

DISSERTATION

EFFICIENT AND TRANSFERABLE GRAPH LEARNING VIA ANCHOR-BASED
STRUCTURAL ENCODINGS

Submitted by

Zheyi Qin

Department of Electrical and Computer Engineering

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Summer 2026

Doctoral Committee:

Advisor: Anura Jayasumana

Edwin Chong

Randy Paffenroth

Indrakshi Ray

Michael Kirby

Copyright by Zheyi Qin 2026

All Rights Reserved

ABSTRACT

EFFICIENT AND TRANSFERABLE GRAPH LEARNING VIA ANCHOR-BASED STRUCTURAL ENCODINGS

Graph-structured data arise in many domains, including biological networks, social systems, recommendation platforms, and knowledge graphs. While graph neural networks (GNNs) have become a dominant framework for learning from such data, their effectiveness often depends on iterative message passing, substantial computational cost, and assumptions that may limit scalability and robustness in large or weakly attributed graphs. In large graphs, message passing may require repeated neighborhood aggregation across multiple layers to capture non-local structural information, which can increase latency and resource consumption while also introducing problems such as over-smoothing, loss of node-level distinction, and sensitivity to weak or missing node attributes. These limitations are especially important for resource-constrained settings such as IoT devices, mobile systems, and large-scale scientific graph analysis, where accuracy alone is not sufficient and the trade-off among performance, parameter count, computation time, and scalability becomes central.

Graph learning does not have to depend only on iterative message passing. Prior work and empirical evidence suggest that graph-distance structure in many practical benchmark graphs often exhibits approximately low-dimensional or low-rank regularity. This dissertation uses that property as a design principle: rather than constructing dense all-pairs distance matrices or repeatedly propagating information through GNN layers, it samples graph-distance structure through anchors, compresses it into structural coordinates, and reuses the resulting representations across prediction tasks and anchor-induced views. The novelty of this work lies not in the existence of landmark distances alone, but in developing them into a unified, scalable graph-learning framework: sampled graph distances are compressed into structural coordinates, reused across prediction tasks and

anchor-induced views, and evaluated as a resource-efficient alternative or complement to message-passing GNNs. This provides a scalable and Green-AI-oriented path for graph learning, especially when full GNNs are too expensive or when node features are weak.

This dissertation develops and evaluates scalable anchor-based structural encodings as an alternative and complement to conventional message-passing GNN pipelines. The central idea is to represent nodes through their relationships to selected anchor sets or landmark nodes, producing structural coordinates that capture graph position and topology even when node attributes are limited, noisy, or unavailable. Building on this idea, the dissertation investigates several coordinate-based representations, including virtual coordinates (VC), topology coordinates (TC), and directional virtual coordinates (DVC), and demonstrates how these representations can be compressed, aligned, and used in downstream prediction tasks. The central goal is to preserve useful local and global topological information while reducing or eliminating dependence on deep message-passing architectures and avoiding dense all-pairs graph computations.

The main contribution of this dissertation is a scalable structural-learning framework that encodes graph topology through anchor-based representations rather than relying solely on iterative message passing. By representing each node through its distances or coordinate relationships to a selected set of anchors, the proposed VC, TC, and DVC methods provide compact structural features that capture both local neighborhood structure and non-local topological relationships. These features can be used with conventional neural predictors for node classification and link prediction. Empirically, these methods show a favorable accuracy–efficiency trade-off. On OGBN-Products, TC-based models match or slightly exceed comparable GCN performance while using roughly 78% fewer trainable parameters, and a smaller TC configuration remains within about one percentage point of GCN accuracy while using roughly 91% fewer parameters. In the selected OGBN-Products comparison, the nearest higher-ranked listed baseline uses about 2.1×10^5 trainable parameters, while the rank-1 listed model uses about 1.4×10^8 , placing the highest-ranked model at a much larger parameter scale. On OGBN-Proteins, compact TC models achieve stronger ROC-AUC than GCN while using about 89% fewer parameters, and achieve performance compa-

rable to GeniePath-BS while using about 97% fewer parameters. In the selected OGBN-Proteins comparison, the nearest higher-ranked listed baseline uses about 4.9×10^5 trainable parameters, while the rank-1 listed model uses about 6.6×10^8 . These results suggest that the proposed methods are not primarily intended to replace the highest-accuracy leaderboard models, but instead provide a resource-adaptable alternative when the goal is to preserve strong predictive performance with substantially lower model complexity. The framework is especially applicable to large sparse graphs, biological and scientific networks, recommendation and social networks, and edge or IoT settings where graph structure is informative but memory, computation, latency, or node-feature quality limits the use of deeper message-passing models. Beyond single-task prediction, the dissertation also contributes scalable construction methods, including anchor sampling, dimensionality reduction, chunked computation, and partition-aware processing, as well as transfer analysis across anchor-induced views. The strongest transfer evidence is for reusing models across different views of the same graph or overlapping subgraphs through lightweight orthogonal Procrustes alignment, while transfer across substantially different graph domains remains a harder open problem.

Across these experiments, the results show that anchor- and distance-based structural representations provide a viable and flexible alternative or complement to traditional GNN pipelines. They are especially useful in settings where graph structure is informative, node features are limited, or scalability is a primary concern. A key practical advantage of this approach is that graph topology can be converted into reusable structural features before downstream learning, allowing conventional neural predictors such as multilayer perceptrons or lightweight classifiers to exploit graph information without performing iterative message passing during prediction. This separation between structural feature construction and neural prediction supports a Green-AI-oriented perspective, in which graph learning methods are evaluated not only by predictive accuracy but also by parameter efficiency, computational cost, memory use, and deployability. More broadly, this dissertation contributes a structural learning perspective that connects graph topology, representation design, transfer, and practical deployment, and it offers guidance for developing graph learning systems that are both effective and computationally manageable.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor, Anura Jayasumana, for their guidance, patience, and support throughout my doctoral studies. Their insight, encouragement, and high standards have shaped not only this dissertation but also my growth as a researcher. I am especially grateful for the freedom to explore new ideas, the thoughtful feedback provided at every stage of this work, and the steady encouragement during the more difficult parts of the Ph.D. journey.

I would also like to thank my committee members, Edwin Chong, Randy Paffenroth, Indrakshi Ray, and Michael Kirby, for their time, feedback, and valuable suggestions. Their questions and perspectives helped strengthen this dissertation and broaden my thinking about graph learning research.

My appreciation also goes to my collaborators, lab mates, and colleagues in the CNRL. I have benefited greatly from your discussions, technical advice, and encouragement. Whether through research conversations, classes, teaching, or everyday support, you helped make graduate school a meaningful and enriching experience.

I am grateful to the faculty and staff of the ECE for their support throughout my program. I also appreciate the students I worked with as a teaching assistant, as teaching helped me grow in clarity, patience, and confidence.

I would like to acknowledge the support of ACCESS Allocations. Access to computational resources and research support made this work possible.

Finally, and most importantly, I would like to thank my family and friends for their love, understanding, and constant encouragement. Your support sustained me throughout this journey. This dissertation is dedicated to you.

DEDICATION

To my family and friends, whose encouragement sustained me throughout this journey.

TABLE OF CONTENTS

ABSTRACT	ii
ACKNOWLEDGEMENTS	v
DEDICATION	vi
LIST OF TABLES	xii
LIST OF FIGURES	xiii
LIST OF ACRONYMS	xiv
 Chapter 1 Introduction	 1
1.1 Motivation	1
1.2 Research Questions and Scope	6
1.2.1 Research Questions	6
1.2.2 Scope and Assumptions	7
1.3 Contributions	8
1.4 Dissertation Organization	10
 Chapter 2 Problem Statement	 12
2.1 Graph Neural Networks: Advantages and Limitations	12
2.2 Problem Statement	13
2.3 Anchor-Based Structural Representation Setting	15
2.4 Multiple-View Structural Representation Problem	16
2.5 Evaluation Problem	17
2.6 Summary	17
 Chapter 3 Background and Preliminaries	 18
3.1 Notation and Definitions	18
3.1.1 Sets, Indices, and Basic Conventions	18
3.1.2 Graphs	18
3.1.3 Node Features, Labels, and Splits	19
3.1.4 Distances, Anchors, and Distance Matrices	19
3.1.5 Dimensionality Reduction and Representations	20
3.1.6 Predictive Models and Loss Functions	20
3.1.7 Evaluation Metrics	21
3.1.8 Subgraphs, Partitions, and Connected Components	21
3.2 Problem Settings in Graph Learning	21
3.2.1 Data Modalities: Attributed vs. Non-Attributed Graphs	22
3.2.2 Node Classification	23
3.2.3 Link Prediction	23
3.2.4 Graph-Level Prediction	24
3.2.5 Self-Supervised and Pretext Objectives	25
3.2.6 Scalability and Computational Constraints	25
3.2.7 Message-Passing GNNs and Embedding-Based Neural Networks	26

3.2.8	Summary	26
3.3	Distance-Based and Anchor-Based Representations	27
3.3.1	Full Pairwise Distance Representations	27
3.3.2	Anchor-Based Approximations	28
3.3.3	Anchor Selection Strategies	28
3.3.4	From Distances to Features	29
3.3.5	Multiple Structural Views from Anchors and Graph Layers	30
3.3.6	Practical Considerations: Complexity and Robustness	30
3.3.7	Summary	31
3.4	Evaluation Protocols and Metrics	31
3.4.1	General Principles and Reproducibility	31
3.4.2	Transductive Protocols and Information Access	32
3.4.3	Splitting Strategies	32
3.4.4	Node Classification Metrics	33
3.4.5	Link Prediction Metrics	33
3.4.6	Early Stopping and Model Selection	34
3.4.7	Reporting Format	34
3.4.8	Summary	34
Chapter 4	Related Work	35
4.1	Graph Neural Networks and Message Passing	35
4.2	Structural and Positional Encodings	37
4.3	Distance- and Landmark-Based Methods	39
4.4	Cross-View and Cross-Graph Transfer Learning	42
4.5	Scalable Graph Learning	44
4.6	Synthetic Graph Generation and Benchmarking	47
Chapter 5	Anchor-Based Representations for Graph Learning	49
5.1	Overview	49
5.2	Problem Formulation and Design Goals	50
5.2.1	Graph Learning Tasks	50
5.2.2	Anchor-Based Representation: Problem Statement	51
5.2.3	Design Goals	52
5.2.4	Chapter Focus	53
5.3	Anchor Selection and Sampling Strategies	54
5.3.1	Notation and General Considerations	54
5.3.2	Uniform Random Sampling	54
5.3.3	Degree-Biased Sampling	55
5.3.4	Stratified Sampling by Degree or Partitions	55
5.3.5	Farthest-Point (Diversity-Promoting) Sampling	56
5.3.6	Centrality-Driven Selection	56
5.3.7	Practical Strategy Used in This Dissertation	57
5.3.8	Summary of Trade-offs	57
5.4	Representation Construction Pipeline	57
5.4.1	Distance Matrix Construction	58

5.4.2	Dimensionality Reduction (PCA)	59
5.5	Computational Complexity and Resource Analysis	60
5.5.1	Cost Model and Notation	61
5.5.2	Anchor-to-Node Distance Construction	61
5.5.3	Dimensionality Reduction Costs	62
5.5.4	Downstream Training Costs	63
5.5.5	Practical Resource Management	64
5.5.6	Summary	64
5.6	Experimental Setup	64
5.6.1	Datasets	65
5.6.2	Baselines	66
5.6.3	Implementation Details	67
5.7	Results	69
5.7.1	Node Classification Performance	69
5.7.2	Topology Coordinates and Directional Virtual Coordinates	70
5.7.3	Virtual Coordinates as a Direct Embedding Baseline	72
5.8	Limitations and Discussion	72
5.8.1	Representation Limits and Structural Ambiguity	73
5.8.2	Sensitivity to Anchor Choice	74
5.8.3	Normalization and Comparability Issues	75
5.8.4	Computational Constraints and Practical Bottlenecks	76
5.8.5	Task-Specific Limitations	76
5.8.6	Discussion and Potential Extensions	77
5.8.7	Summary	78
Chapter 6	Link Prediction	79
6.1	Overview	79
6.2	Task Definition and Data Construction	81
6.2.1	Problem Setup	82
6.2.2	Directed vs. Undirected Graphs	83
6.2.3	Training Graph and Message-Passing Scope	83
6.2.4	Constructing Edge Examples	84
6.2.5	Candidate Sets for Evaluation	84
6.2.6	Scalable Data Representation	85
6.3	Splits and Evaluation Protocol	85
6.3.1	Edge Splitting	86
6.3.2	Validation and Test Queries	86
6.3.3	Negative Sets for Evaluation	87
6.3.4	Ranking Metrics	88
6.3.5	Data Leakage Prevention	88
6.4	Negative Sampling Strategy	89
6.4.1	Sampled Negatives and Filtering	89
6.4.2	Uniform and Degree-Based Negative Sampling	90
6.4.3	On-the-Fly vs. Precomputed Negatives	90
6.4.4	Choosing the Negative Ratio ρ_{neg}	91

6.5	Model and Training Procedure	92
6.5.1	Node Representations	92
6.5.2	Edge Scoring Functions	94
6.5.3	Loss Function with Negative Sampling	95
6.5.4	Mini-batch Construction and Vectorized Scoring	96
6.5.5	Optimization and Training Schedule	97
6.5.6	Scalable Training Pipeline	97
6.5.7	Algorithmic Summary	98
6.6	Experimental Setup	98
6.6.1	Datasets	99
6.6.2	Baselines	100
6.6.3	Training and Evaluation Protocol	100
6.6.4	Implementation Details	100
6.6.5	Hyperparameters	101
6.7	Results	102
6.7.1	Synthetic Graph Results	102
6.7.2	Real-World Benchmark Results	103
6.7.3	No-Leak Coordinate Ensembles for OGBL-DDI Link Prediction	103
6.7.4	Discussion of Observed Trends	109
6.8	Error Analysis and Discussion	110
6.8.1	Error Taxonomy	110
6.8.2	Diagnostics and What to Log	112
6.8.3	Qualitative Error Patterns	112
6.8.4	Discussion: Trade-offs and Practical Recommendations	113
Chapter 7	Train-on-One and Train-on-Many Transfer Across Anchor Sets and Graphs	115
7.1	Chapter Overview	115
7.2	Motivation and Use Cases	116
7.2.1	Why Train-on-One, Evaluate-on-Many?	116
7.2.2	Robustness to Anchor Choice and Graph Variations	117
7.2.3	Computational Benefits and Deployment Considerations	118
7.3	Problem Setting and Notation	118
7.3.1	Graphs, Tasks, and Splits	118
7.3.2	Anchor Sets and Distance Constructions	119
7.3.3	Representations as Multiple Views	120
7.4	Sources of Variation	120
7.4.1	Anchor-Set Variability (Sampling, Size, Coverage)	120
7.4.2	Representation Variability	121
7.4.3	Graph Variability (Perturbations, Subgraphs, Synthetic Families)	123
7.4.4	When Train-on-One is Expected to Fail	123
7.5	Train-on-One, Evaluate-on-Many Protocol	124
7.5.1	Reference Anchor Set Selection	126
7.5.2	Training Procedure on the Reference View	126
7.5.3	Evaluation on Multiple Anchor Sets for the Same Graph	127
7.5.4	Evaluation Across Multiple Graphs	127

7.6	Methods for Cross-Graph Transfer	128
7.6.1	Inductive vs Transductive Assumptions	128
7.6.2	Potential Anchor Strategies for Cross-Graph Transfer	129
7.7	Implementation Details	131
7.7.1	Data Pipeline	132
7.7.2	Model Architectures and Training Hyperparameters	133
7.7.3	Alignment Implementation Notes	135
7.8	Experiments	137
7.8.1	Datasets and Experimental Setup	138
7.8.2	Baselines and Comparisons	138
7.8.3	Metrics and Reporting Protocol	139
7.8.4	Results and Analysis	139
7.9	Discussion	150
7.9.1	When Train-on-One Appears Promising	150
7.9.2	Why Alignment Must Be Interpreted Carefully	151
7.9.3	Node Features May Currently Dominate Transfer	151
7.9.4	What Multi-Source Training Currently Tells Us	152
7.9.5	Updated Reporting Guidance	152
7.9.6	Limitations	152
7.10	Summary	153
Chapter 8	Discussion, Limitations and Future Work	154
8.1	Summary of Findings	154
8.2	Guidelines for Applying Anchor-Based Structural Representations	158
8.3	Contributions and Advantages	162
8.4	Limitations	164
8.5	Broader Impacts and Ethical Considerations	166
8.6	Future Work	168
8.7	Conclusions	171
Bibliography		173
Appendix A	Justification for Orthogonal Procrustes Alignment	185
Appendix B	Codebase	188
B.1	Graph Coordinate Generation	188
B.2	Model Training and Evaluation	188
B.3	Transfer and Alignment Experiments	189
B.4	Supplemental Implementation Notes	189

LIST OF TABLES

5.1	Qualitative trade-offs of anchor selection strategies.	57
5.2	Comparison of selected OGBN leaderboard entries for OGBN-Products. Blank value indicates that the rank is not listed in the official leaderboard.	71
5.3	Comparison of selected OGBN leaderboard entries for OGBN-Proteins. Blank value indicates that the rank is not listed in the official leaderboard.	71
5.4	Ambiguity diagnostics for virtual-coordinate representations on OGBN-Products with 1000 anchors.	74
6.1	Dataset statistics.	99
6.2	Synthetic graph results (testHits@1). “w/ transf.” denotes runs using TC or DVC embeddings; “VC only” denotes runs using raw VC. Each entry is a mean over all matching configurations.	102
6.3	OGBL-DDI results (testHits@20). ρ_{neg} denotes the ratio of negative to positive training samples.	103
6.4	Main no-leak coordinate-ensemble results on full OGBL-DDI.	106
6.5	Active channels in the 8-channel z-score ensemble. All rows use the no-leak protocol with the training positives as the training-negative forbidden set.	107
6.6	Channels in the best under-1M coordinate ensemble. The total parameter count is 937,410.	108
7.1	Train-on-one, evaluate-on-another across anchor sets. The model is trained on anchor set 1 and then evaluated (with no retraining) on other anchor views after Procrustes alignment to the baseline frame.	144
7.2	Train-on-one, evaluate-on-another across anchor sets. The model is trained on anchor set 1 and then evaluated (with no retraining) on other anchor views after Procrustes alignment to the baseline frame, with #Oanc anchors overlapping with anchors in set 1.	144
7.3	Train-on-one, evaluate-on-another across anchor sets. The model is trained on anchor set 1 and then evaluated (without retraining) on other anchor views after Procrustes alignment to the baseline frame, with #Anchor nodes as reference nodes, where #Oref reference nodes overlapping with anchors in set 1.	145
7.4	Train-on-one, evaluate-on-another across subgraphs. The model is trained on subgraph 1 and then evaluated (without retraining) on other subgraphs after Procrustes alignment to the baseline frame.	149
7.5	Single-source cross-dataset transfer results for the cross-attention model on Cornell and Texas. Values are single-run test accuracy and should be interpreted as exploratory rather than final.	149
7.6	Multi-source transfer results to acmv9. Values are single-run test accuracy.	150
7.7	Cornell + Wisconsin $\rightarrow$ Texas results. These numbers are retained as diagnostic observations because the corresponding implementation was not fully stabilized.	150

LIST OF FIGURES

1.1	Big-picture view: anchor-based structural representations constructed from a graph and used for downstream graph learning tasks.	2
3.1	Problem settings: node classification and link prediction.	23
5.1	Anchor-based representation construction pipeline.	58
5.2	VCNN performance on OGBN-Products as the number of virtual coordinates and hidden-layer configuration vary. Each point is annotated with the number of virtual coordinates.	69
5.3	VCNN performance on OGBN-Proteins as the number of virtual coordinates and hidden-layer configuration vary. Each point is annotated with the number of virtual coordinates.	70
5.4	Leading singular values of partial distance matrices from ten random 1000-anchor sets on OGBN-Products. The close agreement across anchor sets indicates that random anchors recover a similar dominant structural subspace.	74
6.1	Procedure for link prediction with sampled negatives: node representation computing, filtered negative sampling, edge scoring, loss computation, and parameter updates, with periodic evaluation.	93
7.1	Train-on-One, Evaluate-on-Many pipeline. A predictor f_θ is trained on a reference representation view defined by an anchor set and coordinate construction, then evaluated without retraining on alternative anchor-induced views and, when applicable, related graph instances.	125
7.2	Implementation-level pipeline used in the transfer experiments. Node features feed both the linear projection and the positional encoding, while anchor-derived information is aggregated into a context vector before fusion in the main model.	132
7.3	Cross-attention variant of the main model. Each node forms a query from $\mathbf{h}_v$ (optionally concatenated with $\mathbf{e}_v$), and attends over anchor tokens $\{\mathbf{c}_a\}_{a \in \mathcal{A}}$ to produce a context-aware embedding $\mathbf{z}_v$	134
7.4	TC plots of anchor set 1 (baseline).	140
7.5	TC plots of anchor set 2, before applying Procrustes.	142
7.6	TC plots of anchor set 2, after applying Procrustes.	142
7.7	TC plots of anchor set 3, before applying Procrustes.	143
7.8	TC plots of anchor set 3, after applying Procrustes.	143
7.9	TC plots of subgraph 1 (baseline).	146
7.10	TC plots of subgraph 2, before applying Procrustes.	147
7.11	TC plots of subgraph 2, after applying Procrustes.	147
7.12	TC plots of subgraph 3, before applying Procrustes.	148
7.13	TC plots of subgraph 3, after applying Procrustes.	148

LIST OF ACRONYMS

AI	Artificial Intelligence
AP	Average Precision
BFS	Breadth-First Search
CPU	Central Processing Unit
DVC	Directional Virtual Coordinates
DVCNN	Directional Virtual Coordinate Neural Network
GCN	Graph Convolutional Network
GCNN	Graph Coordinates Neural Network
GNN	Graph Neural Network
IoT	Internet of Things
LCC	Largest Connected Component
MLP	Multilayer Perceptron
MPNN	Message Passing Neural Network
OGB	Open Graph Benchmark
OGBN	Open Graph Benchmark Node Property Prediction
OGBL	Open Graph Benchmark Link Property Prediction
PCA	Principal Component Analysis
ROC-AUC	Area Under the Receiver Operating Characteristic Curve
SVD	Singular Value Decomposition
TC	Topology Coordinates
TCNN	Topology Coordinate Neural Network
VC	Virtual Coordinates

VCNN

Virtual Coordinate Neural Network

Chapter 1

Introduction

1.1 Motivation

Graphs provide a natural representation for many real-world systems in which data are inherently relational, distributed, and resource constrained. Examples include IoT and sensor networks, communication networks, mobile and edge-computing systems, molecular interactions, biological networks, knowledge graphs, and large-scale recommendation platforms. In many of these settings, information is not stored or processed in a single dense table; instead, it is distributed across nodes, devices, users, entities, databases, or interacting components, often under constraints on memory, computation, communication bandwidth, latency, or energy consumption. As a result, graph learning is not only a question of predictive accuracy, but also a question of data access and data movement: methods based on repeated neighborhood aggregation may require repeated access to graph neighborhoods, movement of node features or embeddings across memory hierarchies, or communication across devices, partitions, or storage systems. The learning objective depends not only on node- or edge-level attributes, but also on *relational structure*: predicting missing links, classifying entities, detecting functional roles, or inferring system behavior from patterns of connectivity. This motivates representation learning methods that can encode both local neighborhood information and broader graph topology in a form suitable for statistical learning under practical resource constraints.

Graph Neural Networks (GNNs) have become a dominant paradigm for graph representation learning due to their end-to-end differentiability and their ability to exploit message passing over edges. However, the same message-passing mechanism that makes GNNs effective can also make them costly in large, weakly attributed, or distributed graphs. First, many deployed graphs contain weak, noisy, or even absent node features, forcing models to rely heavily on structure alone. Second, message passing can struggle to capture long-range dependencies efficiently, especially when

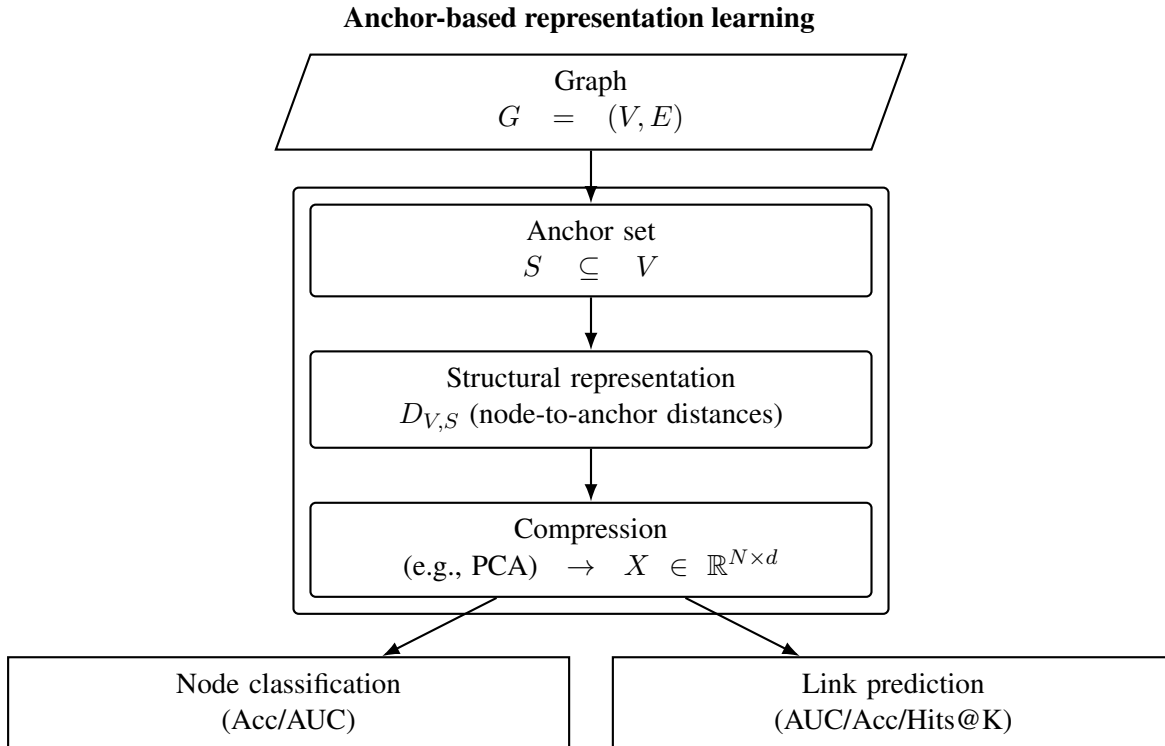


Figure 1.1: Big-picture view: anchor-based structural representations constructed from a graph and used for downstream graph learning tasks.

relevant signals are distributed across distant parts of the graph. Third, scaling to graphs with millions of nodes/edges often requires aggressive sampling, partitioning, or simplified architectures, which can weaken global structural awareness. These issues are amplified in scientific graphs (e.g., drug-drug interaction networks) where subtle global topology may correlate with function.

