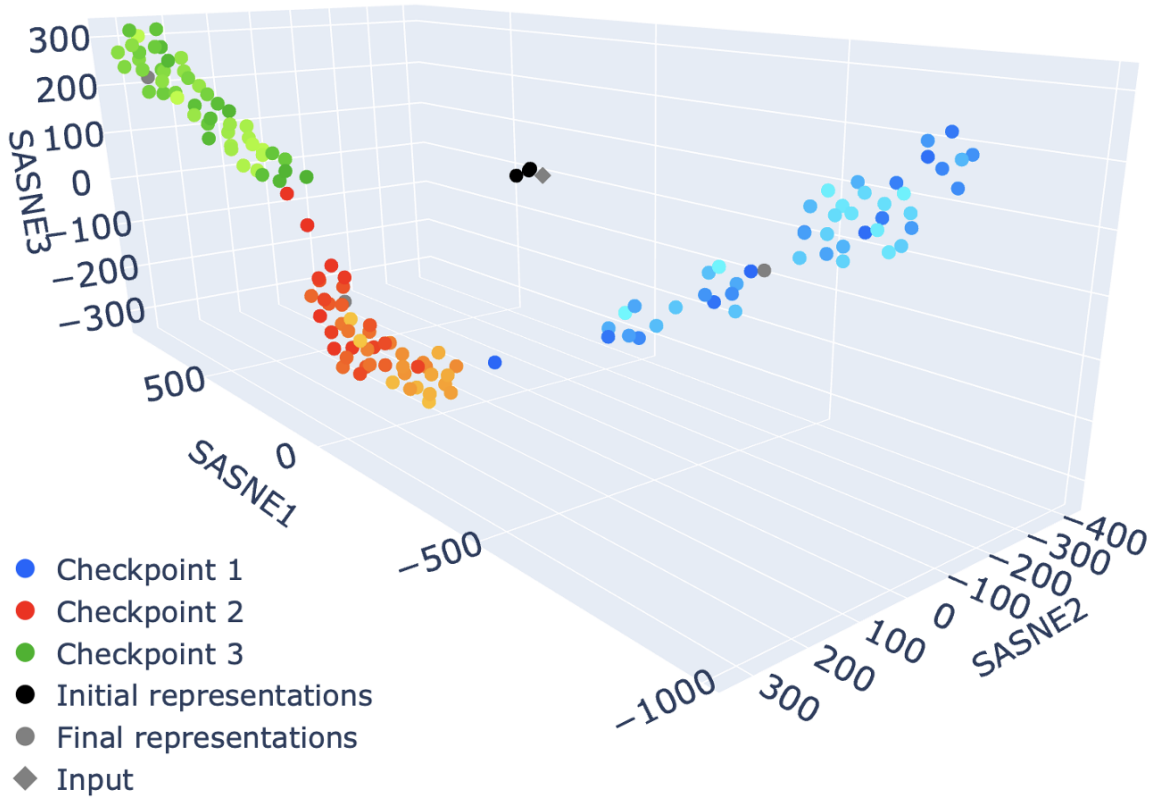


Figure B.6: **Joint embedding of the shallow checkpoints without temporal trajectories.** Corresponding point-cloud visualization obtained after removing the connecting lines. Colors identify the network checkpoints, while progressively lighter shades indicate increasing training epochs. Black markers denote the earliest trainable representations and gray markers the final epochs. The point-cloud representation reveals that the elongated manifolds observed for the shallower checkpoints are intrinsic geometric structures rather than artifacts of the trajectory visualization. Furthermore, the three checkpoints occupy neighboring curved regions that exhibit similar temporal organization, supporting the interpretation that the shallow checkpoints of the network evolve through a common refinement process while preserving geometrically distinct representations.

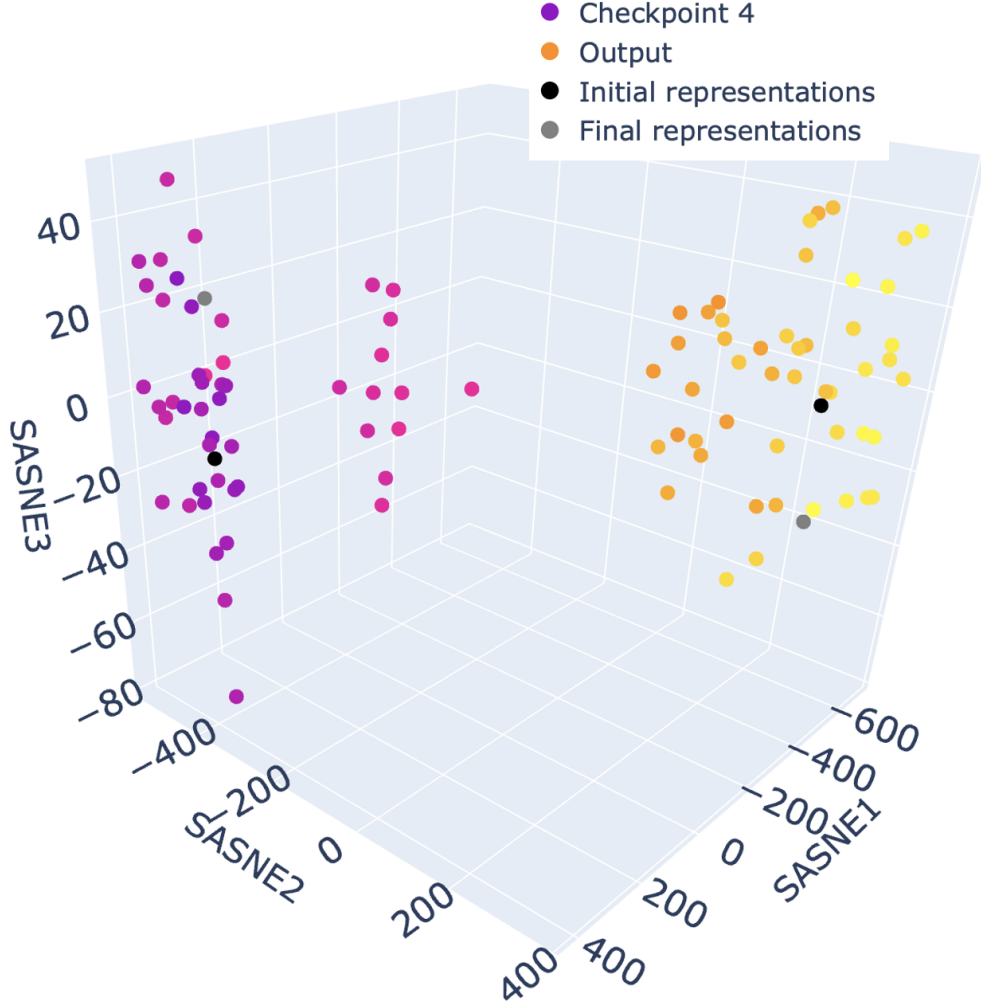


Figure B.7: **Joint SASNE embedding of Checkpoint 4 and the output representation without temporal trajectories.** Corresponding point-cloud embedding obtained after removing the temporal connections between consecutive epochs. The figure highlights the strong geometric similarity between the two deepest network representations. In contrast to the elongated manifolds observed for the shallow checkpoints, both representations occupy relatively compact regions of the embedding and exhibit a clear temporal partition into early and late stages of optimization.