In parallel, classical approaches such as spectral methods, matrix factorization, graph embeddings, and coordinate-based representations provide principled ways to encode graph structure beyond immediate adjacency. These methods show that graph topology can often be represented through structural coordinates or latent spaces rather than through repeated local aggregation alone. However, many existing global-structure methods either require expensive dense computations or do not explicitly exploit the combination of sparse graph connectivity and approximately low-rank distance-derived structure as the main mechanism for scalability. From a distance-vector perspective, prior work on link dimension and exact graph construction shows that distances to appropriately chosen reference nodes can contain substantial structural information about a graph [1]. This

dissertation builds on that observation in a scalable learning setting: rather than constructing or storing a full all-pairs distance matrix, it samples a limited set of anchor nodes and uses node-to-anchor shortest-path distances as a tractable proxy for global topology. Figure 1.1 summarizes the overall anchor-based structural representation framework used throughout this dissertation.

The resulting approach exploits two empirical properties that frequently arise in large practical graphs. First, the adjacency structure is often sparse, which makes it feasible to compute distances from a limited number of anchors using graph traversals rather than from every node to every other node. Second, the resulting distance-derived structural signals often contain low-dimensional or approximately low-rank regularities, allowing sampled node-to-anchor distance matrices to be compressed through projection-based coordinates such as TC. In this way, the proposed framework retains much of the global structural sensitivity of distance-based representations while avoiding the memory and computation costs of dense all-pairs graph computations.

This dissertation is motivated by a central trade-off in graph learning: useful predictive information is often embedded in global graph topology, but directly using global structure can be computationally expensive, while standard message-passing methods often rely on repeated local aggregation. The goal of this work is to retain the structural sensitivity of distance-based graph representations while improving scalability, parameter efficiency, and deployability through anchor-based approximation and compact coordinate construction. This motivation has three related aspects, representation, computational, and systems perspectives.

First, from a representation perspective, graph learning methods should be able to capture structural information beyond immediate adjacency. A central idea is to approximate global graph geometry using a small set of anchors or landmark nodes and to construct compact representations from node-to-anchor distance relationships. By sampling anchors and computing anchor-based distance vectors, the model obtains a tractable proxy for global structure. Coordinate transformations such as VC, TC, and DVC, together with dimensionality reduction such as SVD or PCA, further convert these signals into representations that are efficient to store and fast to consume in downstream predictors.

Second, from a computational perspective, this work is motivated by the contrast between adjacency-based message passing and distance-based structural representation. Standard GNNs operate primarily on the adjacency structure of a graph, which is usually sparse and local: each layer aggregates information from immediate neighbors, and multiple layers are needed to propagate information over longer ranges. This design can increase computation time, memory usage, parameter complexity, and energy consumption, especially on large graphs. From the perspective of Green AI, this creates a need for graph learning methods that do not evaluate success by accuracy alone, but also by computational efficiency, model size, scalability, and deployability. Distance-based graph representations provide a useful alternative view. Although the full graph distance matrix is dense and expensive to store, it captures global relationships among nodes rather than only local adjacency. More importantly, in many practical graph datasets, the distance matrix exhibits low-rank structure, meaning that its dominant topological information can often be approximated from a small subset of its columns. This observation motivates the anchor-based approach studied in this dissertation: by sampling anchor nodes and using node-to-anchor distances, the method captures much of the useful global graph topology without constructing or learning from the full all-pairs distance matrix. Coordinate transformations such as VC, TC, and DVC then convert these sampled distance relationships into compact structural encodings that can be used by conventional neural models with lower computational cost than deep message-passing pipelines.

Finally, from a systems perspective, real-world graph learning differs from many simplified benchmark settings in how graph data are stored, accessed, and processed. In one common case, the graph is maintained in a centralized database or graph store, such as a social network, recommendation platform, biological interaction database, citation network, or knowledge graph. Even in this setting, the graph may be extremely large, sparse in adjacency form, noisy, incomplete, and continuously updated, making repeated multi-layer message passing expensive in memory, computation, and latency. In a second case, the graph is inherently distributed: nodes and edges may be spread across IoT devices, sensors, mobile systems, edge servers, or organizational data silos, and no single machine may have efficient access to the entire graph. Standard message-passing

GNNs are not always well matched to these two cases because they often assume repeated access to neighborhoods, centralized sampling or aggregation, and sufficient memory and communication bandwidth to support multi-hop propagation. These assumptions become restrictive when graph learning must be performed under limited power, storage, computation, data availability, or latency constraints. This motivates the anchor-based structural approach studied in this dissertation: instead of requiring repeated message passing over the full graph during prediction, graph topology is summarized through sampled distance relationships and compact coordinate representations that can be stored, compressed, transferred, and used by lightweight downstream models. This setting also highlights a practical advantage of anchor-based coordinates. Once a relatively small anchor set is selected, node-to-anchor coordinates can be constructed as a preprocessing step through anchor-rooted graph traversals, which are naturally parallelizable across anchors and can be implemented in a partition-aware manner. Although exact distance computation in a distributed graph may still require boundary communication across partitions, the resulting coordinates can be materialized, compressed, and reused by downstream models. Thus, communication and graph-access costs are concentrated in the structural feature-construction stage rather than repeated during every message-passing layer or prediction step. To evaluate such methods realistically, this dissertation uses both public benchmark graphs that reflect practical large-scale graph learning scenarios and synthetic graphs that allow controlled analysis of how structural properties affect performance.

Thesis Direction. This dissertation develops a resource-adaptable approach to graph learning based on anchor-sampled structural encodings. The central direction is to capture useful global topology through compact node-to-anchor distance and coordinate representations, rather than relying solely on repeated message passing over the full graph. This approach is designed for practical graph-learning settings in which graphs may be large, sparse, noisy, centrally stored in databases, or distributed across devices and systems. The work evaluates how these anchor-based representations support node classification and link prediction, how they can be scaled through sampling, compression, chunked computation, and partition-aware processing, and how they can

be reused across related structural views under realistic constraints on computation, memory, latency, and deployability.

1.2 Research Questions and Scope

This dissertation focuses on scalable methods for injecting global structural information into learning systems for graphs. The central hypothesis is that *anchor-based structural representations*, especially when examined across *multiple anchor-induced views* and, in some datasets, across *multiple graph layers*, can provide a practical and expressive alternative to dense all-pairs structural encodings and can complement (or partially substitute) message-passing GNNs in attribute-sparse or large-scale regimes.

1.2.1 Research Questions

The work is organized around the following research questions (RQs).

RQ1: Anchor-based structural approximation. *How can global graph structure be approximated using a small set of anchors without incurring quadratic cost?* This question covers anchor selection strategies, scalable computation of anchor-to-node operators, and the effect of anchor budget k on accuracy, memory, and runtime.

RQ2: Representation construction and compression. *How should anchor-based signals be transformed into compact embeddings that remain predictive?* This includes dimensionality reduction, normalization choices, and how compression interacts with distance-based coordinate constructions such as VC, TC, and DVC across graph families.

RQ3: Multiple structural views. *How should multiple anchor-induced views, and where naturally available multiple graph layers, be used in graph learning?* This question studies when different anchor sets on the same graph produce stable or complementary representations, when

simple reuse across views is possible, and how naturally layered datasets such as OGBN-Proteins can be represented using multiple graph layers over a shared node set.

RQ4: Generalization and transfer across graphs. *How stable are learned structural representations across datasets, graph regimes, and anchor-induced views?* This includes “train on one, evaluate on many” settings, sensitivity to anchor resampling, and whether models trained on one structural view can be reused on other views of the same graph or on related graph instances.

RQ5: Scalability and practical deployment constraints. *What are the computational trade-offs of the proposed framework in realistic training pipelines?* This covers chunked computation of structural features, memory behavior, partitioning/subgraph extraction, and end-to-end cost comparisons against common baselines under fixed resource budgets.

1.2.2 Scope and Assumptions

To keep the study focused and reproducible, this dissertation adopts the following scope.

Primary tasks. The methods are evaluated on **node classification** and **link prediction**, with an emphasis on settings where structural information is essential (e.g., limited or absent node features).

Graph types. The primary focus is on large, sparse graphs commonly used in machine learning benchmarks and scientific network analysis. Both real graphs (public benchmarks) and synthetic graphs (controlled topology) are included to disentangle confounding factors.

Operators considered. The work primarily considers shortest-path-based node-to-anchor distances and coordinate transformations derived from them, including VC, SVD/PCA-based TC, and DVC. Other graph similarities or kernels are discussed as background and possible future directions, but they are not the main experimental focus of this dissertation.

Model class. The downstream predictors are neural models that consume structural embeddings (and optionally node features). While message-passing GNNs are used as baselines or complements, the contributions emphasize *structural feature construction, compression, and reuse across views* rather than proposing a new message-passing rule.

In summary, the scope is to develop and evaluate scalable, anchor-based structural representations, to study their behavior across multiple anchor-induced views and graph layers when relevant, and to quantify when they provide accuracy and robustness benefits under realistic computational constraints.

1.3 Contributions

This dissertation makes the following contributions toward scalable structural learning on graphs:

1. **Anchor-based structural representation framework.** We formulate a practical framework that approximates global graph structure by representing each node through its relationships to a small set of anchors. Unlike message-passing GNNs, which use graph topology inside repeated neighborhood-aggregation layers, the proposed framework decouples topology encoding from downstream prediction by first converting graph structure into reusable node-level coordinates. These coordinates, including VC, TC, and DVC, can be used alone or coupled with node attributes, allowing downstream node classification to be treated as conventional feature-based learning over structural feature vectors. This provides a consistent and scalable pipeline for generating node representations under tight memory and runtime constraints.
2. **Scalable computation and training pipeline for large graphs.** We develop an end-to-end implementation strategy that enables anchor-based structural features to be computed and consumed at scale, including chunked feature generation, memory-aware data handling, and integration with neural training loops for node classification and link prediction. This pipeline is designed to operate on graphs where dense $n \times n$ operators are infeasible.

3. **Compressed structural embeddings via projection based methods.** We investigate projection based compression (e.g., PCA-style low-dimensional embeddings) for anchor-based structural vectors, and analyze how the compression level affects downstream performance. This contribution clarifies the trade-off between embedding dimensionality and predictive quality, and identifies regimes where compact projections retain most of the structural signal.
4. **Multiple structural views across anchor sets and graph layers.** We characterize how anchor-based representations behave across multiple views of the same graph, especially under different anchor sets, and how naturally layered datasets such as OGBN-Proteins can be represented through multiple graph layers defined over a shared node set.
5. **Alignment-based reuse across views.** We examine train-on-one, evaluate-on-many settings and show that, for different anchor-induced views of the same graph and for overlapping subgraphs, lightweight orthogonal Procrustes alignment can make structural representations more reusable without retraining the downstream predictor.
6. **Evaluation methodology across synthetic and real benchmarks.** We provide a systematic evaluation protocol spanning controlled synthetic graphs (to isolate topological effects) and realistic public benchmarks (to test scalability and generalization). The experiments characterize sensitivity to anchor sampling, view variation, and resource budgets, and establish empirical guidance on when anchor-based structural representations improve over standard baselines.
7. **Empirical insights and practical guidelines.** From extensive experiments, we distill actionable findings on selecting anchor budgets, choosing representation dimensionality, and configuring model complexity for stable performance. These guidelines are intended to help practitioners deploy structural learning methods under real-world compute constraints.

The remainder of this dissertation elaborates these contributions, beginning with background and related work, followed by the proposed methods, experimental validation, and discussion of limitations and future directions.

1.4 Dissertation Organization

The remainder of this dissertation is organized as follows.

Chapter 2 presents the problem statement for the dissertation. It discusses the advantages and limitations of message-passing graph neural networks, identifies the specific problem addressed in this work, and formulates the core research challenge of preserving useful graph topology under constraints on computation, memory, latency, and feature availability. It also introduces the anchor-based structural representation setting, the multiple-view representation problem, and the evaluation goals used to guide the dissertation.

Chapter 3 presents background and preliminaries. It introduces the notation used throughout the dissertation, defines the graph-learning problem settings considered in this work, and reviews distance-based and anchor-based representations. It also summarizes the evaluation protocols and metrics used for node classification, link prediction, and related graph-learning tasks.

Chapter 4 reviews related work. It situates the dissertation within prior research on graph neural networks and message passing, structural and positional encodings, distance- and landmark-based methods, cross-view and cross-graph transfer learning, scalable graph learning, and synthetic graph generation and benchmarking.

Chapter 5 develops anchor-based representations for graph learning. It formalizes the representation framework, discusses anchor selection and sampling strategies, describes the representation-construction pipeline and its computational complexity, and evaluates the resulting methods on node classification tasks.

Chapter 6 studies link prediction using graph-coordinate representations. It defines the task setup, split and negative-sampling protocols, and scalable training procedure. It then reports results

on synthetic and real-world benchmarks and analyzes the conditions under which graph-coordinate methods succeed or degrade.

Chapter 7 investigates transfer across anchor sets, subgraphs, and graph datasets. It studies train-on-one, evaluate-on-many and train-on-many, evaluate-on-one settings, analyzes sources of variation across structural views, presents alignment-based reuse protocols, and distinguishes strong same-graph or overlapping-subgraph transfer evidence from more exploratory cross-dataset transfer results.

Chapter 8 discusses the broader lessons of the dissertation. It summarizes the main findings across chapters, provides practical guidance on when to use the proposed methods, discusses limitations and broader impacts, and outlines promising directions for future work.

Finally, Appendix A provides supporting material, including a justification for orthogonal Procrustes alignment.

Chapter 2

Problem Statement

2.1 Graph Neural Networks: Advantages and Limitations

Graph Neural Networks (GNNs) have become one of the dominant approaches for learning from graph-structured data. Their main advantage is that they combine node or edge attributes with relational information encoded by the graph topology. In message-passing GNNs, each node updates its representation by aggregating information from neighboring nodes, allowing the model to learn features that reflect both local connectivity and node-level attributes. Representative architectures include Graph Convolutional Networks (GCNs), which use localized graph convolution for semi-supervised node classification [2]; GraphSAGE, which samples and aggregates neighborhood features to support inductive representation learning [3]; Graph Attention Networks (GATs), which learn attention weights over neighboring nodes [4]; and the general Message Passing Neural Network (MPNN) framework, which formalizes a broad class of neighborhood aggregation models [5]. Surveys of the field show that GNNs have been applied successfully across social networks, recommendation systems, molecules, biological networks, knowledge graphs, and other relational domains [6, 7].

Despite these advantages, standard message-passing GNNs also have limitations that are central to this dissertation. First, message passing is inherently local: one layer aggregates information from immediate neighbors, and multiple layers are required to incorporate information from distant nodes. Increasing the number of layers can enlarge the receptive field, but it also increases computation, memory use, and training complexity. Second, deeper GNNs can suffer from over-smoothing, where repeated propagation makes node representations increasingly similar and reduces their discriminative power [8]. Third, graph bottlenecks can lead to over-squashing, where information from many distant nodes must be compressed into fixed-size node representations, limiting the effectiveness of long-range dependency modeling [9]. Fourth, large-scale GNN training

often faces neighborhood explosion and high memory requirements, motivating scalable variants such as Cluster-GCN and GraphSAINT [10, 11]. These methods improve scalability, but they still rely on repeated graph access, sampling, and message propagation.

These limitations become more restrictive in practical graph-learning settings. Real-world graphs may be extremely large, noisy, incomplete, weakly attributed, or distributed across devices and systems. In centralized graph databases, repeated multi-hop neighborhood aggregation may be expensive in memory and latency. In distributed settings such as IoT, sensor networks, mobile systems, and edge computing environments, communication cost, power consumption, and limited local storage further constrain the use of deep message-passing pipelines. From the perspective of Green AI, graph learning methods should therefore be evaluated not only by predictive accuracy, but also by parameter count, computational cost, energy use, latency, and deployability [12]. This motivates the search for graph learning methods that preserve useful structural information while reducing dependence on deep message passing and dense all-pairs computation.

2.2 Problem Statement

This dissertation addresses one particular issue within the broader limitations of GNNs: how to preserve useful global graph topology for downstream learning without relying solely on repeated message passing and without constructing dense all-pairs structural matrices. The focus is not to replace all GNNs or to claim universal superiority over message-passing models. Instead, the goal is to develop a resource-adaptable structural representation framework that can complement or partially substitute GNN pipelines when graph topology is informative but node features, computation, memory, latency, or communication resources are limited.

This dissertation investigates how to construct and learn from *structural representations* of large graphs when (i) global topology matters, (ii) node attributes may be missing, incomplete, noisy, or unreliable, and (iii) computational budgets prohibit dense all-pairs computations.

Formally, let $G = (V, E)$ be a graph with $n = |V|$ nodes and $m = |E|$ edges. The graph may be equipped with a node-feature matrix $X \in \mathbb{R}^{n \times F}$, but such features are not assumed to be

available for all graphs. Similarly, labels Y are typically observed only for a subset of nodes or edges, depending on the learning task. The goal is therefore to learn predictors for common graph mining tasks such as:

- **Node classification:** predict labels y_v for unlabeled nodes $v \in V$ using the graph structure G and any available node features X .
- **Link prediction:** predict whether a candidate pair (u, v) forms an edge, or predict its type or weight, using the observed graph structure and any available node or edge information.

These tasks require embeddings or representations that preserve discriminative structural cues at multiple scales, especially when attribute information is limited or unavailable.

The most direct way to represent graph structure is to use the adjacency matrix $A \in \mathbb{R}^{n \times n}$, where each row records the immediate one-hop neighbors of a node. However, adjacency-based representations primarily encode local connectivity; capturing non-local relationships requires repeated propagation, higher-order transformations of A , or an alternative structural representation. A straightforward way to encode global structure is to compute a full pairwise graph-distance matrix $D \in \mathbb{R}^{n \times n}$, such as a shortest-path distance matrix, and use rows of D as node descriptors. However, this approach is not viable for large graphs: storing D is $O(n^2)$ memory, and computing it can be $O(nm)$ or worse depending on the distance construction. Even when approximations exist, they often trade away the very global information that motivates their use.

On the other hand, GNNs and message-passing architectures provide scalable local aggregation mechanisms but can exhibit several failure modes relevant to this work: limited long-range propagation under fixed depth, over-smoothing and over-squashing in deep regimes, and sensitivity to weak or absent node attributes. In practice, these limitations motivate hybrid or alternative approaches that can inject global structural information into learnable models without incurring quadratic cost.

The core problem addressed in this dissertation is how to perform effective graph learning when useful predictive information is embedded in the global topology of a large graph, but the

available data and computational setting make standard approaches insufficient. Message-passing GNNs provide a scalable way to aggregate local neighborhood information, but they may require multiple propagation layers to capture long-range structure, increasing computation, memory use, communication cost, and the risk of over-smoothing. In contrast, full distance-based or spectral representations can encode global graph structure more directly, but dense all-pairs matrices require $O(n^2)$ storage and are infeasible for large practical graphs. This creates a central challenge: how can graph learning preserve discriminative structural information across local and non-local scales while remaining usable when node attributes are incomplete or unreliable and when computation, memory, latency, energy, or communication resources are limited?

This challenge is especially important in practical graph settings where the graph may be stored centrally in a large database or distributed across devices, sensors, edge systems, or organizational data silos. In both cases, the learning method must operate under constraints that standard dense structural methods and deep message-passing pipelines do not always satisfy. Therefore, the problem is characterized by three requirements:

1. **Scalability:** learning should avoid dense $O(n^2)$ storage and computation and remain practical for large graphs.
2. **Attribute robustness:** the method should remain useful when node features X are missing, sparse, noisy, or weakly informative.
3. **Multi-scale structural fidelity:** the representation should preserve structural cues from both local connectivity and non-local/global graph geometry that are relevant to downstream prediction.

2.3 Anchor-Based Structural Representation Setting

The approach studied in this dissertation is based on anchor-based structural representation. A central approximation is to select a set of $k \ll n$ *anchors* $A = \{a_1, \dots, a_k\} \subset V$ and define a structural encoding for each node v through its relationship to the anchors. For a shortest-path

distance function $d(\cdot, \cdot)$, the basic anchor representation of node v is

$$r_A(v) = [d(v, a_1), \dots, d(v, a_k)] \in \mathbb{R}^k.$$

Stacking these vectors over all nodes yields a partial distance matrix $D_A \in \mathbb{R}^{n \times k}$. The dissertation then studies several coordinate constructions derived from this matrix, including direct virtual coordinates (VC), SVD/PCA-based topology coordinates (TC), and directional virtual coordinates (DVC) formed from anchor pairs.

This reduces the representation size from $O(n)$ per node for dense all-pairs representations to at most $O(k)$ per node for anchor-based representations. Let $\mathcal{C} \in \{\text{VC}, \text{TC}, \text{DVC}\}$ denote the coordinate construction applied to the anchor-distance matrix D_A . The key questions are how to choose the anchor set A , how many anchors to use, and which coordinate construction $\mathcal{C}$ to apply so that the resulting embeddings preserve predictive structural signals while remaining computationally feasible.

2.4 Multiple-View Structural Representation Problem

Different anchor sets can produce different views of the same underlying graph, and in some datasets multiple graph layers over a common node set provide additional structural views. This leads to a second problem studied in this dissertation:

Given multiple anchor-induced views of the same graph, and in some cases multiple graph layers associated with the same node set, determine when the resulting structural representations remain stable, when they provide complementary information, and how they can be used effectively for downstream prediction.

This includes questions of view-to-view stability, reuse across anchor sets, and lightweight alignment across related views, rather than the design of a general multi-channel fusion framework.

2.5 Evaluation Problem

Finally, the dissertation addresses the methodological question of how to evaluate such systems reliably. Because graph topology, feature availability, and label structure vary widely across datasets, evaluation must cover both (i) controlled synthetic graphs that isolate structural factors and (ii) realistic benchmarks that stress scalability and generalization. The problem is therefore not only to propose representations and models, but to demonstrate when and why they work across graph families and task regimes. In addition to predictive performance, evaluation should consider resource-relevant quantities such as model size, trainable parameter count, memory footprint, computation time, and deployment feasibility.

2.6 Summary

In short, this dissertation develops and evaluates methods for approximating and learning from global graph structure using anchor-based structural representations. The work is motivated by the practical limitations of message-passing GNNs, the infeasibility of dense all-pairs structural matrices, and the need for Green-AI-oriented graph learning systems that balance accuracy with computation, memory, latency, energy, and deployability. The proposed direction enables effective node and link prediction by using compact anchor-induced structural encodings, including multiple anchor-induced views and, where appropriate, multiple graph layers, without incurring quadratic computational cost.

Chapter 3

Background and Preliminaries

This chapter introduces the notation, graph-learning problem settings, structural representations, and evaluation protocols used throughout the dissertation. It first defines the basic graph, feature, label, distance, anchor, and representation notation. It then reviews the main learning tasks considered in this work, including node classification and link prediction, and summarizes the evaluation metrics and experimental protocols used in later chapters. Unless stated otherwise, all graphs are assumed to be finite.

3.1 Notation and Definitions

3.1.1 Sets, Indices, and Basic Conventions

We use calligraphic letters (e.g., $\mathcal{V}, \mathcal{E}$) to denote sets, bold lowercase letters (e.g., $\mathbf{x}$) to denote vectors, and bold uppercase letters (e.g., $\mathbf{A}$) to denote matrices. For an integer n , we write $[n] := \{1, 2, \dots, n\}$. For a set $\mathcal{S}$, $|\mathcal{S}|$ denotes its cardinality.

Given a matrix $\mathbf{M} \in \mathbb{R}^{n \times m}$, $\mathbf{M}_{i:}$ denotes its i -th row and $\mathbf{M}_{:,j}$ denotes its j -th column. The identity matrix of size n is $\mathbf{I}_n$. For vectors, $\|\cdot\|_2$ denotes the Euclidean norm.

3.1.2 Graphs

A (simple) graph is denoted by

$$\mathcal{G} = (\mathcal{V}, \mathcal{E}), \tag{3.1}$$

where $\mathcal{V}$ is the node set with $|\mathcal{V}| = N$, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the edge set. We use $u, v \in \mathcal{V}$ to index nodes and $(u, v) \in \mathcal{E}$ to denote an edge. When $\mathcal{G}$ is undirected, $(u, v) \in \mathcal{E}$ implies $(v, u) \in \mathcal{E}$, and we sometimes write $\{u, v\}$.

The (binary) adjacency matrix is $\mathbf{A} \in \{0, 1\}^{N \times N}$, where $\mathbf{A}_{uv} = 1$ iff $(u, v) \in \mathcal{E}$. The neighborhood of node u is

$$\mathcal{N}(u) := \{v \in \mathcal{V} : (u, v) \in \mathcal{E}\}, \quad (3.2)$$

and the degree is $d_u := |\mathcal{N}(u)|$. The diagonal degree matrix is $\mathbf{D} \in \mathbb{R}^{N \times N}$ with $\mathbf{D}_{uu} = d_u$.

3.1.3 Node Features, Labels, and Splits

If node attributes are available, we denote the node feature matrix by $\mathbf{X} \in \mathbb{R}^{N \times F}$, where $\mathbf{x}_u := \mathbf{X}_{u:} \in \mathbb{R}^F$ is the feature vector of node u . For node classification with C classes, we denote labels by $\mathbf{y} \in [C]^N$ or equivalently one-hot vectors $\mathbf{Y} \in \{0, 1\}^{N \times C}$.

We use disjoint index sets $\mathcal{V}_{\text{train}}, \mathcal{V}_{\text{val}}, \mathcal{V}_{\text{test}} \subseteq \mathcal{V}$ for train/validation/test node splits (node classification), and $\mathcal{E}_{\text{train}}, \mathcal{E}_{\text{val}}, \mathcal{E}_{\text{test}} \subseteq \mathcal{E}$ for edge splits (link prediction). When negative edges are required, we denote a sampled set of non-edges by $\mathcal{E}^- \subseteq (\mathcal{V} \times \mathcal{V}) \setminus \mathcal{E}$.

3.1.4 Distances, Anchors, and Distance Matrices

A recurring object in this dissertation is a node-to-node or node-to-anchor distance. Let $d : \mathcal{V} \times \mathcal{V} \rightarrow \mathbb{R}_{\geq 0}$ be a distance or dissimilarity measure on nodes. Common examples include shortest-path distance, diffusion-based distances, or distances induced by kernels/embeddings.

Given an *anchor set* $\mathcal{A} \subseteq \mathcal{V}$ with $|\mathcal{A}| = M$, we define the node-to-anchor distance matrix

$$\mathbf{D}^{(\mathcal{A})} \in \mathbb{R}^{N \times M}, \quad \mathbf{D}_{u,a}^{(\mathcal{A})} = d(u, a), \quad a \in \mathcal{A}. \quad (3.3)$$

When $\mathcal{A} = \mathcal{V}$, we write $\mathbf{D} \in \mathbb{R}^{N \times N}$ for the full pairwise distance matrix. In some settings, we consider multiple structurally related distance constructions indexed by $k \in [K]$, for example when comparing multiple anchor-induced views of the same graph or multiple graph layers defined on a common node set:

$$\{\mathbf{D}^{(k)}\}_{k=1}^K, \quad \mathbf{D}^{(k)} \in \mathbb{R}^{N \times M_k}. \quad (3.4)$$

To reduce notational clutter, we omit $(\mathcal{A})$ or (k) when the context is clear.

3.1.5 Dimensionality Reduction and Representations

We denote a node representation (embedding) matrix by $\mathbf{Z} \in \mathbb{R}^{N \times d}$, where $\mathbf{z}_u \in \mathbb{R}^d$ is the representation of node u . When representations are derived from a preprocessing step such as PCA, we use

$$\mathbf{Z}^{(k)} = \text{PCA}(\mathbf{M}^{(k)}; d), \quad (3.5)$$

where $\mathbf{M}^{(k)}$ may be a distance-derived matrix, a kernel matrix, or another graph-derived signal in general; in the methods developed in this dissertation, $\mathbf{M}^{(k)}$ is primarily a node-to-anchor distance matrix or a coordinate representation derived from it. We use d for the retained dimension (not to be confused with the distance function $d(\cdot, \cdot)$; the meaning is disambiguated by context).

3.1.6 Predictive Models and Loss Functions

A parametric predictor is denoted by $f_\theta(\cdot)$ with parameters θ . For node classification, the model outputs logits $\mathbf{H} \in \mathbb{R}^{N \times C}$ and class probabilities

$$p_\theta(y = c \mid u) = \text{softmax}(\mathbf{H}_{u:})_c. \quad (3.6)$$

We use the cross-entropy loss over training nodes,

$$\mathcal{L}_{\text{CE}}(\theta) = - \sum_{u \in \mathcal{V}_{\text{train}}} \log p_\theta(y = y_u \mid u). \quad (3.7)$$

For link prediction, a scoring function assigns a real-valued score to a candidate edge:

$$s_\theta(u, v) \in \mathbb{R}. \quad (3.8)$$

Scores may be converted to probabilities via a sigmoid, and trained using a binary cross-entropy objective over positive edges and sampled negatives:

$$\mathcal{L}_{\text{BCE}}(\theta) = - \sum_{(u,v) \in \mathcal{E}_{\text{train}}} \log \sigma(s_{\theta}(u, v)) - \sum_{(u,v) \in \mathcal{E}^-} \log (1 - \sigma(s_{\theta}(u, v))). \quad (3.9)$$

3.1.7 Evaluation Metrics

For node classification, we primarily report accuracy and/or macro/micro F1 scores on $\mathcal{V}_{\text{val}}$ and $\mathcal{V}_{\text{test}}$. For link prediction, common metrics include ROC-AUC and average precision (AP), as well as ranking-based metrics such as Hits@ K depending on the experimental protocol.

3.1.8 Subgraphs, Partitions, and Connected Components

When working with large graphs, we may partition $\mathcal{V}$ into P parts $\{\mathcal{V}_p\}_{p=1}^P$, inducing subgraphs

$$\mathcal{G}_p = (\mathcal{V}_p, \mathcal{E}_p), \quad \mathcal{E}_p := \{(u, v) \in \mathcal{E} : u, v \in \mathcal{V}_p\}. \quad (3.10)$$

For a subgraph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$, we denote its connected components by $\{\mathcal{C}_j\}_j$ and write $\text{LCC}(\mathcal{G}')$ for its largest connected component.

The above notation is intended to be modular: later chapters primarily instantiate these definitions using node-to-anchor graph distances, coordinate transformations derived from those distances, anchor selection strategies, and learning architectures.

3.2 Problem Settings in Graph Learning

This section formalizes the main learning problems considered in this dissertation and clarifies the differences between commonly used experimental protocols. We focus on *transductive* learning on a single observed graph, which covers most benchmark settings in graph representation learning, while also noting the *inductive* variants when relevant.

These problem settings are standard in graph representation learning rather than being specific to this dissertation. Surveys and benchmark suites commonly organize graph learning around node-level prediction, link prediction, and graph-level prediction, and the Open Graph Benchmark (OGB) provides large-scale datasets and protocols for node-property, link-property, and graph-property prediction across multiple application domains [7, 13]. Link prediction is especially important because it formalizes the inference of missing or future relations in network-structured data [14]. The anchor-based representations used in this dissertation are also motivated by a separate body of work on virtual and topology coordinates in sensor and IoT networks, where hop-distance coordinates relative to anchors have supported localization-free routing, topology-preserving map construction, boundary detection, and node-level network awareness [15–19]. This prior use of anchor-coordinate systems in distributed and resource-constrained networks makes them a natural candidate for graph-learning settings in which topology is informative but full graph access, node features, or message-passing computation may be limited.

3.2.1 Data Modalities: Attributed vs. Non-Attributed Graphs

We distinguish between two common input regimes:

- **Attributed graphs:** nodes have features $\mathbf{X} \in \mathbb{R}^{N \times F}$ available at training and test time.
- **Structure-only graphs:** node features are missing or uninformative, i.e., $\mathbf{X}$ is absent or replaced by a constant/identity encoding. In this regime, useful signals must be extracted primarily from $\mathcal{G}$ (e.g., distances, positional encodings, or learned structural embeddings).

The methods studied in this dissertation operate in both regimes by treating graph-derived distance-to-anchor vectors and their coordinate transformations as alternative feature matrices.

Figure 3.1 illustrates the node classification and link prediction settings considered in this work.

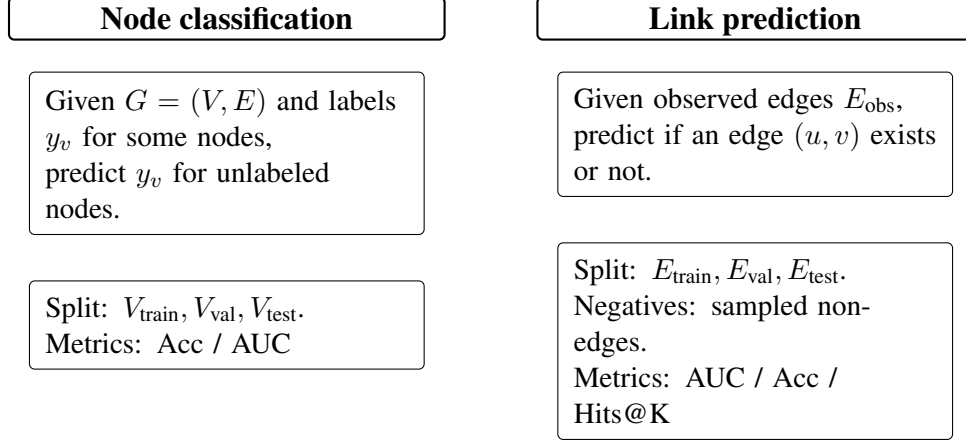


Figure 3.1: Problem settings: node classification and link prediction.

3.2.2 Node Classification

In node classification, each node $u \in \mathcal{V}$ is associated with a label $y_u \in [C]$ (for C classes). Given $\mathcal{G}$ and optional features $\mathbf{X}$, the goal is to learn a predictor

$$f_\theta : \mathcal{V} \rightarrow [C] \quad (3.11)$$

from labeled nodes in $\mathcal{V}_{\text{train}}$, and evaluate on $\mathcal{V}_{\text{val}}$ and $\mathcal{V}_{\text{test}}$. In practice, f_θ is implemented as a differentiable model producing logits $\mathbf{H}_u \in \mathbb{R}^C$ and trained by minimizing $\mathcal{L}_{\text{CE}}$ (Section 3.1).

In our experiments, the full graph $\mathcal{G}$ is observed during training, including test nodes and their connectivity, but labels are only revealed for $\mathcal{V}_{\text{train}}$.

3.2.3 Link Prediction

Link prediction aims to infer missing or future edges. Given $\mathcal{G}$ and a training edge set $\mathcal{E}_{\text{train}}$, we learn a scoring function

$$s_\theta : \mathcal{V} \times \mathcal{V} \rightarrow \mathbb{R} \quad (3.12)$$

that assigns higher scores to true edges than to non-edges. Evaluation is performed on held-out positives $(u, v) \in \mathcal{E}_{\text{val}}$ and $(u, v) \in \mathcal{E}_{\text{test}}$ along with sampled negatives $\mathcal{E}^-$. Depending on protocol, metrics include ROC-AUC, AP, and Hits@ K .

Edge splitting protocols and leakage considerations. When splitting edges into train/ validation/test sets, one must specify whether the training graph used for message passing includes only $\mathcal{E}_{\text{train}}$ or also includes additional edges (e.g., to maintain connectivity). In this dissertation, unless explicitly stated, we assume the *training graph* uses only $\mathcal{E}_{\text{train}}$ to avoid information leakage from validation/test edges into learned representations.

3.2.4 Graph-Level Prediction

In graph classification/regression, each graph $\mathcal{G}_i$ is associated with a label y_i (or target $t_i \in \mathbb{R}$). The input is a dataset of graphs $\{\mathcal{G}_i\}_{i=1}^n$ with optional node/edge features. The goal is to learn

$$g_\theta : \{\text{graphs}\} \rightarrow \mathcal{Y} \tag{3.13}$$

where $\mathcal{Y}$ is a discrete label space (classification) or $\mathbb{R}$ (regression). This problem is typically inductive because test graphs are disjoint from training graphs. While not the primary focus of this dissertation, many components, especially positional encodings and anchor-based structural summaries, carry over directly.

Graph-level prediction appears in many application domains where the object of interest is an entire relational structure rather than a single node or edge. Examples include molecular property prediction, where atoms and bonds define a molecular graph; protein or biological-network classification, where the label describes a whole biological structure or interaction network; and program, social, or communication-network analysis, where the graph label may describe function, category, or system behavior [13,20,21]. Although this dissertation focuses primarily on node classification and link prediction, graph-level prediction is relevant because anchor-based structural summaries can be pooled or aggregated over nodes to form graph-level descriptors. For example, statistics of node-to-anchor distances, low-dimensional topology coordinates, or distributions of structural coordinates could be used as compact graph-level features.

3.2.5 Self-Supervised and Pretext Objectives

When labels are scarce or absent, representation learning can be framed using self-supervised objectives defined on $\mathcal{G}$. Common choices include:

- **Contrastive objectives:** align representations under graph augmentations while repelling mismatched pairs.
- **Reconstruction objectives:** predict masked node attributes, edges, or structural signals.
- **Distillation/consistency objectives:** enforce agreement between multiple views derived from the same graph, such as different anchor sets or related graph layers.

In later chapters, these ideas motivate comparing and reusing graph-derived structural views, especially across different anchor sets on the same graph and, in some datasets, across multiple graph layers associated with the same node set.

3.2.6 Scalability and Computational Constraints

Large-scale graph learning often requires approximations due to memory and runtime limits. We highlight two recurring constraints:

- **Neighborhood explosion:** message passing over L layers can expand receptive fields exponentially, stressing memory and computation.
- **Global signals:** distances, kernels, and spectral objects can be dense ($N \times N$), motivating anchor-based approximations $\mathbf{D}^{(\mathcal{A})} \in \mathbb{R}^{N \times M}$ with $M \ll N$, sampling strategies, and partitioned computation.

The use of anchor-derived coordinates is not limited to graph machine learning. In wireless sensor and IoT-style networks, virtual-coordinate systems based on hop distances to anchors have been used for localization-free routing, topology-preserving map construction, boundary detection, and node-level network awareness [15–19, 22, 23]. These applications provide practical evidence that compact anchor-based coordinates can support distributed decision-making when physical

locations, full global topology, or repeated network-wide computation are unavailable. This makes them a natural representation candidate for scalable graph-learning pipelines that seek to separate structural feature construction from downstream prediction.

These constraints directly motivate the anchor-based distance matrices, multiple-view structural representations, and partition-aware pipelines used throughout the dissertation.

3.2.7 Message-Passing GNNs and Embedding-Based Neural Networks

A useful distinction in this dissertation is between message-passing graph neural networks and graph-embedding-based neural networks. In a message-passing GNN, node representations are updated through repeated aggregation over graph neighborhoods, so the graph structure is used directly inside the neural architecture. In contrast, a graph-embedding-based neural network first constructs a graph-derived representation, such as a distance-to-anchor vector or a coordinate embedding, and then uses that representation as input to a conventional neural model such as a linear classifier or MLP.

The methods developed in this dissertation follow the second pattern. Graph topology is encoded before or outside the downstream predictor through virtual coordinates (VC), topology coordinates (TC), or directional virtual coordinates (DVC). The resulting models, such as VCNN, TCNN, and DVCNN, therefore differ from standard message-passing GNNs: they use graph structure to construct features, but they do not perform iterative neighborhood aggregation during the downstream neural prediction step.

3.2.8 Summary

Across these problem settings, the common theme is to learn predictors that exploit graph topology and (optional) attributes while remaining robust to data sparsity and scalable to large graphs. The notation in Section 3.1 enables a unified view: whether the input is $\mathbf{X}$, a distance-to-anchor matrix $\mathbf{D}^{(A)}$, or a coordinate representation derived from such a matrix, the downstream tasks reduce to learning f_θ (node-level) or s_θ (pairwise) under clearly defined train/validation/test protocols.

3.3 Distance-Based and Anchor-Based Representations

Many graph learning pipelines rely on message passing to propagate information locally along edges. In contrast, *distance-based* and *anchor-based* representations provide a complementary view: each node is described by its relationship to other nodes (globally or semi-globally) via distances, similarities, or kernels. This section introduces these representations and motivates their use as scalable, topology-aware features.

3.3.1 Full Pairwise Distance Representations

Let $d : \mathcal{V} \times \mathcal{V} \rightarrow \mathbb{R}_{\geq 0}$ be a graph-derived distance or dissimilarity (Section 3.1). The most direct distance-based representation of node u is the vector of distances to all nodes:

$$\mathbf{r}_u := (d(u, v))_{v \in \mathcal{V}} \in \mathbb{R}^N, \quad (3.14)$$

or equivalently the u -th row of the full distance matrix $\mathbf{D} \in \mathbb{R}^{N \times N}$ with $\mathbf{D}_{uv} = d(u, v)$. This representation is maximally informative with respect to the chosen distance, but it is typically impractical for large graphs due to $O(N^2)$ storage and computation.

Distances can be instantiated in multiple ways depending on the structural signal one wants to emphasize:

- **Shortest-path distance:** $d_{\text{sp}}(u, v)$ captures discrete hop-based proximity.
- **Diffusion-based distances:** based on random walks or heat diffusion, capturing multi-scale connectivity beyond shortest paths.
- **Kernel-induced distances:** e.g., from a similarity/kernel matrix $\mathbf{K}$ using standard transformations (when applicable).

Later chapters primarily instantiate $d(\cdot, \cdot)$ as shortest-path distance and then construct VC, TC, and DVC representations from the resulting node-to-anchor distance matrices.

3.3.2 Anchor-Based Approximations

To reduce complexity, anchor-based representations select a subset of nodes $\mathcal{A} \subseteq \mathcal{V}$ with $|\mathcal{A}| = M \ll N$ and represent each node by its distances to anchors:

$$\mathbf{r}_u^{(\mathcal{A})} := (d(u, a))_{a \in \mathcal{A}} \in \mathbb{R}^M, \quad \mathbf{D}^{(\mathcal{A})} \in \mathbb{R}^{N \times M}. \quad (3.15)$$

This reduces storage from $O(N^2)$ to $O(NM)$ and often enables efficient mini-batch computation by focusing on a small set of landmark relationships.

Interpretation. Anchors provide a coordinate system over the graph: each node is located by its graph-distance proximity to a set of reference nodes. When anchors are well-chosen, $\mathbf{r}_u^{(\mathcal{A})}$ preserves coarse global position and community structure while remaining computationally manageable. It is well known that the distance matrix of a practical graph is low rank, whereas the adjacency matrix, which is the basis of message passing in a GNN is high rank [1, 24]. Therefore, a fraction of columns of $\mathbf{D}$ can capture the entire topology information [16]. Also note that notions such as metric dimension and link dimension of a graph and the corresponding anchor sets, the resolution set and the resolving set respectively, that allow the characterization of nodes and edges uniquely, and thus able to capture aspects of graph topology, are directly related to VCs.

3.3.3 Anchor Selection Strategies

Anchor selection affects both representation quality and computational properties. Common strategies include:

- **Uniform random sampling:** simple and scalable; often surprisingly effective on large graphs.
- **Degree-based sampling:** prefer high-degree nodes to improve reachability and reduce disconnectedness effects.

- **Coverage-based sampling:** select anchors to cover the graph (e.g., via farthest-point sampling or clustering), improving geometric diversity.
- **Partition-aware anchors:** select anchors within partitions and/or include shared boundary anchors to stabilize representations under graph partitioning.
- **Algorithmic or topology-aware anchor search:** use graph-specific procedures to identify structurally informative anchors rather than selecting them only by random, degree, or simple coverage rules. For example, Extreme Node Search (ENS) uses Directional Virtual Coordinate (DVC) information to identify anchor candidates in a distributed manner and was shown to improve anchor placement for virtual-coordinate routing and topology-preserving maps in wireless sensor networks [25].

In this dissertation, anchors are primarily used as a scalable surrogate for global signals; thus, simplicity and reproducibility of the selection procedure are emphasized unless a task demands stronger coverage guarantees.

3.3.4 From Distances to Features

The raw distance-to-anchor matrix $\mathbf{D}^{(\mathcal{A})}$ can be used directly as features, but it is also common to apply transformations:

- **Normalization:** column-wise standardization or scaling to reduce sensitivity to graph diameter and anchor placement.
- **Nonlinear transforms:** e.g., $d \mapsto \exp(-d/\tau)$ to convert distances to similarities with a temperature $\tau > 0$.
- **Centering:** subtract per-node or per-anchor means to remove global offsets.

We denote a generic feature construction by

$$\mathbf{X}_{\text{dist}} := \psi(\mathbf{D}^{(\mathcal{A})}) \in \mathbb{R}^{N \times M}, \quad (3.16)$$

where $\psi(\cdot)$ is a (possibly learned) transformation.

3.3.5 Multiple Structural Views from Anchors and Graph Layers

In some settings, more than one structural view is available. In this dissertation, these multiple views arise primarily from different anchor sets on the same graph and, for datasets such as OGBN-Proteins, from multiple graph layers defined over a common node set. We denote the resulting family of matrices by $\{\mathbf{D}^{(k)}\}_{k=1}^K$ (Section 3.1). Each view yields a representation

$$\mathbf{r}_u^{(k)} \in \mathbb{R}^{M_k}, \quad (3.17)$$

and downstream analyses may compare them, concatenate them, or evaluate their stability across views. This viewpoint is useful when different anchor choices or graph layers expose complementary aspects of the same underlying topology.

3.3.6 Practical Considerations: Complexity and Robustness

Computation. For shortest-path-based anchors, computing $\mathbf{D}^{(A)}$ can be done via M single-source shortest path computations (e.g., BFS on unweighted graphs), often feasible when M is small. For diffusion-based distances, which are discussed here as background rather than used as the main experimental construction, approximate methods (truncated walks, sparse linear solves, or low-rank approximations) are commonly used.

Disconnected graphs. If $\mathcal{G}$ has multiple components, distances between nodes in different components may be infinite. Practical implementations typically handle this by working on the largest connected component, adding a large constant for unreachable pairs, or defining a diffusion distance that remains finite. In our work, we mainly work on the largest connected component, and for the few nodes outside the largest connected component, we assign 0 for their graph coordinates.

Stability. Anchor-based representations trade information for efficiency. Empirically, increasing M improves fidelity but also increases compute and memory. Later chapters examine this trade-

off and propose strategies such as sampling, partitioning, and alignment across views to improve stability at fixed budgets.

3.3.7 Summary

Distance-based representations encode global graph structure directly but are often quadratic in N . Anchor-based representations replace all-pairs distances by node-to-anchor distances, providing a scalable coordinate system over the graph. This abstraction underpins the anchor-distance matrices, multiple-view structural representations, and alignment-based analyses used throughout the dissertation.

3.4 Evaluation Protocols and Metrics

Reliable empirical conclusions in graph learning depend critically on (i) how data are split into training/validation/test sets, (ii) what information is available to the model at training time, and (iii) which metrics are reported. This section summarizes the evaluation protocols used throughout the dissertation and the metrics used for node classification and link prediction. Several experiments in later chapters use standard benchmark splits and metrics from the Open Graph Benchmark (OGB), with dataset-specific details deferred to the corresponding experimental sections.

3.4.1 General Principles and Reproducibility

We adopt the following general practices unless explicitly stated otherwise:

- **Fixed splits when provided:** if a dataset defines standard splits, we use them to ensure comparability with prior work.
- **Validation-based model selection:** hyperparameters and early stopping are selected using $\mathcal{V}_{\text{val}}$ (node tasks) or $\mathcal{E}_{\text{val}}$ (edge tasks), and final results are reported on the held-out test set.
- **Multiple runs:** Unless otherwise stated, stochastic experiments (random initialization, sampling, anchor selection) are reported as mean $\pm$ standard deviation over multiple runs; some

failure-case or diagnostic experiments are reported as single runs when repetition is not informative.

- **No test leakage:** all preprocessing that could incorporate supervision (e.g., label-dependent normalization) is restricted to training data.

3.4.2 Transductive Protocols and Information Access

Many benchmark graph learning settings are transductive: the full node set $\mathcal{V}$ is known in advance, and the full adjacency structure may be observed, but supervision is only available on training indices.

Node classification. In transductive node classification, the graph $\mathcal{G}$ is observed during training. The model may use adjacency information involving test nodes, but must not access their labels. Formally, training minimizes $\mathcal{L}_{\text{CE}}$ over $u \in \mathcal{V}_{\text{train}}$ while representations may depend on all of $\mathcal{G}$.

Link prediction. In link prediction, the protocol must specify the *training graph* used to compute representations. Unless stated otherwise, we use a conservative protocol: representations are computed using only $\mathcal{E}_{\text{train}}$, and edges in $\mathcal{E}_{\text{val}}$ and $\mathcal{E}_{\text{test}}$ are treated as unobserved during training. This avoids leakage where validation/test edges could implicitly inform node embeddings.

3.4.3 Splitting Strategies

Different tasks induce different split objects.

Node splits. For node classification, we use disjoint node subsets $\mathcal{V}_{\text{train}}, \mathcal{V}_{\text{val}}, \mathcal{V}_{\text{test}}$. If multiple split realizations are used, we report averaged results to reduce sensitivity to a particular partition of labels.

Edge splits and negative sampling. For link prediction, we use disjoint edge subsets $\mathcal{E}_{\text{train}}, \mathcal{E}_{\text{val}}, \mathcal{E}_{\text{test}}$. Evaluation requires negative examples (non-edges) drawn from $(\mathcal{V} \times \mathcal{V}) \setminus \mathcal{E}$. We denote by $\mathcal{E}_{\text{val}}^-$

and $\mathcal{E}_{\text{test}}^-$ the negative sets used during evaluation; these may be resampled per run or fixed by the dataset protocol.

3.4.4 Node Classification Metrics

Let $\hat{y}_u$ be the predicted class for node u .

Accuracy.

$$\text{Acc} := \frac{1}{|\mathcal{V}_{\text{test}}|} \sum_{u \in \mathcal{V}_{\text{test}}} \mathbb{I}[\hat{y}_u = y_u]. \quad (3.18)$$

F1 scores. When class imbalance is present, we additionally report macro-F1 and/or micro-F1.

Let F1_c denote the F1 score for class c ; then

$$\text{F1}_{\text{macro}} := \frac{1}{C} \sum_{c=1}^C \text{F1}_c, \quad (3.19)$$

while micro-F1 aggregates counts across classes (equivalently, global precision/recall before computing F1).

3.4.5 Link Prediction Metrics

Let $s_\theta(u, v)$ be the score assigned to candidate edge (u, v) .

ROC-AUC. ROC-AUC measures the probability that a randomly chosen positive edge is scored higher than a randomly chosen negative edge. It is threshold-free and commonly used when negative sampling is balanced.