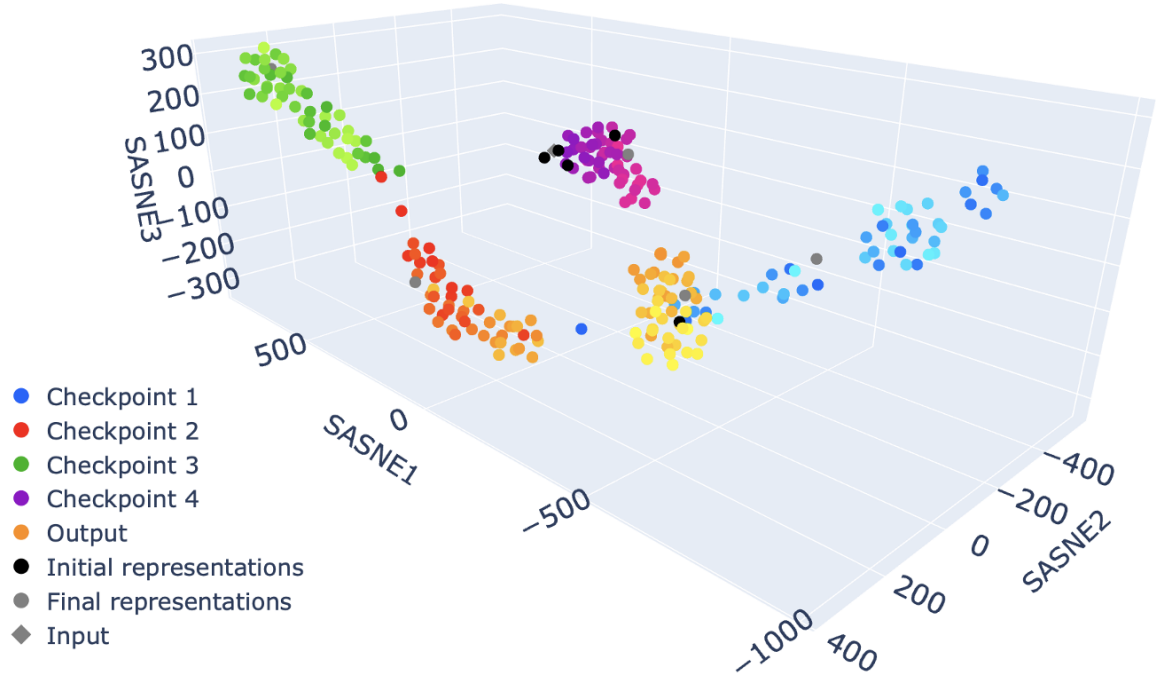


Figure B.8: **All-checkpoint SASNE embedding of the complete Representation Graph Family with and without temporal trajectories.** Corresponding point-only embedding. It emphasizes the spatial separation between checkpoints and highlights the two large representational regimes formed by the early and late network stages. Visualizes that the geometric organization revealed by SAGD is an inherent property of the Representation Graph Family and not exclusively a consequence of the trajectory representation.

Bibliography

- [1] Simon Kornblith et al. «Similarity of Neural Network Representations Revisited». In: *Proceedings of the 36th International Conference on Machine Learning (ICML)*. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 3519–3529.
- [2] Maithra Raghu et al. «SVCCA: Singular Vector Canonical Correlation Analysis for Deep Learning Dynamics and Interpretability». In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017.
- [3] Y. Li et al. «Convergent Learning: Do different neural networks learn the same representations?» In: *International Conference on Learning Representations (ICLR)*. May 2016.
- [4] An-Min Chen, Hsiao-Ming Lu, and Robert Hecht-Nielsen. «On the Geometry of Feedforward Neural Network Error Surfaces». In: *Neural Computation* 5.6 (1993), pp. 910–927.
- [5] Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. «A global geometric framework for nonlinear dimensionality reduction». In: *Science* 290.5500 (2000), pp. 2319–2323.
- [6] Ulrike von Luxburg. *A Tutorial on Spectral Clustering*. 2007. arXiv: 0711.0189 [cs.DS]. URL: <https://arxiv.org/abs/0711.0189>.
- [7] Francois Fouss et al. «Random-Walk Computation of Similarities between Nodes of a Graph with Application to Collaborative Recommendation». In: *IEEE Transactions on Knowledge and Data Engineering* 19.3 (2007), pp. 355–369. DOI: 10.1109/TKDE.2007.49.
- [8] Francois Fouss et al. *A Novel Way of Computing Dissimilarities between Nodes of a Graph, with Application to Collaborative Filtering and Subspace Projection of the Graph Nodes*. Tech. rep. Technical Report, ICTEAM, Université Catholique de Louvain, 2006.
- [9] D. J. Klein and M. Randić. «Resistance Distance». In: *Journal of Mathematical Chemistry* 12.1 (1993), pp. 81–95. ISSN: 1572-8897. DOI: 10.1007/BF01164627. URL: <https://doi.org/10.1007/BF01164627>.