Average precision (AP). AP summarizes the precision–recall curve and is often more informative than ROC-AUC under class imbalance, which is typical in link prediction (many more non-edges than edges).

Hits@ K and ranking metrics. In ranking-based protocols, each positive query edge is evaluated against a set of candidate negatives; Hits@ K is the fraction of cases where the positive appears in the top- K by score. Variants include MRR (mean reciprocal rank). These metrics depend strongly on the candidate set definition and are therefore reported together with the negative sampling protocol.

3.4.6 Early Stopping and Model Selection

For node classification, early stopping is typically based on validation accuracy or validation loss; for link prediction, on validation ROC-AUC/AP (or Hits@ K). We select hyperparameters based only on validation performance, and the test set is used exactly once for reporting the final selected model.

3.4.7 Reporting Format

Unless stated otherwise, we report:

- mean $\pm$ standard deviation over multiple runs,
- the best validation checkpoint per run (early-stopped),
- the corresponding test performance for that checkpoint.

When computational budgets are constrained (e.g., large graphs, many anchors, or multiple views /layers), we additionally report runtime- or memory-relevant quantities (e.g., number of anchors M and, when applicable, the number of views or graph layers K) to make trade-offs explicit.

3.4.8 Summary

Evaluation in graph learning is sensitive to split construction, information access, and the definition of negative examples. Throughout this dissertation, we use standard splits when available, avoid validation/test leakage in representation construction, and report task-appropriate metrics (accuracy/F1 for node classification; ROC-AUC/AP/Hits@ K for link prediction) with transparent protocols.

Chapter 4

Related Work

This chapter reviews prior work related to scalable and structural graph learning. It situates the dissertation within research on message-passing GNNs, structural and positional encodings, distance- and landmark-based representations, transfer across graph views and domains, scalable graph learning, and synthetic graph benchmarking. The goal is to clarify how the proposed anchor-based framework differs from, complements, and extends existing approaches.

4.1 Graph Neural Networks and Message Passing

Recent work has continued to refine the message passing view of graph neural networks rather than abandoning it. The position paper by [26] argues that many methods described as going “beyond message passing” can still be interpreted as pairwise message passing carried out on a modified graph or with augmented structure. This perspective is useful because it frames message passing not as a narrow architectural choice, but as a general computational primitive whose limitations often motivate changes in graph construction, aggregation, or update mechanisms rather than a complete departure from the paradigm.

A recent example of strengthening message passing directly is given by [27]. They propose Scalable Message Passing Neural Networks (SMPNNs), which replace the attention operator in a Transformer-style block with standard convolutional message passing while retaining pre-layer normalization and deep residual design. Their results suggest that carefully engineered message passing architectures can remain competitive with, and in some settings outperform, graph transformers on large transductive learning problems, while avoiding the computational and memory costs of full attention.

Scalability has also motivated new formulations of mini-batch message passing. The work by [28] introduces Topological Compensation (TOP), an approach for large-graph transductive learning based on the notion of message invariance. Their method reformulates the contribution

of out-of-batch neighbors so that whole-graph message passing can be approximated through efficient in-batch computations, thereby addressing a core bottleneck of sampled mini-batch training while preserving the effect of the original propagation process more faithfully than naive subgraph restriction.

Another recent direction revisits the assumption that every node should use the same message passing depth. [29] propose ADMP-GNN, a framework that dynamically adjusts the number of message passing layers on a per-node basis. Their empirical analysis shows that the optimal propagation depth can vary across nodes with different structural characteristics, and their adaptive mechanism is designed to exploit this heterogeneity. This line of work is especially relevant for graphs where shallow propagation may underreach some regions while deeper propagation may oversmooth others.

In parallel with these architectural developments, [30] provide a recent survey focused specifically on the generalization theory of graph neural networks. Their review emphasizes that, although message-passing neural networks have been studied extensively from the perspective of expressivity, less attention has been given to when and why they generalize beyond the training graph distribution. By organizing recent theoretical work on stability, transferability, and statistical generalization, this survey helps place modern message passing models within a broader theoretical landscape and highlights open questions that remain relevant for both practical and methodological research.

[6] provide a broad overview of graph neural networks and help situate message passing within the larger graph learning literature. Their survey organizes GNN models into recurrent graph neural networks, convolutional graph neural networks, graph autoencoders, and spatial-temporal graph neural networks, and also reviews benchmark datasets, open-source implementations, and application domains. This paper is useful in the related work chapter because it frames message passing not as an isolated technique, but as part of a broader family of graph-based neural architectures that address different forms of relational data and downstream tasks.

The work by [31] extends the role of message passing beyond node-level representation learning by integrating it with graph pooling. They revisit spectral clustering as a way to group strongly connected nodes, but instead of relying on graph-specific Laplacian eigendecomposition for every input graph, they formulate a differentiable relaxation of the normalized minCUT objective and train a GNN to learn cluster assignments directly. This produces a pooling operator that is both end-to-end trainable and applicable to out-of-sample graphs, showing how message passing can be combined with hierarchical coarsening mechanisms to improve graph-level representation learning.

An example of message passing in scientific applications is given by [32]. They model polycrystalline microstructures as graphs in which grains are nodes and neighboring grain interactions are explicitly represented through graph connectivity. Their GNN combines grain-level physical features with neighborhood interactions to predict material properties, and they report that the learned model is both accurate and interpretable, including feature-importance analysis at the grain level. This paper is useful in the present section because it illustrates how message passing naturally captures local physical interactions that conventional feature-only models may ignore.

The limitations of message passing have been studied from several complementary perspectives, including WL-bounded expressivity, heterophily, neighborhood explosion, scalable sampling, and decoupled propagation [10, 11, 33–38]. These results motivate representations that expose structural positions or long-range topology without relying solely on deeper local aggregation.

4.2 Structural and Positional Encodings

A central motivation for structural and positional encodings is that standard message-passing graph neural networks are permutation invariant and therefore do not provide nodes with a canonical notion of position in the graph. The graph transformer paper [39] argues that this missing positional information is one of the main reasons why vanilla Transformers underperform on graphs, and it proposes Graphormer as a Transformer architecture equipped with explicit structural encodings. In particular, the model incorporates graph-aware biases so that shortest-path structure, edge

information, and node centrality are injected directly into attention, showing that careful structural encoding can make Transformer-style models competitive on graph representation tasks.

The paper [40] develops the LSPE framework, which explicitly separates structural features from positional features and lets the model learn both representations jointly. Their motivation is that arbitrary graphs do not provide a natural positional system analogous to word order in sequences, so purely structural message passing can fail to distinguish symmetric or structurally similar nodes. By decoupling these two channels and allowing positional encodings to be learned rather than only injected as fixed features, LSPE provides a flexible framework that can be applied to both sparse message-passing GNNs and Transformer-like graph models.

A broader organizing view is provided by [41], which introduces GraphGPS and gives a unified recipe for graph transformers. This work is especially useful for a related-work chapter because it distinguishes positional and structural encodings into local, global, and relative categories, and argues that a strong graph transformer can be built from three ingredients: a positional or structural encoding, a local message-passing mechanism, and a global attention mechanism. Rather than treating encodings as a small implementation detail, this paper places them at the center of graph-transformer design and shows that the choice of encoding strongly affects both expressivity and scalability.

Recent work has also highlighted that not all positional encodings are equally stable. The paper [42] studies Laplacian-eigenvector-based positional encodings and points out two fundamental issues: non-uniqueness, since a graph can admit multiple equivalent eigendecompositions, and instability, since small perturbations to the graph can induce large changes in the eigenspace. To address this, the authors propose Stable and Expressive Positional Encodings (SPE), which use eigenvalues to softly partition eigenspaces and provide a basis-invariant and provably stable encoding mechanism. This is highly relevant when discussing structural and positional encodings because it shows that the usefulness of a positional encoding depends not only on expressivity but also on robustness across graphs.

The paper [43] focuses on making positional encodings both expressive and computationally efficient. The authors argue that graph positional encodings should ideally satisfy four properties—stability, expressive power, scalability, and genericness—and they observe that many eigenvector-based methods do not satisfy all four at once. They introduce PEARL, a learnable positional encoding framework in which message-passing GNNs are used as nonlinear mappings of eigenvectors, yielding encodings with linear complexity while maintaining strong expressive power. This paper is a useful recent addition because it shifts the discussion from whether positional encodings help to how they can be designed so that they remain practical on larger graphs.

Another extension of the Laplacian-based view is given by [44], which generalizes classical Laplacian positional encoding beyond the standard Euclidean formulation. Instead of restricting the construction to the usual quadratic objective underlying Laplacian eigenmaps, the paper defines a family of generalized positional encodings based on broader dissimilarity functions, including p -norm variants. This line of work is useful in the present section because it shows that structural and positional encodings are not limited to one fixed spectral recipe; rather, they form a broader design space in which different notions of graph geometry can be encoded to enhance the expressive power of message-passing models.

Structural and positional encodings have become a common mechanism for injecting graph position, relative distance, centrality, or stable spectral information into GNNs and graph transformers [39, 44–48]. This line of work is closely related to the present dissertation because it treats graph structure as an explicit representation signal rather than only as an implicit message-passing substrate.

4.3 Distance- and Landmark-Based Methods

Distance- and landmark-based methods provide an alternative to purely local neighborhood aggregation by representing each node through its relationship to a selected set of reference nodes. A recent example is [49], which proposes GraphReach, a position-aware inductive GNN that captures global node position through reachability estimates to anchor nodes. The paper is motivated

by the observation that conventional GNNs often map two far-apart nodes to similar embeddings when their local neighborhoods look alike, and it addresses this by explicitly incorporating anchor-based positional information into the representation. This makes it a strong citation for the idea that landmark-style information can complement or correct the locality bias of standard message passing.

Distance- and landmark-based methods are especially relevant here because they encode node position through distances to anchors, distance features, structural-role similarity, or virtual coordinate systems rather than through adjacency aggregation alone [1, 16, 50–54].

Before their use as graph-learning features, virtual-coordinate and topology-coordinate ideas were developed for localization-free operations in wireless and sensor networks. In a virtual coordinate system (VCS), a node is represented by its hop-count distances to selected anchors, providing a logical coordinate system that can support routing and topology-aware operations without requiring physical node locations [22, 23]. Dhanapala and Jayasumana developed topology-preserving maps from virtual coordinates using dimension-reduction ideas, which can be viewed as an important predecessor of the topology-coordinate (TC) representation used in this dissertation [15, 54, 55]. They also introduced Directional Virtual Coordinates (DVC) to restore directionality information that is lost in raw VC vectors and to support deterministic routing in the virtual-coordinate domain [16]. This coordinate-based line of work was applied to anchor selection through Extreme Node Search (ENS), geo-logical and convex-subspace routing, boundary detection in 2-D and 3-D sensor networks, and node-level topology awareness in which nodes gradually infer network structure from observed communication behavior [17–19, 25, 56]. Subsequent work extended these ideas to partial virtual-coordinate and graph-geodesic recovery, as well as to theoretical questions concerning when landmark distance vectors uniquely determine graph structure, including the link dimension of a graph [1, 57]. These works position VC, TC, and DVC not merely as ad hoc feature vectors, but as a broader family of landmark-based graph-coordinate systems developed for routing, mapping, recovery, and structural inference; this dissertation adapts that family to scalable graph representation learning and downstream prediction.

Building on this earlier coordinate-system literature, our previous work [58] proposed Topology Coordinate Neural Networks (TCNN) and Directional Virtual Coordinate Neural Networks (DVCNN) as coordinate-based alternatives to standard message-passing graph neural networks. In that work, nodes are represented through graph-coordinate systems derived from topology rather than through repeated neighborhood aggregation, and the resulting coordinates are processed with conventional neural architectures. The study showed that such coordinate-based embeddings can achieve competitive or superior performance while requiring far fewer trainable parameters than many message-passing baselines, demonstrating that graph topology can be encoded effectively through coordinates without relying on standard message passing.

The theoretical side of this direction is strengthened by [57]. That paper studies the *link dimension* of a graph, defined as the minimum size of a construction set from which the graph can be exactly reconstructed using distance vectors. By focusing on ambiguity conditions and reconstruction guarantees, it provides a principled foundation for why distance-vector representations can be informative enough to recover substantial graph structure. In the context of landmark-based learning, this is useful because it supports the broader claim that a relatively small set of reference-node distances may already encode enough topological information for downstream inference.

A recent extension of coordinate-based representation learning is given by [59]. SCNode combines spatial and contextual coordinates to produce node embeddings that remain informative in both homophilic and heterophilic settings. The method explicitly leverages relative positioning to blend structural and contextual cues, and the resulting embeddings can be used as plug-and-play inputs for downstream graph models. This makes SCNode a useful citation in this section because it shows that coordinate systems need not be limited to pure shortest-path information; they can also be designed to integrate multiple complementary views of node position and context.

Recent theoretical work has also revisited how well distance-preserving embeddings capture graph structure. The paper [60] studies local distance-preserving node embeddings inspired by landmark-based shortest-path methods and evaluates how well such embeddings recover global distance information. The authors emphasize that many existing node embedding methods pre-

serve local similarity but do not faithfully capture graph distances, and they analyze when GNN-based distance-preserving embeddings can improve downstream performance. This paper is particularly useful for the related-work chapter because it connects landmark-based node embeddings to a more explicit study of metric preservation, which is closely aligned with the goals of anchor-distance representations.

Our previous work [61] extended the virtual-coordinate line of research from representation alone to sampling and embedding strategies for graph machine learning. In that work, we explored how virtual-coordinate information can be used not only to encode graph topology, but also to guide more efficient graph-data processing for downstream learning tasks. This continuation is important in the present section because it shows that coordinate-based graph learning is not limited to a single embedding method; rather, it can support a broader framework in which topology-derived coordinates are used for both representation and sampling, further strengthening the case for distance- and landmark-based approaches as viable alternatives to standard message passing.

4.4 Cross-View and Cross-Graph Transfer Learning

A useful recent overview of this area is given by [62], which surveys cross-domain graph learning and organizes existing methods according to the type of transferable information they try to preserve, including structure-oriented, feature-oriented, and mixed structure-feature transfer. This survey is especially relevant for the present section because it makes clear that graph transfer is no longer limited to small domain-adaptation settings; instead, it is increasingly framed as a broader attempt to learn representations that remain useful across graphs with different structures, attributes, and downstream tasks.

Graph transfer has been studied through domain adaptation, graph pretraining, contrastive learning, foundation-style graph models, and parameter-efficient adaptation [62–69]. The present dissertation differs by emphasizing reusable anchor-induced structural views rather than transferring only the parameters of a message-passing encoder.

An early recent example of this trend is [70], which proposes PCRec for cross-domain recommendation. That work adopts a pre-training and fine-tuning paradigm, where a graph encoder is first trained with contrastive self-supervision on a source-domain graph and then transferred to initialize learning in a target domain. The paper is useful in this section because it shows a clear “train on one, adapt to another” strategy in which transfer aims to retain useful source-domain structure while reducing bias from directly mixing source and target training.

A stronger “train on many, evaluate on new ones” perspective appears in [66]. UniGraph proposes a unified cross-domain foundation model for text-attributed graphs by using text as a common medium for node representation across otherwise heterogeneous graph domains. The model combines language models and graph neural networks, and the authors show that pretraining on diverse graph domains can support transfer to unseen graphs and tasks, including few-shot and zero-shot settings. This work is important for the present section because it treats transfer not simply as fine-tuning from one dataset to another, but as learning a shared representation space intended to generalize across domains from the start.

A related large-scale multi-domain approach is introduced in [67]. GraphFM is presented as a generalist graph transformer trained across many graph datasets rather than specialized for a single graph. Its main contribution is to show that pretraining on a diverse collection of graphs improves transferability relative to single-source pretraining, supporting the idea that graph models can benefit from the same scaling and diversity principles that have driven foundation models in language and vision. This makes GraphFM a strong reference for the “train on many, evaluate on many” formulation.

Another recent direction is explored by [71], which proposes the “one model for one graph” framework. Instead of forcing a single pretrained graph encoder to capture every domain equally well, that work pretrains a bank of expert models, each tied to a particular dataset, and then uses gating functions to select and combine experts when transferring to a new graph. This is relevant here because it addresses a central problem in multi-graph transfer: how to benefit from many

source graphs without suffering excessive negative transfer from domains that are poorly matched to the target.

Parameter-efficient adaptation is studied in [68]. GraphLoRA adapts a pretrained GNN to a new graph by combining feature-distribution alignment through a structure-aware maximum mean discrepancy term with low-rank adaptation modules and structural regularization. The method is notable because it targets cross-graph transfer under distribution shift while tuning only a small portion of the model parameters, making it especially relevant when target labels are scarce and full retraining is impractical. In the context of this section, it represents a modern form of “train on one or many, then adapt efficiently to another.”

A broader and more task-agnostic formulation is proposed in [69]. GraphBridge augments a pretrained GNN with a bridging network and new prediction heads so that knowledge can be transferred across heterogeneous tasks and domains without redesigning the original model for each new setting. The paper evaluates transfer in several scenarios, including graph-to-graph and node-to-node settings, and positions the framework as a step toward arbitrary graph transfer learning. This makes it a useful concluding citation for the section, since it shows how recent work is moving from narrow domain adaptation toward more general transfer across graph types, task formulations, and output spaces.

4.5 Scalable Graph Learning

Scalable graph learning is motivated by the fact that message passing over large graphs quickly becomes expensive in both computation and memory, especially as the number of layers increases and receptive fields expand. A recent example of this line of work is [72], which proposes GRAPES, an adaptive sampling framework for scalable GNN training. Instead of relying on fixed sampling heuristics, the method learns which nodes are most useful for the downstream task by training a second GNN to predict sampling probabilities. This is particularly relevant because it treats scalability not only as a systems issue, but also as a learning problem in which the sampler itself should adapt to graph structure and task requirements.

Scalable graph learning includes layer-wise sampling, subgraph sampling, decoupled propagation, approximate diffusion, and precomputed graph filters, each reducing repeated whole-graph message passing in a different way [10, 11, 36–38]. This dissertation follows a related preprocessing-oriented philosophy, but uses anchor-derived structural coordinates as the reusable graph signal.

Another recent direction is to make the sampling process itself learnable. The paper [73] argues that many scalable GNN methods depend on random or fixed-probability sampling schemes that are detached from the forward propagation process. To address this, the authors propose a learnable sampling mechanism that parameterizes the sampling process, allows gradients to flow through it, and can be combined with both subgraph-sampling and layer-wise-sampling designs. This work is useful in the present section because it shows that scalable graph learning can benefit from end-to-end optimization of the sampler rather than treating sampling as a purely preprocessing step.

Scalability has also been studied from the perspective of inference rather than only training. In [74], the authors focus on large-scale inductive inference and observe that even scalable GNNs based on linear propagation can remain costly when predictions must be made online for unseen nodes. Their solution is a node-adaptive propagation framework that customizes the propagation depth for each node according to its topology, thereby reducing redundant computation while preserving predictive accuracy. This paper is important for the related work chapter because it highlights that scalable graph learning must address both efficient training and efficient deployment.

A more radical approach is proposed in [75], which introduces Sketch-GNN. Rather than only reducing the constants in mini-batch sampling, this work aims for sublinear time and memory complexity with respect to graph size by training GNNs on compact sketches of the adjacency matrix and node features. Using polynomial tensor-sketch theory and trainable locality-sensitive hashing, the method seeks to break the usual linear dependence on graph size that remains in many sampling-based approaches. This makes Sketch-GNN especially notable in this section,

since it frames scalability as a problem of compressed representation rather than only smarter neighborhood selection.

Recent work has also revisited historical-embedding methods for large-scale GNN training. The paper [76] studies the feature staleness problem that arises when scalable GNN methods reuse historical node embeddings to save computation. The authors show that stale histories can substantially hurt convergence and accuracy on large benchmarks, and they propose REST, a simple training strategy that reduces staleness while remaining compatible with existing scalable pipelines. This paper is useful here because it shows that scalability techniques can introduce their own optimization biases, and that improving scalable graph learning may require revisiting those trade-offs rather than only accelerating propagation.

A different recent direction is explored in [28], which studies large-graph transductive learning through the lens of message invariance. The paper observes that out-of-batch message passing is a major source of cost in mini-batch GNN training, since it recursively depends on higher-order neighbors outside the sampled subgraph. To address this, the authors propose the TOP method, which transforms costly out-of-batch message passing into efficient in-batch computation while preserving the output of whole-graph propagation. This work is especially relevant because it aims to improve scalability without paying the usual penalty of large approximation error, making it a strong example of recent efforts to balance efficiency and fidelity in scalable graph learning.

Finally, scalability can also be pursued through graph reduction that preserves the properties most important for downstream transfer. The recent paper [77] proposes a graph sampling method designed to preserve graph structure more faithfully than random subsampling by leveraging feature homophily. Their method minimizes the trace of the data correlation matrix so as to better preserve the graph Laplacian trace, which serves as a proxy for connectivity, while maintaining lower complexity than spectral alternatives. This paper is useful in the present section because it connects scalability with expressivity, showing that the goal is not merely to sample smaller graphs, but to do so in a way that retains the structural properties that matter for GNN performance and transferability.

4.6 Synthetic Graph Generation and Benchmarking

Recent work has increasingly argued that progress in graph learning is difficult to assess when evaluation relies on only a small collection of fixed real-world datasets. The paper [78] addresses this issue by proposing a synthetic graph generation framework specifically for benchmarking graph learning methods under controlled conditions. Their central argument is that synthetic graphs make it possible to vary structural factors systematically and to study model behavior in ways that are difficult to achieve when evaluation is limited to a few standard citation or social-network benchmarks. This makes the paper an important reference for the present section because it frames synthetic graph generation not merely as data augmentation, but as a methodology for more informative benchmarking.

Benchmarking work has also shifted toward standardized datasets, systematic design-space evaluation, and diagnostic tasks for long-range reasoning [13, 45, 79, 80]. This supports the use of both real benchmarks and controlled synthetic graphs when evaluating anchor-based structural representations.

A closely related large-scale benchmarking system is introduced in [81]. GraphWorld is designed as a tunable and scalable framework that can generate very large populations of synthetic graphs for a wide range of graph learning tasks. A key contribution of this work is its emphasis on exploring regions of graph-property space that are not well covered by standard benchmarks, allowing researchers to compare GNN models across millions of statistically diverse datasets rather than a handful of reused graphs. This is especially useful in a related-work chapter because it shows how synthetic graph benchmarking can move from isolated toy examples to a systematic experimental platform.

Synthetic benchmarking has also been extended to heterogeneous graphs and explainability. The paper [82] proposes SynHING, a framework for generating synthetic heterogeneous information networks that preserve important structural and statistical properties of a target graph. In addition to supporting graph learning experiments, SynHING provides ground-truth motifs that can be used to evaluate GNN explanation methods. This is relevant here because it shows that

synthetic graph generation can serve not only predictive benchmarking, but also the benchmarking of interpretability methods in settings where real heterogeneous graphs rarely come with reliable explanatory ground truth.

Another recent use of synthetic benchmarks is to isolate specific graph-learning difficulties that are hard to measure on standard datasets. The paper [83] introduces GLoRa, a synthetic benchmark designed to evaluate whether graph learning systems can truly capture long-range dependencies. The motivation is that many proposed methods claim to address problems such as over-smoothing, over-squashing, or vanishing gradients, yet existing benchmarks do not test long-range reasoning transparently. By building controlled synthetic tasks around path-based dependencies, GLoRa provides a diagnostic environment for evaluating whether a model has genuinely learned to propagate information over long distances.

Synthetic benchmarking has also been adapted to dynamic graphs. The paper [84] introduces T-GRAB, a benchmark designed to study temporal reasoning in temporal graph neural networks. Instead of focusing only on static structural difficulty, T-GRAB constructs synthetic tasks that separately probe periodicity, delayed causal effects, and long-range spatio-temporal dependencies. This is particularly useful for the present section because it shows how synthetic graph benchmarks can be designed to isolate specific temporal reasoning skills, thereby extending the benchmarking philosophy from static graph structure to evolving graph processes.

A related recent benchmark is [85], which introduces ECHO for evaluating long-range graph propagation. ECHO focuses on synthetic tasks such as single-source shortest paths, node eccentricity, and graph diameter, all constructed to stress very long-range information flow and topological bottlenecks. This benchmark is useful in the present section because it highlights a growing trend in graph benchmarking: instead of measuring performance only on broad end tasks, recent synthetic benchmarks increasingly target specific propagation capabilities that are believed to be fundamental limitations of existing GNN architectures.

Chapter 5

Anchor-Based Representations for Graph Learning

This chapter presents the main anchor-based structural representation framework used in this dissertation. It focuses on how graph topology can be converted into reusable node-level coordinates through anchor selection, distance construction, coordinate transformation, and compression. The chapter also evaluates how these representations support efficient node classification under practical memory and computation constraints.

5.1 Overview

This chapter introduces an *anchor-based* framework for constructing scalable structural representations for graph learning. The core idea is to approximate global graph structure using a compact set of *anchor nodes*, and to represent each node by its relationships to these anchors (e.g., shortest-path distances and derived graph-coordinate features). This yields a controllable representation size determined by the number of anchors, enabling learning on large graphs where full pairwise computations are infeasible. The resulting pipeline is an embedding-based neural learning approach: graph topology is encoded into structural features before training the downstream neural model, rather than being propagated through message-passing layers during training.

The chapter is organized around a practical pipeline: we first state the learning problems of interest and the design objectives that guide the method (accuracy, scalability, and robustness across graphs with heterogeneous structure). We then detail how anchors are chosen and how anchor-to-node signals are constructed, focusing on distance-matrix-based representations and their variants. Next, we describe the transformations applied to these raw signals, including normalization steps that improve numerical stability and comparability across anchor sets, and dimensionality reduction techniques (e.g., PCA) that compress anchor-induced structure into compact embeddings.