- [10] Sergey G. Bobkov and Michel Ledoux. *One-dimensional empirical measures, order statistics, and Kantorovich transport distances*. Vol. 261. Memoirs of the American Mathematical Society. American Mathematical Society, 2014.
- [11] G. Dall’Aglio. «Sugli estremi dei momenti delle funzioni di ripartizione doppia». Italian. In: *Annali della Scuola Normale Superiore di Pisa (3)* 10 (1956), pp. 35–74.
- [12] Kaiming He et al. «Deep Residual Learning for Image Recognition». In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778.
- [13] Tobias Wängberg, Joanna Tyrcha, and Chun-Biu Li. «Shape-aware stochastic neighbor embedding for robust data visualisations». In: *BMC Bioinformatics* 23.1 (2022), p. 477. DOI: 10.1186/s12859-022-05028-8.
- [14] Geoffrey E. Hinton and Sam T. Roweis. «Stochastic Neighbor Embedding». In: *Advances in Neural Information Processing Systems (NIPS)*. Vol. 15. 2002, pp. 857–864.
- [15] Laurens van der Maaten and Geoffrey Hinton. «Visualizing Data using t-SNE». In: *Journal of Machine Learning Research* 9 (2008), pp. 2579–2605.
- [16] Ronald A. Fisher. «The use of multiple measurements in taxonomic problems». In: *Annals of Eugenics* 7.2 (1936), pp. 179–188.
- [17] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. URL: <https://www.deeplearningbook.org>.
- [18] Cédric Villani. *Optimal Transport: Old and New*. Vol. 338. Grundlehren der mathematischen Wissenschaften. Springer, 2008.
- [19] Martin Arjovsky, Soumith Chintala, and Léon Bottou. «Wasserstein GAN». In: *Proceedings of the 34th International Conference on Machine Learning (ICML)*. PMLR. 2017, pp. 214–223.
- [20] A. Emin Orhan and Xaq Pitkow. «Skip Connections Eliminate Singularities». In: *International Conference on Learning Representations (ICLR)*. 2018.
- [21] Martin Wattenberg, Fernanda Viégas, and Ian Johnson. «How to Use t-SNE Effectively». In: *Distill* 1.10 (2016), e2. DOI: 10.23915/distill.00002. URL: <https://distill.pub/2016/misread-tsne/>.
- [22] Yaron Lipman, Raif M. Rustamov, and Thomas A. Funkhouser. «Biharmonic Distance». In: *ACM Transactions on Graphics* 29.3 (2010), pp. 1–11. DOI: 10.1145/1805964.1805971.
- [23] Yulong Wei, Rong-hua Li, and Weihua Yang. «Biharmonic distance of graphs». In: *arXiv preprint arXiv:2110.02656* (2021).

- [24] Duncan J. Watts and Steven H. Strogatz. «Collective dynamics of "small-world" networks». In: *Nature* 393.6684 (1998), pp. 440–442. DOI: 10.1038/30918.
- [25] Yutaka Shimada et al. «Graph distance for complex networks». In: *Scientific Reports* 6 (2016), p. 34944. DOI: 10.1038/srep34944. URL: <https://doi.org/10.1038/srep34944>.
- [26] Alain Barrat and Martin Weigt. «On the properties of small-world network models». In: *The European Physical Journal B* 13.3 (2000), pp. 547–560. DOI: 10.1007/s100510050067.

Masteruppsats 2026:18
Matematisk statistik
September 2026

www.math.su.se

Matematisk statistik
Matematiska institutionen
Stockholms universitet
106 91 Stockholm