A distinguishing focus of this chapter is *resource-aware computation*. We analyze time and memory complexity, and describe implementation strategies (such as chunked construction and

training) that allow anchor-based methods to scale under realistic hardware constraints. We then present an experimental evaluation on real-world benchmarks, comparing against standard baselines and reporting results for node classification. Finally, we provide ablation and sensitivity analyses to isolate the impact of anchor count, sampling strategy, transformations, and PCA settings, and conclude with a discussion of limitations and practical considerations.

Organization. The remainder of this chapter is structured as follows:

- Section 5.2 formalizes the target tasks and summarizes the design goals that motivate anchor-based representations.
- Section 5.3 presents anchor selection and sampling strategies, along with practical trade-offs.
- Section 5.4 describes the full representation construction pipeline, including (i) distance matrix construction, (ii) transformations and normalization, and (iii) dimensionality reduction via PCA.
- Section 5.5 analyzes computational complexity and provides a resource analysis aligned with scalable implementations.
- Section 5.6 describes datasets, baselines, and implementation details.
- Section 5.7 reports empirical results across tasks and datasets.
- Section 5.8 discusses limitations, failure modes, and directions for improvement.

5.2 Problem Formulation and Design Goals

5.2.1 Graph Learning Tasks

Let $G = (V, E)$ be a graph with $n = |V|$ nodes and $m = |E|$ edges. We consider both unweighted and weighted graphs; unless stated otherwise, we assume an undirected simple graph. Let $A \in \{0, 1\}^{n \times n}$ denote the adjacency matrix (or a weighted adjacency matrix), and let $X \in$

$\mathbb{R}^{n \times F}$ denote an optional node-feature matrix. In many settings of interest, X is missing, sparse, or weakly informative; our goal is therefore to construct structural representations derived primarily from G .

The representation framework can be used for both node classification and link prediction. In this chapter, the empirical evaluation focuses on node classification, while link prediction is studied separately in Chapter 6.

Node classification. Given labels for a subset of nodes $V_{\text{train}} \subset V$, with $y_v \in \{1, \dots, C\}$ for $v \in V_{\text{train}}$, the objective is to learn a predictor

$$f_\theta : V \rightarrow \{1, \dots, C\}$$

that generalizes to unlabeled nodes, typically evaluated on disjoint validation and test sets. The model may use X if available, but in this chapter we emphasize representations constructed from graph structure.

Link prediction. Given a set of observed edges E_{obs} and a set of candidate pairs (u, v) (including non-edges), the objective is to learn a scoring function

$$s_\theta : V \times V \rightarrow \mathbb{R}$$

that assigns higher scores to true (held-out) edges than to non-edges, under a specified evaluation protocol (e.g., negative sampling). Link prediction is particularly sensitive to how well the representation captures higher-order connectivity patterns beyond immediate neighborhoods.

5.2.2 Anchor-Based Representation: Problem Statement

A direct structural representation of a graph can be defined via pairwise node relationships, such as the all-pairs shortest-path distance matrix $D \in \mathbb{R}^{n \times n}$. However, constructing and storing such matrices is typically infeasible at scale due to $O(n^2)$ memory and potentially superlinear time.

To address this, we introduce an *anchor set* $S \subseteq V$ with $k = |S| \ll n$. The anchor-based representation maps each node $v \in V$ to a vector of its structural relationships to anchors:

$$r(v) \in \mathbb{R}^k, \quad r(v)_i = \phi(v, s_i), \quad s_i \in S,$$

where $\phi(\cdot, \cdot)$ is a graph-derived structural signal, in this chapter primarily shortest-path distance to an anchor or a transformation derived from those anchor distances. Stacking these vectors yields an anchor-to-node matrix

$$R \in \mathbb{R}^{n \times k}, \quad R_{v,i} = \phi(v, s_i).$$

This representation can be used directly as input features (optionally combined with X), or further transformed into a compact embedding $Z \in \mathbb{R}^{n \times p}$ via normalization and dimensionality reduction.

5.2.3 Design Goals

The proposed anchor-based framework is guided by the following goals.

(G1) Scalability and resource efficiency. The representation should avoid $O(n^2)$ storage and enable computation on graphs with millions of nodes. Practically, this motivates algorithms with:

- memory complexity $O(nk)$ (or lower via chunking/streaming), and
- time complexity dominated by k sparse graph traversals for shortest-path distance construction.

(G2) Expressivity of structural signals. Although $k \ll n$, the representation should capture both local and global structure:

- local connectivity (e.g., neighborhood overlap, short paths), and
- long-range relations (e.g., community structure, bridges, and graph diameter effects).

This motivates anchor selection strategies that cover diverse graph regions, and signal functions ϕ that reflect meaningful graph geometry.

(G3) Robustness to anchor choice and dataset variation. Anchor-based representations should be stable across different anchor sets and across graphs with different degree distributions and connectivity patterns. This goal motivates:

- sampling strategies that avoid pathological anchor sets,
- normalization schemes that make signals comparable across anchors, and
- sensitivity analysis with respect to k and sampling randomness.

(G4) Compatibility with downstream learning objectives. The representation should support both node-level and edge-level prediction. Concretely:

- for node classification, $r(v)$ should separate classes under a simple classifier when labels correlate with structure;
- for link prediction, the representation should admit effective pairwise scoring, e.g., via inner products or MLPs over $(r(u), r(v))$ and their interactions.

(G5) Practical implementability. The full pipeline should be implementable with common graph tooling and commodity GPUs/CPUs. This includes:

- chunked construction of R and chunked training to reduce peak memory,
- sparse-friendly operations, and
- straightforward integration into standard PyTorch training loops.

5.2.4 Chapter Focus

Given these goals, the remainder of this chapter specifies: (i) how anchors are selected, (ii) how anchor-to-node signals are computed (with emphasis on distance matrices), (iii) how signals are transformed and compressed (including PCA), and (iv) how the resulting representations behave empirically under different datasets and resource budgets.

5.3 Anchor Selection and Sampling Strategies

Anchor selection determines which subset of nodes $S \subseteq V$ (with $|S| = k \ll n$) is used to define the anchor-based representation. Since the representation of each node is expressed relative to S , the anchor set controls both (i) the *coverage* of graph regions and (ii) the *quality* of the structural signals $\phi(v, s)$. This section describes practical sampling strategies and the trade-offs among computational cost, structural diversity, and robustness.

5.3.1 Notation and General Considerations

Let $S = \{s_1, \dots, s_k\}$ denote the selected anchors. We consider sampling procedures that operate on graph structure (degrees, neighborhoods, connectivity) and do not require node features. A good anchor set should satisfy the following informal criteria:

- **Coverage:** anchors should be distributed across the graph so that distances/similarities provide discriminative information for most nodes.
- **Diversity:** anchors should avoid heavy redundancy (e.g., many anchors inside the same dense community).
- **Robustness:** small perturbations in S should not cause large performance variance.
- **Efficiency:** anchor selection should be significantly cheaper than constructing the full representation.

In graphs with multiple connected components, we typically select anchors within the largest connected component (LCC) or allocate anchors proportionally to component sizes, depending on the downstream protocol.

5.3.2 Uniform Random Sampling

The simplest strategy samples anchors uniformly at random:

$$S \sim \text{Uniform subsets of size } k \text{ from } V.$$

This method is computationally trivial and often surprisingly competitive when k is sufficiently large. However, its main drawback is variance: random sampling can over-select anchors in high-density regions or miss small but important components.

When it works well. Uniform sampling is effective when the graph is well-mixed and labels depend on broad structural patterns rather than rare substructures.

Limitations. In graphs with skewed degree distributions or strong community structure, uniform anchors may under-represent low-degree or peripheral regions.

5.3.3 Degree-Biased Sampling

A common alternative is to bias selection toward higher-degree nodes, motivated by their centrality and reachability:

$$\Pr(s = v) \propto \deg(v)^\alpha,$$

where $\alpha \geq 0$ controls the bias strength ($\alpha = 0$ reduces to uniform sampling). Degree-biased anchors often shorten typical anchor-to-node distances, which can stabilize distance-based representations by avoiding excessive “infinite” distances in sparse or weakly connected graphs.

Trade-off. While high-degree anchors improve coverage in terms of reachability, they can reduce diversity by concentrating anchors within hubs and their neighborhoods, potentially weakening the representation for community-sensitive tasks.

5.3.4 Stratified Sampling by Degree or Partitions

To balance hub coverage and diversity, we can use stratified sampling. One simple option is degree-bucket sampling: partition nodes into bins by degree quantiles, then sample a fixed number from each bin. More generally, if a graph partitioning is available (e.g., via METIS-style partition-

ing), we can allocate anchors per partition:

$$k_p = \left\lceil k \cdot \frac{|V_p|}{|V|} \right\rceil, \quad S = \bigcup_{p=1}^P S_p, \quad |S_p| = k_p,$$

where $\{V_p\}_{p=1}^P$ is a node partition. This explicitly enforces coverage across graph regions, and is particularly useful when later computation is also partition-aware.

5.3.5 Farthest-Point (Diversity-Promoting) Sampling

Diversity-promoting methods select anchors that are far apart under a chosen distance. A canonical approach is farthest-point sampling (FPS), also known as the k -center greedy heuristic:

1. Choose an initial anchor s_1 (randomly or by degree).
2. Iteratively add $s_{t+1} = \arg \max_{v \in V} \min_{s \in S_t} d(v, s)$.

Here $d(\cdot, \cdot)$ can be shortest-path distance or an approximate distance computed from partial BFS runs. FPS tends to spread anchors across the graph, improving discriminability for distance-based features.

Cost considerations. Exact FPS requires repeated distance computations and can be expensive for large graphs. Practical implementations often use approximations (e.g., sampling candidates, using landmarks, or running FPS on a coarsened graph).

5.3.6 Centrality-Driven Selection

Anchors can also be chosen via centrality measures (e.g., PageRank, betweenness, closeness), aiming to pick nodes that are informative about global connectivity. In practice, computationally cheaper centralities such as degree or approximate PageRank are preferred.

Pros and cons. Centrality anchors can improve reachability and stability, but may reduce sensitivity to community structure if anchors concentrate in the graph core.

Table 5.1: Qualitative trade-offs of anchor selection strategies.

Strategy	Coverage	Diversity	Cost
Uniform random	medium	medium (variable)	low
Degree-biased	high (core)	low–medium	low
Stratified (degree/partition)	high	medium–high	low–medium
Farthest-point (FPS)	high	high	medium–high
Centrality-driven	high (core/global)	low–medium	medium

5.3.7 Practical Strategy Used in This Dissertation

In the experiments reported in this dissertation, the default anchor selection strategy is uniform random sampling from the largest connected component unless otherwise stated. We use this strategy because it is simple, scalable, reproducible, and empirically stable enough for the graph-coordinate experiments considered here. Other strategies, such as degree-biased, stratified, farthest-point, or centrality-based sampling, are useful alternatives and are discussed above for context, but they are not the main anchor-selection strategy evaluated in the reported experiments.

This choice aligns with the design goals in Section 5.2: it is lightweight, scalable, and enables systematic sensitivity studies with respect to k and sampling randomness.

5.3.8 Summary of Trade-offs

Table 5.1 summarizes the main trade-offs among representative strategies.

The next section describes how the selected anchors are used to construct anchor-to-node representations, with emphasis on distance-matrix construction and subsequent transformations.

5.4 Representation Construction Pipeline

This section describes the end-to-end pipeline that converts a graph $G = (V, E)$ and an anchor set $S = \{s_1, \dots, s_k\}$ into node representations suitable for downstream learning. Figure 5.1 summarizes the anchor-based representation construction pipeline. The pipeline consists of (i) constructing an anchor-to-node structural matrix (primarily distance-based in this chapter), (ii) ap-

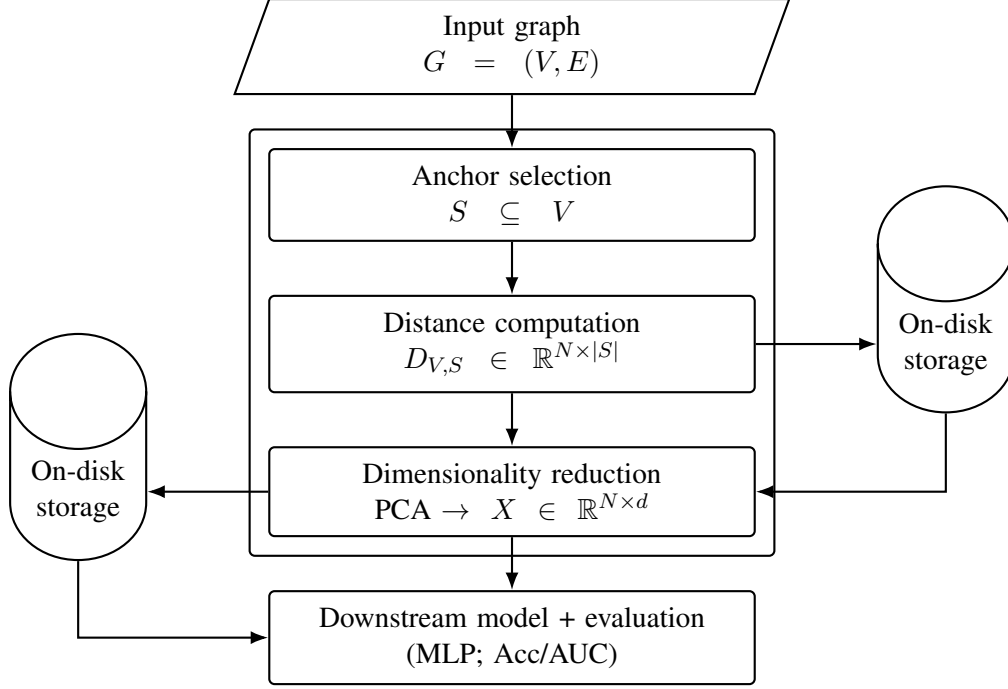


Figure 5.1: Anchor-based representation construction pipeline.

plying transformations and normalization to improve numerical stability and comparability, and (iii) optionally compressing the representation via dimensionality reduction (e.g., PCA).

5.4.1 Distance Matrix Construction

Let $d(\cdot, \cdot)$ denote a graph distance (typically shortest-path distance). The anchor-based distance representation is the matrix

$$D_S \in \mathbb{R}^{n \times k}, \quad (D_S)_{v,i} = d(v, s_i),$$

where $v \in V$ and $s_i \in S$. Compared with the all-pairs distance matrix in $\mathbb{R}^{n \times n}$, D_S reduces storage from $O(n^2)$ to $O(nk)$.

Disconnected graphs and unreachable nodes. If the graph has multiple components, $d(v, s_i)$ may be infinite for anchors outside v 's component. In practice, we simply set it to 0.

Coordinate variants derived from anchor distances. The anchor-to-node distance matrix D_S is the common starting point for the three coordinate variants examined in this chapter. In the virtual-coordinate setting, the raw node-to-anchor distance vector is used directly as the representation. In the topology-coordinate setting, an SVD/PCA-type transformation is applied to the partial distance matrix to produce a compact low-dimensional embedding. In the directional virtual-coordinate setting, or Directional Virtual Coordinates (DVC), each coordinate is constructed from a pair of anchors, yielding a directional representation that avoids the SVD step entirely. When these representations are paired with conventional neural networks, they define the corresponding models VCNN, TCNN, and DVCNN. This distinction is important in the experiments below because DVCNN provides a useful computational contrast to TCNN: both are derived from the same anchor-distance information, but they differ in whether the representation is compressed through SVD or through anchor-pair directions. In all three cases, the graph-coordinate step is a preprocessing or representation-construction stage; the subsequent neural network consumes the resulting coordinates as ordinary feature vectors rather than performing message passing over the graph.

5.4.2 Dimensionality Reduction (PCA)

The transformed anchor representation $\hat{X}_S \in \mathbb{R}^{n \times k}$ can be used directly as node features when k is moderate. When k is large (to improve accuracy) or when downstream models benefit from compact inputs, we apply dimensionality reduction to obtain

$$Z \in \mathbb{R}^{n \times p}, \quad p \ll k,$$

with principal component analysis (PCA) as a primary example.

PCA on anchor-derived features. Let $\hat{X}_S$ be mean-centered (either globally or column-wise). PCA finds orthonormal directions $W \in \mathbb{R}^{k \times p}$ maximizing retained variance:

$$Z = \hat{X}_S W.$$

We select p either as a fixed dimension or based on explained variance. PCA serves two roles: (i) compressing storage/compute, and (ii) denoising redundant anchor dimensions.

Stability across anchor sets. When anchors are resampled, the resulting PCA subspace may differ by rotations and sign flips. In practice, we observe that principal components are often stable up to sign changes, and downstream classifiers are insensitive to these symmetries when trained end-to-end on the resulting embeddings.

Incremental and chunked PCA. For large n , computing PCA on $\hat{X}_S$ directly may be memory-intensive. We therefore support:

- **Incremental PCA:** process $\hat{X}_S$ in row chunks to estimate principal components without storing the full matrix.
- **Randomized SVD:** approximate the top- p components efficiently when k is large.

These approaches align with the resource constraints discussed later in Section 5.5.

Output representation. The final node representation used in downstream learning is either:

$$(i) \hat{X}_S \in \mathbb{R}^{n \times k} \quad (\text{anchor features}), \quad \text{or} \quad (ii) Z \in \mathbb{R}^{n \times p} \quad (\text{compressed embedding}).$$

Both can optionally be concatenated with original node features X when available.

In the next section, we analyze the computational complexity and resource requirements of the above pipeline, including how k , chunk size, and graph sparsity affect runtime and memory.

5.5 Computational Complexity and Resource Analysis

This section analyzes the computational and memory costs of the anchor-based representation pipeline in Section 5.4. The goal is to make explicit how complexity scales with the number of nodes $n = |V|$, edges $m = |E|$, anchors $k = |S|$, and (when used) the reduced dimension $p \ll k$.

We also discuss practical resource-management strategies, including chunked construction and training, which are essential when nk is large.

5.5.1 Cost Model and Notation

We assume a sparse adjacency representation (e.g., CSR/CSC) with $O(n + m)$ storage. Let b denote a chunk size (number of nodes processed at a time) for streaming computations. We consider two graph regimes:

- **Unweighted graphs:** shortest-path distances computed via BFS.
- **Weighted graphs (nonnegative weights):** shortest paths via Dijkstra (or other SSSP routines).

Unless otherwise stated, we focus on the unweighted case, which is the most common in benchmark datasets and in our experiments.

5.5.2 Anchor-to-Node Distance Construction

The anchor distance matrix is $D_S \in \mathbb{R}^{n \times k}$ with entries $(D_S)_{v,i} = d(v, s_i)$.

Time complexity (unweighted). A single BFS on a sparse graph takes $O(n + m)$ in the worst case. Constructing all k columns therefore costs

$$T_{\text{dist}} = O(k(n + m)).$$

In practice, the constant factor depends on graph diameter, cache behavior, and whether the BFS explores the entire graph (e.g., when restricted to the LCC).

Time complexity (weighted). For weighted graphs, running Dijkstra from one source costs $O(m \log n)$ with a binary heap (or $O(m + n \log n)$). The total becomes

$$T_{\text{dist}} = O(k m \log n),$$

which can be substantially heavier; this motivates either smaller k or approximate distances in weighted settings.

Memory complexity. Storing D_S densely requires

$$M_{\text{dist}} = O(nk).$$

If stored in float32, the raw storage is approximately $4nk$ bytes; for int32 it is also $4nk$ bytes. For large n , even moderate k can be prohibitive, which motivates streaming and chunking.

Chunked construction. If we compute and immediately consume (transform/compress) the representation in blocks of b nodes, peak memory for the distance features reduces from $O(nk)$ to

$$M_{\text{dist,chunk}} = O(bk), \quad b \ll n,$$

at the cost of additional I/O or repeated passes depending on the implementation.

5.5.3 Dimensionality Reduction Costs

Let $\hat{X}_S \in \mathbb{R}^{n \times k}$ be the transformed and normalized matrix, and let $Z \in \mathbb{R}^{n \times p}$ be the reduced embedding.

Exact PCA (via SVD). A full SVD of an $n \times k$ matrix costs $O(\min\{nk^2, n^2k\})$, which is infeasible for large n even when k is moderate. In our setting, we only require the top- p components with $p \ll k$.

Randomized SVD / truncated PCA. A common approach computes an approximate rank- p decomposition in

$$T_{\text{pca}} \approx O(nkp) \quad (\text{up to log factors and oversampling constants}),$$

with memory dominated by storing the projection basis $W \in \mathbb{R}^{k \times p}$ and intermediate sketches. This is typically tractable when p is small (e.g., $p \leq 256$).

Incremental PCA (streaming). Incremental PCA processes $\hat{X}_S$ in chunks of size b :

$$M_{\text{pca,inc}} = O(bk + kp),$$

and avoids storing the full $n \times k$ matrix. This is the preferred mode when nk is too large to fit in host memory.

Embedding materialization. The final embedding storage is

$$M_Z = O(np),$$

which can be substantially smaller than $O(nk)$ when $p \ll k$, and is often the decisive factor enabling downstream GPU training.

5.5.4 Downstream Training Costs

Downstream learning consumes either $\hat{X}_S \in \mathbb{R}^{n \times k}$ or $Z \in \mathbb{R}^{n \times p}$.

MLP-based node classification. A two-layer MLP with hidden width h has per-epoch cost

$$T_{\text{MLP}} = O(n \cdot (kh + hC)) \quad (\text{if using } k \text{ dims}), \quad T_{\text{MLP}} = O(n \cdot (ph + hC)) \quad (\text{if using } p \text{ dims}).$$

The training footprint is dominated by the input tensor and activations. Using p instead of k reduces both compute and memory.

5.5.5 Practical Resource Management

The above asymptotics highlight three main bottlenecks: (i) k graph traversals, (ii) storage of $n \times k$ matrices, and (iii) PCA/training on high-dimensional inputs. We adopt the following practical strategies.

Data type control. Distances can be stored as int16/int32 when capped, and transformed features can often be stored as float16/float32 depending on numerical stability requirements. This yields near-linear memory savings.

Anchor budget selection. Increasing k improves representational capacity but increases distance-construction time linearly and feature memory linearly. In practice, we treat k as the primary scalability knob and use sensitivity studies to select a regime that balances accuracy and cost.

Compression before GPU. When GPU memory is limited, we prefer to compute $\hat{X}_S$ on CPU and transfer only the reduced embedding Z to GPU. This separates the heavy $O(nk)$ construction from the learning stage.

5.5.6 Summary

The anchor-based pipeline replaces $O(n^2)$ structural computation with an $O(k(n+m))$ distance-construction stage and an $O(nk)$ feature-processing stage. The primary practical challenge is the $n \times k$ intermediate representation; chunking, streaming, and PCA-based compression reduce peak memory and enable training on large graphs under realistic resource constraints.

5.6 Experimental Setup

This section describes the datasets, baselines, and implementation details used to evaluate the anchor-based representation pipeline introduced in Sections 5.3–5.5. The experiments are designed to assess (i) predictive performance on standard graph learning tasks and (ii) computational feasibility under realistic resource constraints.

5.6.1 Datasets

We evaluate on real-world OGBN benchmarks.

Real-world benchmark graphs. We consider widely-used citation, co-purchase, and interaction graphs that support node classification and/or link prediction. For each dataset, we follow the official split protocol when available; otherwise, we use standard train/validation/test splits with disjoint node or edge sets depending on the task. Unless otherwise stated, all graphs are treated as undirected and unweighted after standard preprocessing (e.g., symmetrization and removal of self-loops).

OGB benchmarks. To assess scalability and reproducibility, we include datasets from the Open Graph Benchmark (OGB) suite and follow the provided splits and evaluation metrics. In this chapter, the two primary node-classification benchmarks are OGBN-Products and OGBN-Proteins.

OGBN-Products. OGBN-Products is an Amazon co-purchasing network with 2,449,029 nodes, 61,859,140 edges, and 100-dimensional node features derived from bag-of-words product descriptions. The task is 47-class node classification. For graph-coordinate methods, we extract coordinates from the largest connected component, which contains 2,385,902 nodes (97.42% of the graph), and map nodes outside this component to zero embeddings. In the reported Graph Coordinates Neural Network (GCNN) experiments, 1000 random anchors are used for the main TC/DVC configurations.

OGBN-Proteins. OGBN-Proteins contains 132,534 nodes and 39,561,252 edges. It is treated here as an 8-layer graph, where each edge-feature dimension defines one graph layer and the weighted distance between adjacent nodes is taken as 1—confidence for that association. Protein species information is encoded with one-hot vectors and concatenated with the structural representation. In the main experiments, anchors are sampled independently for each layer, with 1000 anchors used in the principal TC/DVC configurations.

These benchmarks stress both representation quality and the ability to scale anchor-based computations.

Preprocessing. Across all datasets, we report $n = |V|$ and $m = |E|$ after preprocessing. If graphs contain multiple connected components, we either restrict to the largest connected component (LCC) or apply a consistent handling strategy for unreachable distances (Section 5.4.1), depending on the evaluation protocol.

5.6.2 Baselines

For context, we compare the reported anchor-based coordinate models with published OGB leaderboard entries and commonly used graph-learning baselines reported in prior work. These comparisons are intended to place VCNN, TCNN, and DVCNN in relation to established methods such as GCN, GraphSAGE, Node2Vec, and leaderboard models; they should not be interpreted as a new exhaustive baseline sweep implemented in this dissertation.

Within the anchor-based family, the main internal comparisons are among VCNN, TCNN, and DVCNN, and among different anchor budgets, coordinate dimensions, and downstream neural-network configurations.

Feature-only baselines. When node features X are available, published comparison points include:

- **MLP on X :** a multi-layer perceptron trained directly on node features, without using graph structure.
- **Linear classifier on X :** logistic regression (or linear SVM) as a lightweight reference.

Standard GNN baselines. Relevant comparison categories are widely-used message-passing architectures as structural baselines:

- **GCN:** graph convolutional networks using normalized adjacency aggregation.

- **GraphSAGE:** neighborhood sampling and aggregation.

These baselines test whether anchor-based features can match or complement neighborhood aggregation, especially in regimes where multi-hop message passing is costly or over-smooths. For context, we compare the proposed coordinate models with published OGB leaderboard entries and commonly reported baselines. These comparisons are not an exhaustive reimplementations of all baselines.

Structural/positional baselines. To contextualize anchor-based distances as a form of positional encoding, we include:

- **Laplacian eigenmaps / spectral PE (when feasible):** leading eigenvectors of the normalized Laplacian, used as node positional features.
- **Random-walk-based embeddings:** shallow embedding methods (e.g., node2vec) when computationally feasible under the dataset scale.

These baselines represent alternative ways to capture global structure, often with different computational trade-offs than anchor distances.

Anchor variants (internal baselines). To isolate the effect of specific design choices, we also treat several anchor variants as baselines within the anchor family:

- anchor features ($n \times k$) vs. PCA-compressed embeddings ($n \times p$) (Section 5.4.2);
- different anchor sampling strategies (Section 5.3).

5.6.3 Implementation Details

Anchor selection. Unless otherwise stated, anchors are sampled uniformly at random from the largest connected component, with anchor count k treated as a primary hyperparameter. For robustness, we report averages over multiple random anchor seeds, and we use fixed seeds for reproducibility.

Distance computation. For unweighted graphs, anchor-to-node distances are computed via BFS from each anchor. For weighted graphs (when applicable), we use Dijkstra with nonnegative weights. Distances may be capped at $d_{\max}$ to control dynamic range and to handle unreachable cases consistently (Section 5.4.1).

Dimensionality reduction. When PCA is used, we select an embedding dimension $p \ll k$ and fit PCA on the transformed anchor matrix. For large graphs, we use incremental or randomized PCA to avoid materializing the full $n \times k$ matrix in memory. The resulting embedding $Z \in \mathbb{R}^{n \times p}$ is used as input to downstream models.

Downstream models and training. For node classification, we train a linear classifier or MLP on the anchor-derived representations (optionally concatenated with X). In the single-layer OGBN-Products setting, graph coordinates are concatenated directly with the node feature vector. In the multi-layer OGBN-Proteins setting, graph coordinates extracted from the 8 graph layers are concatenated and then combined with the species one-hot encoding before being passed to the downstream neural network. The same coordinate representations can also be used for link prediction by constructing edge features from pairs of node embeddings; that setting is studied separately in Chapter 6.

Resource-aware execution. To handle large n and k , we implement:

- **CPU–GPU separation:** compute distances and PCA on CPU, then transfer only compressed embeddings to GPU for training;

Evaluation metrics. We report task-standard metrics: accuracy / macro-F1 for node classification (depending on dataset conventions), and ROC-AUC or Hits@K for link prediction following the benchmark protocol. For each setting, we report mean and standard deviation over multiple runs (different random seeds and/or anchor samples) to quantify stability.

5.7 Results

This section reports node-classification results for the anchor-based representations discussed earlier in the chapter. The emphasis is on how much predictive performance can be obtained from graph-coordinate representations while keeping the downstream model shallow and the number of trainable parameters small. We focus on OGBN-Products and OGBN-Proteins because they provide complementary settings: a large single-layer co-purchase graph and a weighted multi-layer biological graph.

5.7.1 Node Classification Performance

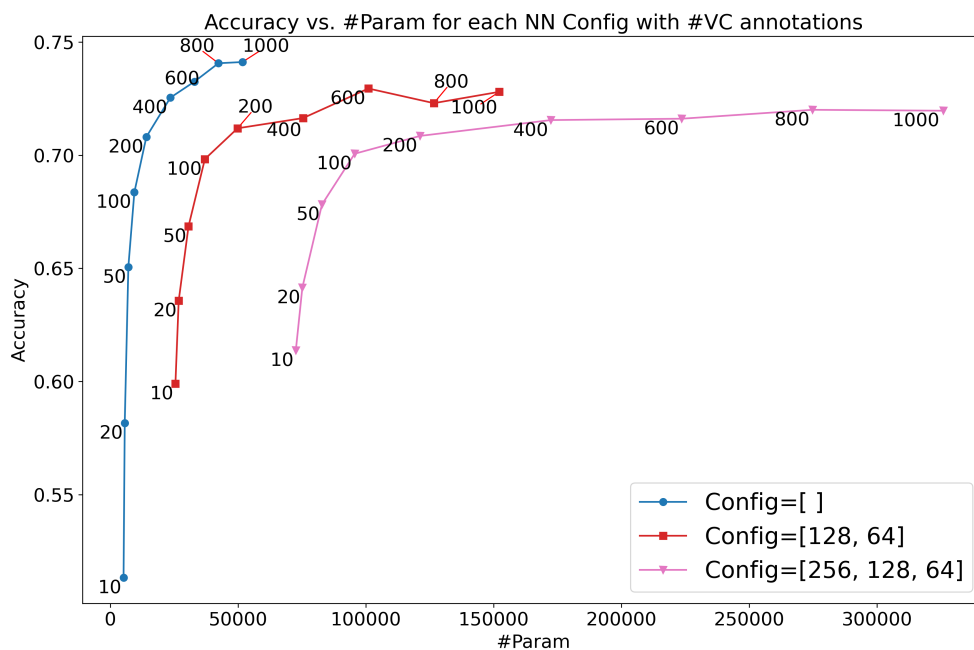


Figure 5.2: VCNN performance on OGBN-Products as the number of virtual coordinates and hidden-layer configuration vary. Each point is annotated with the number of virtual coordinates.

Figure 5.2 and Figure 5.3 summarize the VCNN trade-off between predictive performance and model complexity, while Table 5.2 and Table 5.3 place the broader anchor-based family in the context of selected OGB leaderboard entries. Several patterns are consistent across the two datasets. First, anchor-based graph-coordinate methods remain competitive with substantially more com-

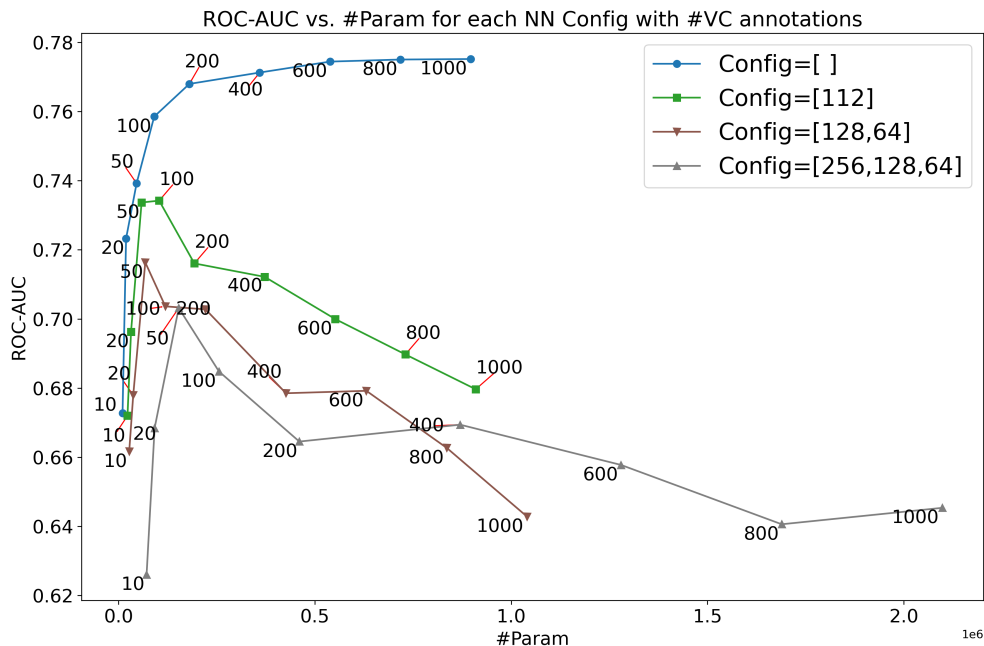


Figure 5.3: VCNN performance on OGBN-Proteins as the number of virtual coordinates and hidden-layer configuration vary. Each point is annotated with the number of virtual coordinates.

plex baselines while using far fewer trainable parameters. Second, the more processed coordinate systems (TC and DVC) generally achieve stronger predictive performance than raw VC features, but the VC results remain strong enough to be practically useful when preprocessing simplicity is important. Third, the best parameter–accuracy trade-off is not always achieved by the most expressive downstream network; shallow models are often sufficient once the structural representation is informative.

5.7.2 Topology Coordinates and Directional Virtual Coordinates

TCNN and DVCNN represent two complementary ways of transforming the same underlying anchor-distance information into coordinates for conventional neural networks. TCNN uses an SVD/PCA-type compression of the partial distance matrix and therefore emphasizes a compact low-dimensional summary of the dominant structural variation. DVCNN instead constructs coordinates from pairs of anchors, which avoids the SVD step and reduces preprocessing complexity while still encoding position relative to the graph topology.

Table 5.2: Comparison of selected OGBN leaderboard entries for OGBN-Products. Blank value indicates that the rank is not listed in the official leaderboard.

Method	Rank	ACC	Trainable Parameters
GLEM+GIANT+SAGN+SCR [86]	1	0.8737 ± 0.0006	139,792,525
LD+GIANT+SAGN+SCR [87]	2	0.8718 ± 0.0004	110,636,896
Full-batch GraphSAGE [3]	49	0.7850 ± 0.0014	206,895
TCNN-100A	51	0.7606 ± 0.0037	22,624
Full-batch GCN [2]	52	0.7564 ± 0.0021	103,727
TCNN-100B		0.7466 ± 0.0027	9,447
VCNN-800		0.7406 ± 0.0041	42,347
Node2Vec [88]	55	0.7249 ± 0.0010	313,612,207
DVCNN-200/100		0.6954 ± 0.0014	9,447

Table 5.3: Comparison of selected OGBN leaderboard entries for OGBN-Proteins. Blank value indicates that the rank is not listed in the official leaderboard.

Method	Rank	ROC-AUC	Trainable Parameters
LD+GAT [89]	1	0.8942 ± 0.0007	664,233,700
GIPA(Wide&Deep) [89]	2	0.8917 ± 0.0007	17,438,716
GEN+FLAG+node2vec [90]	17	0.8251 ± 0.0043	487,436
TCNN-0.999		0.8032 ± 0.0073	352,016
DVCNN-400/200	18	0.7916 ± 0.0086	180,208
TCNN-0.99		0.7827 ± 0.0012	10,416
GeniePath-BS [91]	19	0.7825 ± 0.0035	316,754
VCNN-200		0.7680 ± 0.0008	180,208
GCN [2]	22	0.7251 ± 0.0035	96,880
Node2vec [88]	24	0.6881 ± 0.0065	17,094,000

This distinction produces a meaningful performance–cost trade-off in the benchmark results. On OGBN-Products, TCNN-100A and TCNN-100B are the strongest reported graph-coordinate configurations in Table 5.2, while DVCNN-200/100 remains competitive using the same small downstream parameter count as TCNN-100B. On OGBN-Proteins, TCNN-0.999 is the strongest reported graph-coordinate setting in Table 5.3, but DVCNN-400/200 still reaches competitive ROC-AUC while using far fewer trainable parameters than the larger TCNN setting. In practical terms, TCNN is often the stronger choice when maximizing predictive performance within this family is the main goal, whereas DVCNN is attractive when preprocessing simplicity and efficiency are more important because it removes the SVD stage.

5.7.3 Virtual Coordinates as a Direct Embedding Baseline

The VCNN results are useful because they separate the value of anchor-based sampling from the value of additional feature engineering. In VCNN, the raw virtual-coordinate vector is used directly as the node representation, eliminating the SVD step required by TC and the anchor-pair construction required by DVC. This makes VCNN the simplest member of the anchor-based family from an implementation perspective.

Empirically, the direct VC baseline remains competitive. On OGBN-Products, VCNN-800 reaches 0.7406 ± 0.0041 accuracy with 42,347 trainable parameters, while on OGBN-Proteins, VCNN-200 reaches 0.7680 ± 0.0008 ROC-AUC with 180,208 trainable parameters. These results are below the strongest TC configurations, but they remain strong enough to show that a large fraction of the predictive signal is already present in the raw node-to-anchor distance vectors. The main practical trade-off is therefore clear: TC and DVC can achieve stronger predictive performance in the reported settings, whereas VC substantially reduces preprocessing complexity and remains attractive when ease of deployment or reduced feature engineering is a priority.

A second practical observation is that VCNN often works best with shallow downstream models. As shown in Figure 5.2, performance on OGBN-Products improves as the number of virtual coordinates increases up to roughly 400 coordinates and then begins to saturate, while very shallow configurations are often the strongest settings. Figure 5.3 shows a similar pattern on OGBN-Proteins: the best VC configurations are also shallow, and increasing model depth or increasing the number of coordinates beyond the useful range does not yield consistent improvements. Taken together, these results suggest that when node labels are strongly tied to graph position, a more complex neural network does not automatically improve performance once the anchor-based representation is already informative.

5.8 Limitations and Discussion

This section discusses limitations of anchor-based distance representations and practical considerations that emerge from the method design (Sections 5.3–5.4) and the evaluation protocol

(Sections 5.6). The goal is to clarify when the approach is expected to work well, when it may fail, and which extensions are promising.

5.8.1 Representation Limits and Structural Ambiguity

Information loss from anchor compression. Replacing full pairwise structure with an $n \times k$ anchor-to-node matrix inevitably discards information. When $k \ll n$, different nodes can share similar anchor-distance profiles, particularly in:

- highly regular graphs (e.g., near-regular lattices or expanders),
- graphs with strong automorphisms (symmetries), or
- dense communities where many nodes are structurally indistinguishable at the scale of the anchors.

In such cases, increasing k or using diversity-promoting anchor selection can help, but the representation remains bounded by the anchor resolution.

At the same time, the empirical ambiguity rate can still be quite low even with a relatively small anchor budget. Table 5.4 reports the ambiguity diagnostics for the OGBN-Products virtual-coordinate representation. On OGBN-Products with 1000 anchors, only 35,247 sampled ambiguous edges were observed among 61,859,140 sampled candidate edges, or about 0.057%. In the same setting, approximately 55,801 nodes shared identical coordinates, which is only about 2.28% of the full graph. These numbers indicate that anchor compression does lose information, but not so aggressively that the resulting representation becomes unusable on a large benchmark graph. Instead, the representation appears to preserve most of the coarse topological distinctions needed for downstream prediction.

Shortest-path distances can be too coarse. Shortest-path distance captures geodesic proximity but ignores:

- the number of alternative paths (path multiplicity),

Table 5.4: Ambiguity diagnostics for virtual-coordinate representations on OGBN-Products with 1000 anchors.

Diagnostic	Count	Percent
Sampled ambiguous edges	35,247	0.057%
Nodes with non-unique coordinates	55,801	2.28%

- edge directionality and weights (if simplified),
- local density variations and path multiplicity that are not captured by geodesic length alone.

As a result, nodes in different structural roles can share identical shortest-path distances to anchors. Alternative distance-derived constructions may partially address some of these limitations, but they typically increase preprocessing complexity.

5.8.2 Sensitivity to Anchor Choice

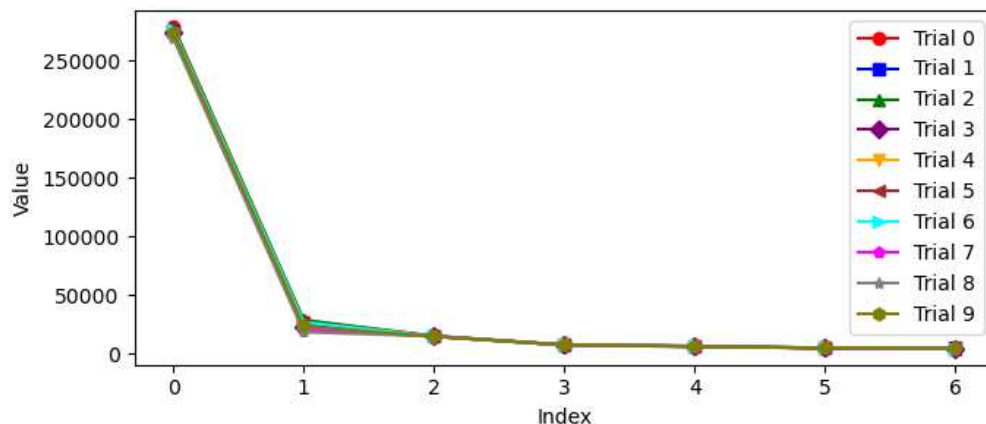


Figure 5.4: Leading singular values of partial distance matrices from ten random 1000-anchor sets on OGBN-Products. The close agreement across anchor sets indicates that random anchors recover a similar dominant structural subspace.

Figure 5.4 shows that random anchor sets recover similar leading singular-value patterns.

Variance across random anchor sets. Although anchor sampling is efficient, performance can vary noticeably across anchor seeds when k is small or the graph has heterogeneous structure (e.g., strong community imbalance or rare motifs). This motivates reporting $\text{mean} \pm \text{std}$ over repeated

random anchor sets. Alternative sampling strategies may improve stability in some settings, but the reported experiments primarily use random anchors to keep the protocol simple and reproducible.

However, the variance induced by random anchors is not arbitrary. In the GCNN experiments, the leading singular values of the partial distance matrices remained close across multiple random anchor sets on OGBN-Products, indicating that different random anchor choices still capture a similar dominant structural subspace. This observation supports the use of random anchors as a practical default: while exact coordinates change across samples, the large-scale structural information extracted from those samples is often stable enough for consistent downstream learning.

Coverage challenges in sparse or fragmented graphs. If the graph has many components or low connectivity, anchors may fail to adequately cover all regions. Even with ∞ imputation and distance capping, the resulting feature distributions can become dominated by the chosen imputation constant, reducing discriminability. Restricting to the LCC improves consistency but may change the task definition if labels exist outside the LCC.

5.8.3 Normalization and Comparability Issues

Dependence on global distance scale. Distance distributions depend on graph diameter and density. Without careful normalization, models may overfit to dataset-specific distance scales, limiting transfer across graphs or across anchor sets.

Potential leakage via statistics. If normalization statistics (means/variances) are computed using the full node set, this may introduce subtle leakage in transductive protocols. In strictly inductive settings, statistics should be computed only from the training graph/portion, and then applied unchanged at evaluation time. While leakage may be negligible in some transductive benchmarks, it should be controlled to ensure fair comparisons.

5.8.4 Computational Constraints and Practical Bottlenecks

Linear scaling in k is still expensive. Distance construction scales as $O(k(n + m))$ for unweighted graphs (Section 5.5). For large k , even linear scaling can be prohibitive, especially when:

- the graph is large and high-diameter (long BFS frontier propagation),
- weighted edges require Dijkstra-like routines, or
- repeated runs over multiple anchor seeds are needed for robustness.

Chunking mitigates memory pressure but does not reduce the inherent traversal cost.

Intermediate storage can dominate. Even when the final embedding is compressed to $n \times p$, intermediate $n \times k$ matrices may exceed memory limits if materialized. Streaming and incremental PCA address this, but increase engineering complexity and may introduce additional I/O overhead.

5.8.5 Task-Specific Limitations

Link prediction requires interaction modeling. Using node embeddings derived from anchor distances is not always sufficient for link prediction unless the edge scoring function captures meaningful interactions (e.g., elementwise products, bilinear forms, or MLP-based pair encoders). Simple inner products can underfit when the representation is primarily geometric rather than metric-preserving in an inner-product space.

Node classification depends on label–structure correlation. Anchor representations are structural; if labels are driven primarily by attributes not reflected in connectivity (or if homophily is weak and structural roles dominate), performance may degrade relative to feature-based models or specialized heterophily-aware GNNs.

5.8.6 Discussion and Potential Extensions

Beyond geodesic distances. A natural extension is to enrich the use of anchor-derived shortest-path information rather than relying on raw distances alone. For example, one could study alternative truncation rules, different normalization choices, or additional coordinate constructions derived from the same distance matrix. These variations may better capture path multiplicity or community separation, although they also increase preprocessing and tuning complexity.

Representation choice within the graph-coordinate family. A practical extension is to choose among VC, TC, and DVC more deliberately according to the target deployment setting. The present results suggest that these representations occupy different points on the preprocessing–accuracy spectrum rather than forming a strict ranking that is valid for every task and graph.

The comparison among VC, TC, and DVC also suggests a practical design principle: representation choice should be treated as a trade-off among preprocessing cost, model size, and predictive performance rather than as a search for a single universally best embedding. VC removes additional feature engineering and can work well with shallow networks, making it attractive when preprocessing simplicity matters. TC adds an SVD/PCA compression step and often gives the strongest performance in the reported settings. DVC avoids that SVD step by constructing coordinates from anchor pairs, which can reduce preprocessing cost while still remaining competitive. This makes the anchor-based family flexible rather than monolithic: different applications may legitimately favor different points on the simplicity–accuracy spectrum.

Adaptive anchor selection. Rather than fixed sampling, anchors can be selected adaptively:

- **data-driven anchors:** select anchors that maximize downstream validation performance,
- **active coverage:** iteratively add anchors in underrepresented regions,
- **task-aware anchors:** favor nodes near decision boundaries or uncertain regions.

These methods may reduce the required k but introduce additional optimization overhead.

Inductive generalization. In fully inductive settings (new nodes/graphs at test time), anchor-based methods require either:

- recomputing anchor distances in the new graph, or
- defining anchors in a transferable way (e.g., via node attributes or structural prototypes).

Designing inductive anchor schemes remains an open direction, especially for cross-graph transfer.

5.8.7 Summary

Anchor-based distance representations offer a scalable and conceptually simple alternative to full pairwise structural computation, but they are limited by anchor resolution, sensitivity to anchor choice, and the coarseness of shortest-path distances. Many of these limitations can be mitigated through better anchor coverage, more careful normalization, alternative coordinate constructions derived from anchor distances, and streaming implementations. These observations motivate the methodological refinements and extensions explored in subsequent chapters.

Chapter 6

Link Prediction

This chapter extends the anchor-based coordinate framework from node classification to link prediction. It studies how node-level structural coordinates can be combined into edge-level representations, how negative sampling and evaluation protocols affect performance, and where graph-coordinate methods succeed or fail on synthetic and real-world link prediction benchmarks.

6.1 Overview

This chapter investigates *link prediction* using Graph Coordinate (GC) embeddings: given a graph with observed edges, the goal is to predict valid but unrecorded edges by learning a scoring function over pairs of node representations derived from anchor-based graph coordinates. We evaluate three coordinate variants (VC, TC, and DVC) and examine how representation choice, edge embedding combination strategy, and negative sampling interact to determine performance across graph types and scales.

The real-world experiments use link-property prediction datasets from the Open Graph Benchmark (OGB) suite. In contrast to the OGBN node-property benchmarks used in Chapter 5, OGBL datasets are designed for evaluating edge-level prediction: models are trained on observed training edges and evaluated by ranking testing positive edges against negative candidate edges. In this chapter, the primary OGBL benchmarks are OGBL-DDI, a drug-drug interaction network evaluated by Hits@20, and OGBL-PPA, a protein-protein association network evaluated by Hits@100. Both benchmarks do not provide node features in the experimental setting considered here, so the link predictors rely on structure-derived graph-coordinate representations.

In this chapter, an edge scorer is a scoring function, usually learned in this chapter, that assigns a real-valued score to a candidate edge. For a candidate node pair (u, v) , the pipeline first constructs node-coordinate embeddings z_u and z_v , combines them into an edge-level feature vector e_{uv} , and maps this edge representation to a score $s_\theta(u, v) = g_\theta(e_{uv})$. In OGBL-style link pre-

diction, candidate edges are ranked by these scores, and Hits@K measures whether the held-out positive edge appears among the top K ranked candidates.

Experiments proceed in two stages. We first validate the approach on controlled synthetic graphs (line and circle topologies at multiple scales), where ground truth is fully known and ablations can be run cleanly. We then evaluate on OGBL benchmarks described above to assess whether findings from synthetic settings hold in practice. The results show that GC-based methods are reliable on synthetic graphs with clear coordinate structure, while real-world benchmarks require more careful treatment. In particular, single-coordinate-channel edge scorers are limited on OGBL-DDI, but late fusion of complementary coordinate views substantially improves the no-leak OGBL-DDI result. The larger OGBL-PPA setting remains unresolved under the current pipeline and highlights the need for further scalability and optimization work.

The remainder of the chapter is organized as follows.

- **Section 6.2** defines the link prediction task and describes how edge examples are constructed from an input graph, including how we treat directionality, edge multiplicity, and any node/edge attributes used by the model.
- **Section 6.3** details the data split strategy and the evaluation protocol, specifying the ranking-based metrics and the exact candidate sets used during evaluation.
- **Section 6.4** introduces the negative sampling strategy used for training and validation, and discusses how sampling choices, particularly the negative ratio, influence both optimization stability and reported performance.
- **Section 6.5** presents the GC-based model and training procedure, including the three coordinate variants, the edge combination functions, the scoring MLP, and the loss.
- **Section 6.6** summarizes datasets, baselines, hyperparameters, and implementation details needed for reproducibility.

- **Sections 6.7–6.8** report results on both synthetic and real-world datasets, and provide error analysis identifying the conditions under which the method succeeds or degrades.

Overall, this chapter characterizes the operating regime of GC-based link prediction: reliable on synthetic graphs with clear coordinate structure, and progressively limited as graph scale and structural complexity increase toward real-world benchmarks.

6.2 Task Definition and Data Construction

In link prediction, the observed graph is treated as an incomplete record of relations: an edge $(u, v) \in E$ represents a known relation between two nodes, while the learning task is to assign scores to candidate node pairs and identify which currently unobserved pairs are likely to correspond to missing or otherwise valid but unrecorded edges. The problem has a long history in network analysis, including classical work on predicting potential links in social networks and proximity-based heuristics such as common neighbors and Adamic-Adar scores [92, 93]. More recent work has studied link prediction using graph neural networks and learned structural heuristics [14], and benchmark suites such as the Open Graph Benchmark provide standardized large-scale link-property prediction datasets and evaluation protocols [13].

For supervised training, observed training edges are treated as positive examples. Because the number of possible node pairs is usually much larger than the number of observed edges, the model also requires negative examples. Ideally, a negative edge example would be a candidate node pair for which one has independent evidence that no relation exists. In many real-world graph datasets, however, such confirmed non-relations in the training split are not available. Therefore, in real-world graph datasets and synthetic datasets, we approximate training negative examples by sampling node pairs that are not present in the training edge set. A negative edge, or negative sample, is a candidate pair $((u, v) \notin E_{\text{train}})$ that is treated as a non-edge during training or evaluation. However, in real graphs, an unobserved pair is not always a confirmed false edge; it may instead be an unknown or missing edge. This distinction is especially important for the coordinate-based methods in this chapter. Graph coordinates are generated from the currently available training

graph, so held-out validation/test edges and genuinely unknown edges are not included when computing shortest-path distances to anchors. As a result, the coordinate construction may not fully reflect the latent graph on which links exist: two nodes that should be close in the complete graph may appear farther apart, disconnected, or structurally less related in the training graph. Thus, link prediction with graph coordinates has an additional challenge beyond edge scoring: the node representations themselves are built from an incomplete graph, and negative sampling must be interpreted as a practical approximation rather than as access to a complete set of true non-edges.

Following this setting, let $G = (V, E)$, where V is the set of nodes and E is the set of observed edges available under the training or evaluation protocol. The objective is to learn a scoring function that assigns high scores to *true* (observed or valid but unrecorded) edges and low scores to *false* (non-existent) edges. In the simplest setting, the graph is homogeneous and unweighted; however, the pipeline described in this chapter is compatible with (i) directed or undirected graphs, (ii) multi-edges, and (iii) optional node/edge attributes.

6.2.1 Problem Setup

Let $|V| = n$ and let the observed edge set be $E \subseteq V \times V$. For a candidate pair (u, v) , the model outputs a real-valued score

$$s_\theta(u, v) \in \mathbb{R}, \quad (6.1)$$

parameterized by θ . Higher scores indicate higher confidence that (u, v) should be an edge. During training, we construct labeled examples from edge pairs:

$$\mathcal{D}^+ = \{(u, v) : (u, v) \in E_{\text{train}}\}, \quad (6.2)$$

$$\mathcal{D}^- = \{(u, v) : (u, v) \notin E_{\text{obs}}\}, \quad (6.3)$$

where E_{train} is the training edge set, and E_{obs} denotes the set of edges that must be treated as *existing* for the purpose of avoiding false negatives (e.g., the union of training/validation/test edges if all are

known, or at least the training graph plus any edges revealed to the sampler). Section 6.4 specifies how $\mathcal{D}^-$ is instantiated in practice.

6.2.2 Directed vs. Undirected Graphs

The construction of candidate pairs depends on whether G is directed.

- **Undirected:** Edges are unordered, so (u, v) and (v, u) are equivalent. We canonicalize pairs (e.g., enforce $u < v$ under an index ordering) to avoid duplicates in datasets and evaluation.
- **Directed:** Edges are ordered, so (u, v) and (v, u) are distinct. Negative sampling and evaluation must respect directionality.

If the original dataset is undirected but the implementation stores adjacency in a directed format (common in libraries), we ensure that data construction does not accidentally double-count reciprocal edges unless explicitly intended.

6.2.3 Training Graph and Message-Passing Scope

Many link prediction models use node representations computed from a *training graph* $G_{\text{train}} = (V, E_{\text{train}})$, for example through message passing or other neighborhood aggregation. In standard link-prediction protocols, validation and test edges are withheld from the graph used to construct node representations so that the model cannot indirectly observe the edges it is later asked to predict [13, 14]. Following this common no-leakage practice, we restrict any representation learning that depends on graph connectivity (e.g., GNN propagation, anchor-based distance features, structural encodings) to E_{train} only. Validation and test edges are used strictly for supervision-free evaluation (ranking) and must not be inserted into the graph for computing node embeddings.

In contrast, features that are independent of edges (e.g., raw node attributes or externally provided embeddings) can be used across all splits without leakage concerns. For edge attributes, we follow the same rule: edge-derived features are available only for edges in E_{train} during training-time representation computation.

6.2.4 Constructing Edge Examples

We form supervised examples by pairing positive edges with sampled negatives. For each positive $(u, v) \in E_{\text{train}}$, we generate K negative pairs

$$\{(u, \tilde{v}_1), \dots, (u, \tilde{v}_K)\}, \quad (6.4)$$

or alternatively K pairs of the form $(\tilde{u}_k, v)$, depending on the sampling scheme. The training dataset is thus a collection of edge pairs and binary labels:

$$\mathcal{D}_{\text{train}} = \{(u_i, v_i), y_i\}_{i=1}^m, \quad y_i \in \{0, 1\}. \quad (6.5)$$

For scalability, we do not materialize $\mathcal{D}_{\text{train}}$ as a single static list when m is large; instead, we stream positives in batches and generate negatives on-the-fly (Section 6.4), which reduces storage and enables flexible K without regenerating files.

6.2.5 Candidate Sets for Evaluation

Evaluation in link prediction is commonly performed via ranking: for each query edge (or query node), the model ranks the true target against a set of negatives. Two typical evaluation modes are:

- **Edge-centric (pairwise) ranking:** Each positive test edge (u, v) is evaluated against a set of sampled negatives that share the same source u (or same destination v), producing a rank for the true edge.
- **Node-centric (all-candidates) ranking:** For a fixed source u , rank the true neighbor v against *all* nodes in V (or a large subset). This is more faithful but often computationally expensive.

In large-scale settings, we typically use sampled candidate sets for tractability while ensuring the evaluation protocol is consistent across methods. Section 6.3 specifies the exact protocol and metrics.

6.2.6 Scalable Data Representation

To support scalable training, we store and process data in representations optimized for streaming and membership testing:

- **Edge lists:** Store E_{train} , E_{val} , and E_{test} as integer index pairs (COO-style) for efficient sequential reads and shuffling.
- **Adjacency for sampling:** Maintain adjacency data structures (e.g., CSR/CSC) for fast neighbor queries when needed, and a hash-based or sorted representation for fast “is-edge” membership checks during negative sampling.

These choices enable the pipeline to scale to large $|E|$ without requiring the full Cartesian product of candidate pairs, and they allow negative sampling to avoid collisions with observed edges efficiently.

Summary. This section defined the link prediction task and described how supervised edge-pair examples are constructed from the training graph while avoiding leakage. The next subsections specify the split protocol (Section 6.3) and the negative sampling strategy (Section 6.4) in detail.

6.3 Splits and Evaluation Protocol

This section describes how we partition the edges in a graph dataset into training/validation/test sets and how we evaluate link prediction performance in a way that is both scalable and comparable across methods. The key requirements are: (i) *no leakage* of validation/test edges into any connectivity-dependent representation learning, (ii) *consistent candidate sets* for ranking-based evaluation, and (iii) *reproducibility* via fixed random seeds and deterministic data construction whenever possible.

6.3.1 Edge Splitting

Given the observed edge set E , we partition it into three disjoint subsets:

$$E = E_{\text{train}} \dot{\cup} E_{\text{val}} \dot{\cup} E_{\text{test}}. \quad (6.6)$$

We use $G_{\text{train}} = (V, E_{\text{train}})$ as the *training graph*. Any operation that depends on graph structure (e.g., message passing, anchor-distance computation, structural encodings) is computed using E_{train} only. Validation and test edges are held out strictly for evaluation, as what OGB leaderboard [94] requires.

Transductive setting. We assume a transductive evaluation protocol: the node set V is fixed and shared across all splits, and only edges are withheld. This matches common benchmarks where all nodes are known at training time, but some edges are hidden.

Undirected graphs. For undirected graphs, we split over canonicalized edges to avoid having (u, v) and (v, u) placed into different splits. Concretely, we store each undirected edge once using a canonical ordering (e.g., $u < v$ under a fixed node indexing) and perform splitting on this canonical set. This is consistent with the convention used by OGB link-prediction datasets: undirected relations are split as graph edges, while the loaded graph representation may expand them into bidirectional edges for computation [94].

Handling multi-edges and self-loops. If multi-edges exist, we treat each observed edge as a positive instance; if the downstream task requires simple graphs, multi-edges can be collapsed (keeping a single instance) prior to splitting. Self-loops may be removed unless they are meaningful for the dataset; if retained, they are handled consistently in both splitting and negative sampling.

6.3.2 Validation and Test Queries

We evaluate the model using the validation and test positive edges defined by the official dataset split. These edges are disjoint from the training edges and are not included in the training graph

used to construct graph coordinates or other connectivity-dependent representations. For each validation or test positive edge, the model score is compared against the scores of the corresponding negative candidate edges under the same evaluation protocol. Thus, evaluation is performed as an edge-ranking problem: true validation or test edges should receive higher scores than sampled non-edges.

For OGB-style link-prediction benchmarks, such as OGBL-DDI, we follow the benchmark convention by using the provided validation/test positive edges and the corresponding negative edge samples when computing Hits@(K) [13]. For undirected graphs, (u, v) and (v, u) represent the same relation, so evaluation is performed consistently with the canonical edge representation used in the split construction.

6.3.3 Negative Sets for Evaluation

Link prediction evaluation is typically framed as ranking a true edge against a set of negatives. Let $(u, v) \in E_{\text{test}}$ be a test positive. We construct a candidate set

$$\mathcal{C}(u, v) = \{v\} \cup \mathcal{N}(u), \quad (6.7)$$

where $\mathcal{N}(u)$ is a set of negative destinations for source u (or analogously negatives for destination-conditioned ranking). A negative candidate $\tilde{v} \in \mathcal{N}(u)$ must satisfy $(u, \tilde{v}) \notin E_{\text{obs}}$ to avoid counting a true-but-unseen (or held-out) edge as negative. Here E_{obs} denotes the set of edges considered *known positives* for filtering; a conservative choice is

$$E_{\text{obs}} = E_{\text{train}} \cup E_{\text{val}} \cup E_{\text{test}}, \quad (6.8)$$

which ensures that no observed edge is treated as negative during evaluation.

Sampled vs. all-candidate evaluation. For small graphs, one can rank against all nodes (all-candidates). For large graphs, this becomes intractable, so we adopt a *sampled evaluation* protocol

where $|\mathcal{N}(u)| = M$ is fixed (e.g., $M \in [50, 1000]$ depending on dataset scale). Using a fixed M yields a controlled compute budget and stable comparisons across methods.

Fixed negatives for reproducibility. To make evaluation comparable across model variants, we use the same validation and test candidate edge sets for all runs. For OGB benchmarks, these candidates are taken from the official validation/test positive and negative edge splits. For synthetic or custom experiments in which negative candidates must be generated, we generate them once using a fixed random seed and a deterministic sampling procedure, and then reuse the resulting candidate sets across model variants.

6.3.4 Ranking Metrics

Following the OGB link-prediction evaluation protocol, we report Hits@ K as the primary ranking metric [13]. Given a query with candidate set $\mathcal{C}(u, v)$, the model produces scores $\{s_\theta(u, c) : c \in \mathcal{C}(u, v)\}$. Let $\text{rank}(v)$ be the rank position of the true destination v when candidates are sorted by descending score (rank 1 is best). We report standard ranking metrics:

- **Hits@ K :** proportion of queries for which $\text{rank}(v) \leq K$.

Hits@ K is typically reported for $K \in \{1, 3, 10, 50\}$ depending on the benchmark.

Filtered evaluation. In many benchmarks, evaluation is *filtered*: when computing the rank of (u, v) , any candidate c such that $(u, c) \in E_{\text{obs}}$ (a known positive edge) is removed from the candidate set, except for the target v . This avoids penalizing the model for ranking another true edge above the specific held-out edge. We follow this filtered protocol by ensuring the negative candidate sets exclude edges in E_{obs} .

6.3.5 Data Leakage Prevention

We enforce the following rules to avoid leakage:

- The training graph used for connectivity-dependent representation learning is G_{train} only.

- Validation/test edges are never added to adjacency structures used for message passing, distance computation, or structural encoding.
- Negative sampling for evaluation is filtered against E_{obs} (train+val+test) to avoid labeling observed positives as negatives.

These constraints ensure that reported performance reflects true generalization to unseen edges rather than inadvertent exposure to held-out connectivity.

Summary. We split edges into disjoint train/validation/test sets, train all structure-dependent components on E_{train} only, and evaluate using filtered, ranking-based metrics (i.e., Hits@ K) over fixed candidate sets. The next subsection details the negative sampling strategy used for training and validation.

6.4 Negative Sampling Strategy

Training a link predictor requires both positive edges and negative, or non-edge, examples. For a simple undirected graph without self-loops, the number of possible node pairs is $\frac{|V|(|V|-1)}{2}$, so the set of candidate non-edges has size $\frac{|V|(|V|-1)}{2} - |E|$. For directed graphs, the analogous candidate set is based on ordered node pairs. Thus, even for sparse graphs, the number of possible negative examples can be much larger than the number of observed positive edges. This quadratic candidate space is a standard issue in link prediction: link-prediction surveys define the task over missing or future links in the set of nonexistent node pairs, and practical GNN-based link-prediction pipelines commonly sample non-existent node pairs as negative examples [14, 95]. The negative sampling strategy is therefore a primary driver of optimization stability, training-example hardness, and the overall throughput of the scalable pipeline.

6.4.1 Sampled Negatives and Filtering

Let $(u, v) \in E_{\text{train}}$ be a positive training edge. For each positive edge, we sample K negative candidate pairs, such as $\{(u, \tilde{v}_k)\}_{k=1}^K$, or alternatively $\{(\tilde{u}_k, v)\}_{k=1}^K$ depending on the sampling

scheme. These sampled pairs are treated as non-edges during training, but they must not collide with observed positive edges.

A sampled negative is retained only if $(u, \tilde{v}) \notin E_{\text{obs}}$, where E_{obs} is the set of edges treated as positives for filtering. During *training*, the minimal safe choice is $E_{\text{obs}} = E_{\text{train}}$, which avoids contradictions in the training labels. During *validation/test evaluation*, we use a stricter filter, such as $E_{\text{obs}} = E_{\text{train}} \cup E_{\text{val}} \cup E_{\text{test}}$ (Section 6.3), so that no observed edge is counted as a negative candidate. For undirected graphs, collision checks are performed on canonicalized pairs to ensure that $(u, \tilde{v})$ and $(\tilde{v}, u)$ are treated consistently.

6.4.2 Uniform and Degree-Based Negative Sampling

The simplest strategy samples negatives uniformly from the node set, $\tilde{v} \sim \text{Unif}(V)$, followed by rejection if $(u, \tilde{v}) \in E_{\text{obs}}$ or if $\tilde{v} = u$ when self-loops are disallowed. Uniform negatives are cheap and unbiased with respect to the node distribution, but they are often *too easy* in highly skewed graphs, leading to fast but potentially suboptimal training.

To increase the hardness of negatives and better match the empirical frequency of nodes, we can instead sample from a degree-based, or popularity-biased, distribution, $\Pr(\tilde{v} = x) \propto (\deg(x))^\alpha$, where $\alpha \in [0, 1]$ controls the strength of the popularity bias. Common choices are $\alpha = 0.5$ or $\alpha = 1$. This produces harder negatives by proposing high-degree nodes more often, which is beneficial when the graph exhibits heavy-tailed degree distributions.

6.4.3 On-the-Fly vs. Precomputed Negatives

For scalability, negative edge examples, i.e., sampled candidate node pairs that are treated as non-edges, can be generated in two ways:

- **On-the-fly sampling:** generate negatives during each batch. This reduces storage and provides stochasticity across epochs, but requires fast membership checks against E_{obs} .

- **Precomputed negatives:** store fixed negatives for each positive. This improves determinism and may simplify distributed evaluation, but can be expensive to store and may overfit to a static negative set.

In our pipeline, we prefer on-the-fly sampling for training, and fixed (seeded) negatives for validation/test evaluation to reduce variance.

6.4.4 Choosing the Negative Ratio ρ_{neg}

The negative ratio ρ_{neg} , reported in later tables, is the ratio of sampled negative edge examples to positive edge examples used during training:

$$\rho_{neg} = \frac{\# \text{ sampled negative training edge examples}}{\# \text{ positive training edge examples}}.$$

For example, $\rho_{neg} = 1$ uses one sampled negative edge example per positive training edge on average, while $\rho_{neg} = 6$ uses six sampled negative edge examples per positive training edge. A value below one, such as $\rho_{neg} = 0.1$, means that fewer negative examples than positive examples are sampled, e.g., approximately one negative example for every ten positive examples.

The negative ratio ρ_{neg} controls both compute cost and supervision strength:

- Larger ρ_{neg} typically improves ranking performance up to a point by providing stronger contrast.
- Larger ρ_{neg} increases per-batch compute and memory, and may require chunked scoring for scalability.

We treat ρ_{neg} as a tunable hyperparameter and keep it fixed across methods when performing controlled comparisons.

Summary. We construct negatives by sampling node pairs that do not appear in the observed edge set, using uniform or degree-biased distributions for efficiency and hardness. Negatives are generated on-the-fly during training (with efficient collision checking) and fixed for evaluation

to reduce variance. The next section describes the model architecture and the scalable training procedure built on top of this sampling pipeline.

6.5 Model and Training Procedure

This section describes the link prediction model used in this chapter and the end-to-end training procedure designed for scalability. The overall design separates **node representation construction** (which may be precomputed and cached) from **edge scoring and optimization** (which is performed in streamed mini-batches with negative sampling).

Figure 6.1 summarizes the full link-prediction training and evaluation pipeline used in this chapter. Starting from the training edge split, graph-coordinate node representations are computed from the training graph, positive training edges are paired with sampled negative edge examples, edge embeddings are scored by the downstream predictor, and the resulting loss is used to update the trainable parameters. Periodically, the trained scorer is evaluated on the validation or test candidate edge sets using the dataset-specific ranking metric.

6.5.1 Node Representations

Each node $u \in V$ is associated with a vector representation $\mathbf{z}_u \in \mathbb{R}^d$ used for scoring candidate edges. The pipeline is agnostic to how $\mathbf{z}_u$ is obtained, but two common cases are:

Graph Coordinate embeddings. We represent each node using anchor-based Graph Coordinates (GCs), computed solely from the training graph $G_{\text{train}} = (V, E_{\text{train}})$ to avoid leakage (Section 6.3). Let $A = \{a_1, \dots, a_m\} \subset V$ be a set of m anchor nodes. The raw distance vector for node u is

$$\mathbf{d}_u = [\text{dist}(u, a_1), \dots, \text{dist}(u, a_m)] \in \mathbb{R}^m, \quad (6.9)$$

where $\text{dist}(\cdot, \cdot)$ denotes shortest-path distance on G_{train} . We evaluate three coordinate variants that differ in how $\mathbf{d}_u$ is processed into the final embedding $\mathbf{z}_u$:

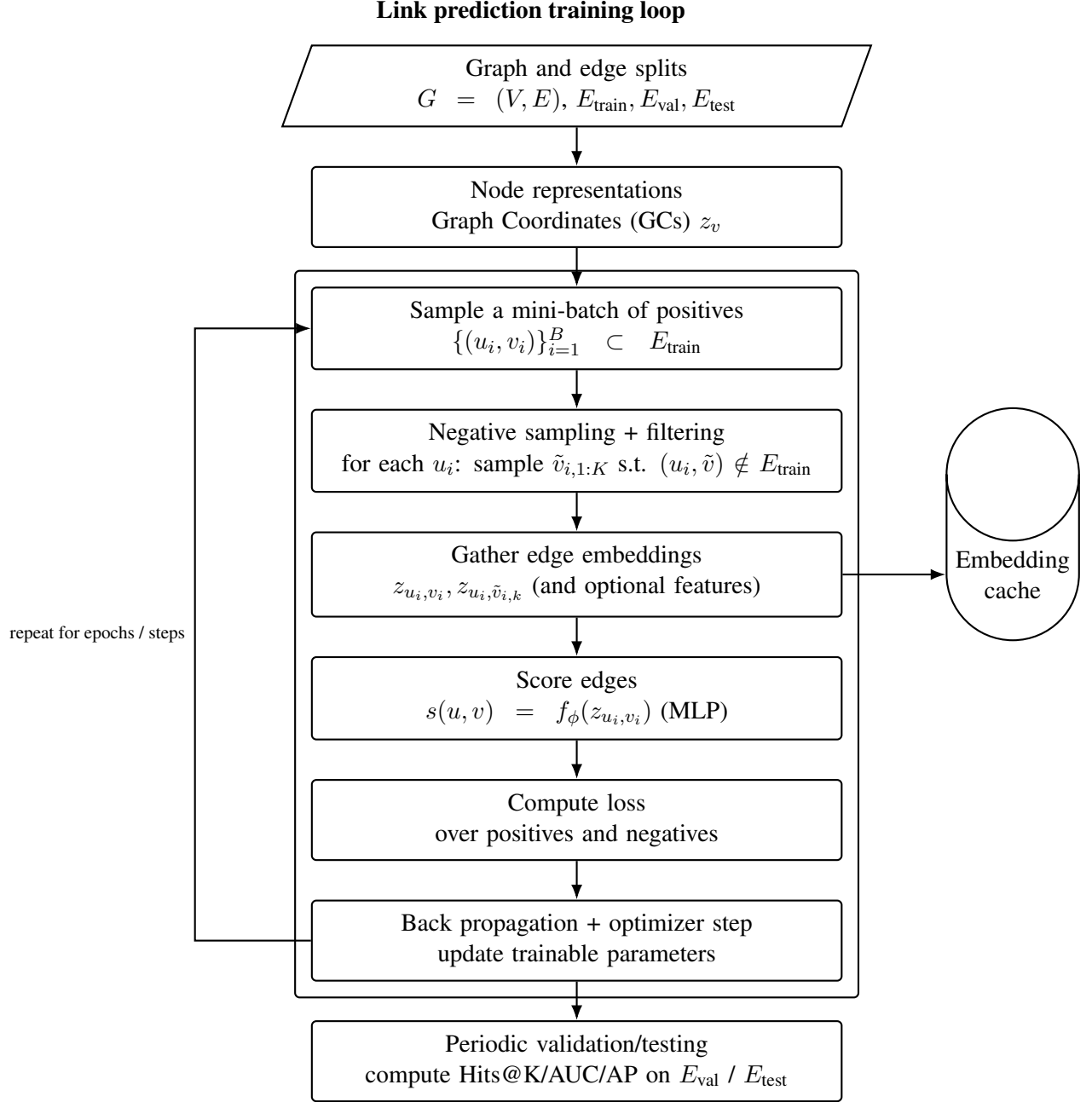


Figure 6.1: Procedure for link prediction with sampled negatives: node representation computing, filtered negative sampling, edge scoring, loss computation, and parameter updates, with periodic evaluation.

- **VC (Virtual Coordinates):** The raw distance vector is used directly as the node embedding:

$$\mathbf{z}_u^{\text{VC}} = \mathbf{d}_u. \quad (6.10)$$

- **TC (Topology Coordinates):** The distance matrix $D \in \mathbb{R}^{n \times m}$ (rows indexed by nodes, columns by anchors) is factored via truncated SVD:

$$D \approx U\Sigma V^\top, \quad (6.11)$$

and each node's embedding is taken as the corresponding row of the low-rank approximation:

$$\mathbf{z}_u^{\text{TC}} = (U\Sigma)_u. \quad (6.12)$$

This transformation decorrelates the anchor distances and compresses the representation into a lower-dimensional space.

- **DVC (Directional Virtual Coordinates):** Directional virtual coordinates are constructed from pairs of anchors. For each selected anchor pair (A_p, A_q) , the corresponding coordinate of node u is computed from the distances from u to the two anchors, producing a virtual direction in the graph. This construction avoids the SVD step used by TC while still deriving node coordinates from anchor-based shortest-path distances.

All three variants are precomputed once before training and cached; the downstream scoring MLP operates on fixed embeddings.

6.5.2 Edge Scoring Functions

Given node representations $\mathbf{z}_u$ and $\mathbf{z}_v$, the model outputs a score $s_\theta(u, v)$.

Dot-product scorer. A simple and efficient choice is

$$s_\theta(u, v) = \langle \mathbf{z}_u, \mathbf{z}_v \rangle, \quad (6.13)$$

optionally with learned linear projections for directed graphs:

$$s_\theta(u, v) = \langle \mathbf{W}_s \mathbf{z}_u, \mathbf{W}_t \mathbf{z}_v \rangle. \quad (6.14)$$

MLP scorer with pairwise features. To capture the relational structure between two node embeddings, we form a pairwise feature vector and pass it through an MLP. We consider two combination strategies:

- **Concat:** The two embeddings are concatenated in both orders to preserve directional information:

$$\mathbf{e}_{uv}^{\text{concat}} = [\mathbf{z}_u \parallel \mathbf{z}_v \parallel \mathbf{z}_v \parallel \mathbf{z}_u]. \quad (6.15)$$

- **SumAndDiff:** The sum and absolute difference are concatenated, encoding both the shared structure and the pointwise discrepancy between the two embeddings:

$$\mathbf{e}_{uv}^{\text{sad}} = [(\mathbf{z}_u + \mathbf{z}_v) \parallel |\mathbf{z}_u - \mathbf{z}_v|]. \quad (6.16)$$

This construction is permutation-invariant (symmetric in u and v), which is appropriate for undirected graphs, and retains complementary information that a simple concatenation may lose.

The edge score is then:

$$s_\theta(u, v) = g_\theta(\mathbf{e}_{uv}), \quad (6.17)$$

where g_θ is a multi-layer perceptron. SumAndDiff is the default combination in our experiments; its advantage over Concat is examined in Section 6.7.

6.5.3 Loss Function with Negative Sampling

We train with sampled negatives (Section 6.4). For each positive edge $(u, v) \in E_{\text{train}}$, sample N_{neg} negatives $\{(u, \tilde{v}_k)\}$ (or destination-conditioned variants). We use a sampled binary cross-

entropy objective:

$$\mathcal{L}(u, v) = -\log \sigma(s_\theta(u, v)) - \frac{1}{N_{neg}} \sum_{k=1}^{N_{neg}} \log \sigma(-s_\theta(u, \tilde{v}_k)), \quad (6.18)$$

and minimize the average over a mini-batch of positives.

Class imbalance and weighting. When N_{neg} is large, the effective negative mass can dominate gradients. If needed, we apply a weight λ to the negative term:

$$\mathcal{L}(u, v) = -\log \sigma(s_\theta(u, v)) - \frac{\lambda}{N_{neg}} \sum_{k=1}^{N_{neg}} \log \sigma(-s_\theta(u, \tilde{v}_k)), \quad (6.19)$$

where λ can be tuned to stabilize optimization.

6.5.4 Mini-batch Construction and Vectorized Scoring

Let a mini-batch contain B positive edges $\{(u_i, v_i)\}_{i=1}^B$ and N_{neg} negatives per positive. We construct tensors:

$$\mathbf{U} = [u_1, \dots, u_B], \quad \mathbf{V} = [v_1, \dots, v_B], \quad \tilde{\mathbf{V}} \in \mathbb{R}^{B \times N_{neg}}, \quad (6.20)$$

and gather the corresponding embeddings into matrices

$$\mathbf{Z}_U \in \mathbb{R}^{B \times d}, \quad \mathbf{Z}_V \in \mathbb{R}^{B \times d}, \quad \mathbf{Z}_{\tilde{V}} \in \mathbb{R}^{B \times N_{neg} \times d}. \quad (6.21)$$

For dot-product scoring, positives are computed as row-wise inner products:

$$\mathbf{s}_i^+ = \langle \mathbf{Z}_U[i], \mathbf{Z}_V[i] \rangle, \quad (6.22)$$

and negatives are computed in a batched form:

$$\mathbf{s}_{i,k}^- = \langle \mathbf{Z}_U[i], \mathbf{Z}_{\tilde{V}}[i, k] \rangle. \quad (6.23)$$

This keeps training throughput high by avoiding per-edge Python loops and relying on dense GEMM-like kernels.

6.5.5 Optimization and Training Schedule

We optimize parameters θ (and any encoder parameters ϕ) using a first-order optimizer such as Adam/AdamW with standard regularization:

- **Weight decay:** applied to encoder and scorer weights.
- **Gradient clipping:** optional, to control rare large updates when hard negatives are present.
- **Learning rate schedule:** optional warmup followed by cosine decay or step decay, depending on stability.

Mixed precision and gradient accumulation. To improve throughput, we optionally use mixed precision for scoring and backpropagation. When the desired effective batch size exceeds GPU memory, we use gradient accumulation over A micro-batches (each of size B), yielding an effective batch size $A \cdot B$ without increasing peak memory.

6.5.6 Scalable Training Pipeline

The complete training step is organized to minimize data movement and synchronize points:

1. Sample a mini-batch of positive edges from E_{train} .
2. Generate N_{neg} negatives per positive on-the-fly with collision filtering against E_{train} .
3. Gather the required node embeddings $\mathbf{z}_u$, $\mathbf{z}_v$, and $\mathbf{z}_{\tilde{v}}$ (CPU→GPU as needed).
4. Compute scores s_θ in a fully vectorized manner (chunking negatives if necessary).

Algorithm 6.1 Scalable Link Prediction Training with Sampled Negatives

Require: Training edges E_{train} , node embeddings (precomputed or encoder), negative ratio ρ_{neg} , batch size B

```
1: for each epoch do
2:   for each mini-batch of positives  $\{(u_i, v_i)\}_{i=1}^B$  from  $E_{\text{train}}$  do
3:     Sample negatives  $\tilde{v}_{i,1:\rho_{\text{neg}}}$  with filtering:  $(u_i, \tilde{v}_{i,k}) \notin E_{\text{train}}$ 
4:     Gather embeddings  $\mathbf{z}_{u_i}, \mathbf{z}_{v_i}, \mathbf{z}_{\tilde{v}_{i,k}}$  (chunk if needed)
5:     Compute scores  $s_i^+ = s_\theta(u_i, v_i)$  and  $s_{i,k}^- = s_\theta(u_i, \tilde{v}_{i,k})$ 
6:     Compute loss  $\mathcal{L}$  (e.g., Eq. 6.18); backpropagate
7:     Update parameters with optimizer step (optionally with mixed precision / accumulation)
8:   end for
9: end for
```

5. Compute loss (Eq. 6.18), backpropagate, and update parameters.

For reproducibility, we fix the RNG seed for (i) shuffling positive edges and (ii) the negative sampler. For evaluation, we additionally use fixed candidate sets (Section 6.3) to reduce variance across runs.

6.5.7 Algorithmic Summary

Algorithm 6.1 summarizes a scalable training loop for link prediction with sampled negatives.

Summary. The model learns a scoring function over node representations, trained via mini-batches of positive edges and on-the-fly sampled negatives. Scalability is achieved through streaming data construction, efficient embedding gathers, vectorized scoring, and optional chunking/mixed precision to bound memory usage.

6.6 Experimental Setup

This section describes the datasets, baselines, implementation details, and hyperparameter settings used to evaluate the scalable link prediction pipeline. The goal is to make the experiments reproducible and to ensure that comparisons isolate the impact of the proposed training pipeline and representation choices.

Table 6.1: Dataset statistics.

Dataset	$ V $	$ E $	Metric	Features
Synthetic (line/circle)	20–10k	varies	Hits@1	none (structure only)
OGBL-DDI	4,267	1,334,889	Hits@20	none
OGBL-PPA	576,289	~42M	Hits@100	none

6.6.1 Datasets

We evaluate on three dataset families covering controlled synthetic settings and real-world benchmarks.

Synthetic graphs. To validate the approach under fully controlled conditions and isolate the effect of individual design choices, we generate synthetic graphs with known edge structure. Two topologies are used: **line** graphs (nodes arranged in a path) and **circle** graphs (nodes arranged in a cycle). Both are instantiated at five scales: $|V| \in \{20, 100, 500, 1000, 10000\}$. All edges are included in the training graph; a held-out subset of edges (and an equal number of non-edges) forms the test set. The evaluation metric is **Hits@1**.

OGBL-DDI. A drug–drug interaction network from the OGB benchmark [13]. The graph contains 4,267 nodes and 1,334,889 edges (unweighted, undirected). No node features are provided; all representations are structure-derived. Edges are split according to the standard OGB protocol: approximately 80% for training and 10% each for validation and test. The evaluation metric is **Hits@20**.

OGBL-PPA. A protein–protein association network from OGB, with 576,289 nodes and approximately 42 million edges. The graph is nearly connected (the largest component contains 99.96% of nodes). No node features are available. Due to scale, training is substantially more expensive than on OGBL-DDI. The evaluation metric is **Hits@100**.

Table 6.1 summarizes key statistics.

6.6.2 Baselines

We compare the proposed scalable pipeline against representative link prediction baselines spanning simple similarity scoring and learned embedding approaches. The specific baseline set can be adapted to the datasets used, but a typical configuration includes:

- **Heuristic scorers:** Common Neighbors, Adamic–Adar, Resource Allocation (for undirected graphs).
- **Matrix factorization / embeddings:** shallow embedding models with dot-product scoring (e.g., logistic factorization).
- **GNN-based methods (if applicable):** neighborhood-aggregation encoders trained on G_{train} , followed by a dot-product or MLP edge scorer.

All learning-based baselines use the same split protocol and negative sampling constraints to ensure comparability.

6.6.3 Training and Evaluation Protocol

We follow the split and evaluation procedure in Section 6.3. In particular:

- **Training graph:** all connectivity-dependent computations use E_{train} only.
- **Negatives:** training uses on-the-fly negatives with collision filtering against E_{train} ; validation/test use fixed candidate sets for reproducibility.
- **Metrics:** we report Hits@ K .

When sampled evaluation is used, we fix the number of negatives per query and store the random seed and sampling procedure to ensure identical candidate sets across runs.

6.6.4 Implementation Details

All methods are implemented in a unified training codebase to control for engineering differences. Key implementation details include:

- **Batching:** positives are streamed as edge-index batches of size B ; negatives are generated per batch with ratio ρ_{neg} .
- **Embedding storage:** node representations are stored on CPU with batched index-gather to GPU; optional GPU caching is used when memory permits.
- **Chunked scoring:** negative scores are computed in chunks when $B \times N_{neg}$ exceeds GPU memory limits.
- **Precision:** mixed precision is enabled when stable; otherwise training defaults to full precision.

We set random seeds for data shuffling and negative sampling. Each experiment is repeated across multiple seeds when feasible to estimate variance; otherwise, we report a single deterministic run with fixed evaluation negatives.

6.6.5 Hyperparameters

We tune hyperparameters on the validation set and report test performance using the best validation checkpoint. Unless otherwise stated, the search space includes:

- **Embedding dimension:** $d \in \{64, 128, 256\}$.
- **Negative ratio:** $\rho_{neg} \in \{0.1, 1, 6\}$.
- **Batch size:** B chosen to saturate GPU memory (with gradient accumulation when needed).
- **Optimizer:** Adam/AdamW with learning rate $\eta \in [10^{-4}, 10^{-2}]$ and weight decay $\in \{0, 10^{-5}, 10^{-4}, 10^{-3}\}$.
- **Training epochs:** up to a fixed budget, with early stopping based on validation Hits@K.

Summary. This section specifies the datasets, baselines, and training/evaluation configuration used throughout the chapter. The next sections report results (Section 6.7) and analyze typical error modes and limitations (Section 6.8).

Table 6.2: Synthetic graph results (testHits@1). “w/ transf.” denotes runs using TC or DVC embeddings; “VC only” denotes runs using raw VC. Each entry is a mean over all matching configurations.

Graph	$ V $	w/ transf.	VC only	SumAndDiff [†]	Concat [†]
circle	20	1.000	1.000	1.000	1.000
	100	1.000	0.696	0.870	0.588
	500	0.996	0.808	0.717	0.945
	1,000	1.000	0.456	1.000	0.239
	10,000	0.380	0.128	0.234	0.023
line	20	1.000	1.000	1.000	1.000
	100	1.000	0.919	1.000	0.776
	500	1.000	0.772	0.990	0.391
	1,000	0.945	0.432	1.000	0.107
	10,000	0.246	0.133	0.225	0.072

[†] SumAndDiff and Concat columns report VC-only (noTransf) condition means.

6.7 Results

This section reports link prediction performance under the split and evaluation protocol described in Section 6.3. We report predictive performance measured by ranking metrics (Hits@ K).

6.7.1 Synthetic Graph Results

Table 6.2 reports testHits@1 on line and circle graphs across scales, comparing VC (raw coordinates, no transformation) against the best-performing GC variant (TC or DVC) and the two combination strategies (Concat vs. SumAndDiff). Each cell is the mean over all runs sharing that configuration on the given dataset. This confirms that the coordinate-based scorer can recover clear structural link patterns when the underlying topology is simple. The lower performance of raw VC and the degradation at 10,000 nodes suggest that coordinate transformation and anchor coverage become more important as graph size increases. These results validate the basic pipeline before moving to more complex real-world benchmarks.

Table 6.3: OGBL-DDI results (testHits@20). ρ_{neg} denotes the ratio of negative to positive training samples.

GC	Anchors	Combine	Hidden	ρ_{neg}	Hits@20
TC	4267	SumAndDiff	[128]	1	0.2523
TC	4267	Avg	[128]	1	0.2448
VC	400	SumAndDiff	[128]	1	0.1884
VC	800	Concat	[128]	1	0.1693
TC	400	Concat	[128]	0.1	0.1125
TC	400	SumAndDiff	[128]	0.1	0.1198
TC	4267	SumAndDiff	[256,128]	1	0.0268
TC	1000	SumAndDiff	[256,128]	6	<0.001

6.7.2 Real-World Benchmark Results

Table 6.3 reports results on OGBL-DDI. Each row corresponds to a distinct configuration; we report the best validation-selected checkpoint. Among these single-channel configurations, the strongest result is obtained by TC with all 4267 nodes used as anchors and the SumAndDiff edge representation. The lower performance of smaller-anchor or raw-VC configurations suggests that richer coordinate information is important on OGBL-DDI. The results also show sensitivity to the training negative ratio: too few negatives provide limited contrast, while the large negative-ratio setting collapses in this experiment. These observations motivate the coordinate-ensemble experiments in Section 6.7.3, where multiple complementary coordinate views are fused instead of relying on a single scorer.

6.7.3 No-Leak Coordinate Ensembles for OGBL-DDI Link Prediction

This subsection reports the final link-prediction results obtained on the full OGBL-DDI benchmark using graph-coordinate edge scorers. The main ensemble results are summarized in Table 6.4. The purpose is to test whether anchor-derived node coordinates, originally used as node-level representations, can also support high-quality edge ranking when converted into pairwise edge features. The main finding is that single coordinate channels are not sufficient for strong Hits@20 on this benchmark, but late fusion of complementary coordinate views is substantially more effective. Under the no-leak protocol described below, the best validation-selected 8-channel coordinate

ensemble reaches 0.8564 test Hits@20. A compact 10.23M-parameter ensemble reaches 0.8095 test Hits@20, and a compressed ensemble with fewer than one million trainable parameters still reaches 0.7599 test Hits@20.

No-leak split protocol. All formal results in this subsection use a split-aware no-leak protocol. The training graph is built only from training positive edges, and the training negative sampler forbids only training positive edges:

$$E_{\text{graph}} = E_{\text{train}}^+, \quad E_{\text{forbid}} = E_{\text{train}}^+.$$

Validation and test positive edges are treated as unknown during training. They are not inserted into the training graph, and they are not inserted into the training-negative forbidden set. The validation and test positive splits each contain 133,489 edges. Evaluation uses the official OGB link-prediction negative edges and reports Hits@K values, with AUC recorded only as an auxiliary metric.

This distinction is important because an earlier diagnostic protocol forbade all positive edges, including validation and test positives, when sampling training negatives. That setting leaks split information into training and is therefore excluded from the formal claims in this subsection.

Coordinate channels. Each ensemble channel first constructs a node-coordinate matrix from the training graph using one or more graph-coordinate families. The final ensembles combine the following coordinate views:

- **TC:** topology coordinates obtained from the train-positive graph.
- **VC:** normalized virtual coordinates
- **DVC:** directional virtual coordinates, which provide a different coordinate geometry from TC and VC

- **RAW**: raw shortest-path-to-anchor distance columns (VCs) computed on the train-positive graph

Each channel chooses an anchor budget and a coordinate-column budget. Most final channels use random anchors. The compressed under-1M ensemble additionally uses one degree-stratified TC+VC channel, which improves complementarity without increasing the total parameter budget above one million. The term `degree-stratified` refers only to anchor selection. Given A requested anchors, the implementation computes node degrees in the train-positive graph, stably sorts the candidate nodes by degree, splits this ordered list into A bins, and samples one anchor uniformly from each bin using the run seed.

Edge features and single-channel scoring. Given a node-coordinate representation $z_m(v)$ for channel m , each candidate edge (u, v) is represented by the symmetric edge feature

$$\phi_m(u, v) = [z_m(u) + z_m(v), |z_m(u) - z_m(v)|].$$

This `sum_and_diff` representation is used by all final reported models. Before training, edge features are normalized using statistics computed from the training edge set. A dense MLP then maps $\phi_m(u, v)$ to a logit score $s_m(u, v)$. Single-channel training uses binary cross entropy with logits, Adam with learning rate 10^{-3} , weight decay 10^{-4} , batch size 4096, evaluation batch size 16384, and early stopping by validation Hits@20.

Training negatives. The final channels use mixed static negative sampling. Most high-score and under-1M mix channels use a train negative ratio of 7.0, with the negative sampler mixing uniform, two-hop, and high-common-neighbor negatives with fractions 0.3, 0.4, and 0.3, respectively.

Train-calibrated late score ensemble. The strongest results are obtained by late fusion of saved single-channel models. For each channel m , logits are evaluated on a train-sampled calibration set containing training positives and sampled training negatives. Each channel’s logits are then z-score

Table 6.4: Main no-leak coordinate-ensemble results on full OGBL-DDI.

Model	Params	Valid H@20	Test H@20	Test H@50	Test H@100	Test AUC
8-channel z-score ensemble	32,001,800	0.7057	0.8564	0.9101	0.9411	0.9989
10.23M compact ensemble	10,230,787	0.6933	0.8095	0.8751	0.9135	0.9988
Under-1M ensemble	937,410	0.6680	0.7599	0.8367	0.8781	0.9982

normalized using calibration statistics:

$$\tilde{s}_m(u, v) = \frac{s_m(u, v) - \mu_m}{\sigma_m}.$$

The ensemble score is a weighted linear sum,

$$S(u, v) = \sum_{m=1}^M w_m \tilde{s}_m(u, v).$$

Weights are selected by validation Hits@20, not by test Hits@20. No final ensemble reported here uses explicit structural-score channels such as common neighbors, Adamic-Adar, or resource allocation; all reported active channels are learned coordinate scorers. The best ensemble is obtained by re-scoring previously trained no-leak model runs, applying the same train-calibrated normalization, enumerating model-channel subsets, and using coordinate descent over a small nonnegative weight grid to maximize validation Hits@20.

Table 6.4 summarizes the main results. The 8-channel z-score ensemble is the strongest result under the split-aware protocol in this set of experiments. The under-1M ensemble is the strongest saved compressed variant: it is 34.14 times smaller than the 8-channel ensemble by active parameter count, while retaining 88.7% of the 8-channel ensemble’s test Hits@20. In absolute terms, compression costs 0.0965 Hits@20 while reducing the trainable parameter count from 32.00M to 0.94M.

Table 6.5: Active channels in the 8-channel z-score ensemble. All rows use the no-leak protocol with the training positives as the training-negative forbidden set.

Channel	Hidden dims	Params	Val H@20	Test H@20	Test H@100	Weight
DVC+RAW, 1024/512	2048/1024/512	6,819,841	0.6643	0.1574	0.8773	3.50
TC+DVC, 1024/512	1024/512/256	2,754,561	0.6264	0.0967	0.6624	1.00
TC+VC+DVC+RAW, 512/256	384/128	836,225	0.6006	0.2269	0.7465	3.00
TC+VC+DVC+RAW, student	384/128	836,225	0.5311	0.2430	0.7359	0.50
DVC+RAW, 512/256	256/128	295,425	0.5812	0.1208	0.6086	1.00
VC+DVC, 1024/512	2048/1024/512	6,819,841	0.6180	0.1022	0.8294	0.25
VC+RAW, 1024/512	2048/1024/512	6,819,841	0.5841	0.1157	0.7573	0.75
TC+RAW, 1024/512	2048/1024/512	6,819,841	0.6262	0.0741	0.7264	3.00

The ensemble in Table 6.5 combines eight active channels selected from a larger candidate set. The row marked as `student` is compact MLP trained by distilling a previously trained no-leak teacher ensemble. The student has its own TC+VC+DVC+RAW coordinates with 512 anchors and 256 retained coordinates per view, giving a 1024-dimensional node coordinate and a 2048-dimensional `sum_and_diff` edge input. It is trained on 300,000 train-positive edges and 300,000 train-negative edges, all sampled from the training graph with validation and test edges excluded. The selected active weights are [3.5, 1.0, 3.0, 0.5, 1.0, 0.25, 0.75, 3.0]. Although no individual channel has test Hits@20 above 0.2430, their validation-selected combination reaches 0.8564. The largest weights fall on a high-capacity DVC+RAW channel, a compact all-view TC+VC+DVC+RAW channel, and a high-capacity TC+RAW channel. This indicates that the gain comes from complementary coordinate views rather than from one dominant scorer.

The compressed ensemble in Table 6.6 uses four small models and the weights listed in the final column. Its strength is not single-channel performance: all four channels have individual

Table 6.6: Channels in the best under-1M coordinate ensemble. The total parameter count is 937,410.

Channel	Hidden dims	Params	Val H@20	Test H@20	Test H@100	Weight
TC+VC, degree-stratified 512/256	256/128	295,425	0.5597	0.1088	0.6113	4.00
DVC+RAW, random 512/256	256/128	295,425	0.5812	0.1208	0.6086	4.00
TC+DVC, random 512/256	192/64	209,217	0.5835	0.1247	0.7093	4.00
TC+VC, random 512/256	126/64	137,343	0.5615	0.0970	0.5416	0.25

test Hits@20 between 0.0970 and 0.1247. The gain instead comes from error complementarity. Different coordinate families rank the hard-negative tail differently, so the normalized weighted sum moves true positives upward more reliably than any individual channel.

Interpretation. These results refine the earlier single-channel observation for real-world link prediction. Single graph-coordinate scorers are limited on OGBL-DDI, but the coordinate representation family contains enough complementary structural information to support strong ranking once multiple coordinate views are fused. AUC is near 0.998 for the main ensembles, which indicates that most positive/negative comparisons are easy. Hits@20 is much harder because it is dominated by a small set of high-scoring, structurally plausible negative edges. For example, if one true edge is scored above 99,970 out of 100,000 negative candidates but 30 structurally plausible negatives still receive higher scores, the AUC contribution is very strong, but the edge is ranked only 31st and therefore fails Hits@20. This illustrates why a model can have a very high AUC while still having lower Hits@20: the metric is sensitive to the hardest high-scoring negatives near the top of the ranked list. Late fusion succeeds because the hard-negative tail is not identical across coordinate families and anchor strategies. Validation-selected z-score fusion therefore improves the top of the ranking without introducing validation or test positives into training.

The main conclusion is that the limitation of coordinate-based link prediction on OGBL-DDI is not the absence of useful structural information in the coordinates. Rather, the limitation is that one coordinate channel gives an incomplete view of the edge-ranking problem. Coordinate ensembles reduce this channel-specific weakness and provide a stronger no-leak link-prediction result. At the same time, these results should be interpreted as evidence for OGBL-DDI; the larger OGBL-PPA setting remains unresolved under the current training pipeline and requires separate scalability and optimization work.

6.7.4 Discussion of Observed Trends

The results suggest three main observations. First, the synthetic results in Table 6.2 show that the graph-coordinate pipeline behaves as expected in controlled structural settings. On both line and circle graphs, the transformed coordinate representations, TC or DVC, achieve nearly perfect Hits@1 for graph sizes up to 1,000 nodes, except for a modest drop on the 500-node circle graph. This indicates that when edge existence follows a simple geometric rule in the graph, anchor-derived coordinates can preserve enough information for a downstream scorer to distinguish true edges from sampled non-edges. At the same time, the 10,000-node results show a clear degradation, with transformed-coordinate performance falling to 0.380 on the circle graph and 0.246 on the line graph. This suggests that the same pipeline becomes sensitive to graph scale and anchor coverage: as the graph becomes larger, a fixed or limited anchor budget gives a coarser view of the topology, and nearby true edges are harder to separate from structurally similar negative pairs.

Second, the initial real-world results in Table 6.3 show that single-channel coordinate scorers are much more limited on OGBL-DDI. The strongest single-channel configuration uses TC with all 4267 nodes as anchors and the SumAndDiff edge representation, reaching 0.2523 Hits@20. Smaller-anchor settings and raw VC configurations are weaker, suggesting that richer coordinate information is important for this dataset. The results also show sensitivity to the training negative ratio: low negative ratios provide limited contrast, while the large negative-ratio setting collapses in this experiment.

Third, the ensemble results in Table 6.4 change the interpretation of the OGBL-DDI results. Although individual coordinate channels are limited, late fusion of multiple coordinate views substantially improves ranking performance. This suggests that graph coordinates do contain useful structural information for OGBL-DDI, but a single coordinate view captures only part of the edge-ranking signal. The active-channel results in Tables 6.5 and 6.6 further support this interpretation: the ensemble gain comes from combining complementary coordinate families rather than relying on one dominant scorer.

The larger OGBL-PPA benchmark remains unresolved under the current pipeline. In that setting, the interaction between graph scale, anchor coverage, embedding dimensionality, and training stability appears to be more difficult than in OGBL-DDI. Overall, the experiments indicate that graph-coordinate link prediction is reliable in controlled structural settings, can be strong on OGBL-DDI when complementary views are fused, and requires additional scalability and optimization work for larger real-world graphs.

6.8 Error Analysis and Discussion

This section analyzes common failure modes observed in scalable link prediction pipelines and discusses how they relate to data construction, negative sampling, model capacity, and system constraints. The goal is to interpret performance beyond aggregate metrics and to identify actionable directions for improvement.

6.8.1 Error Taxonomy

We group link prediction errors into four broad categories.

(1) Structural ambiguity (indistinguishable candidates). In sparse graphs, multiple candidate nodes may be structurally similar with respect to the source node u under the training graph G_{train} . When the representation $\mathbf{z}_u$ primarily encodes coarse structural proximity (e.g., via neighborhood

aggregation or distance-to-anchors), distinct candidates can collapse to similar scores:

$$s_{\theta}(u, v_1) \approx s_{\theta}(u, v_2), \quad v_1 \neq v_2, \quad (6.24)$$

leading to unstable rankings under small perturbations (seed changes, sampling noise). This is especially common when evaluation uses large candidate sets and the model lacks fine-grained attributes to disambiguate nodes.

(2) Hub dominance and the Hits@ K ceiling (OGBL-DDI). The OGBL-DDI graph has a heavily skewed degree distribution: some nodes have degree $> 2,000$, connecting to more than 50% of all other nodes. Hits@ K evaluation requires the true target to rank above all sampled negatives for a query. For high-degree nodes, the pool of valid negatives is enormous, making it structurally difficult for any score to place a specific true edge in the Top- K , regardless of embedding quality. This is illustrated concretely by a non-NN distance-threshold classifier on VC embeddings: it achieves 63% overall binary accuracy but a Hits@20 of exactly 0, because even correctly classified positives cannot rank above the large volume of negatives for hub-adjacent queries.

(3) Train/test distribution mismatch (OGBL-DDI). The OGBL-DDI split assigns edges to train/validation/test randomly, without ensuring node-level disjointness. As a result, some nodes appearing predominantly in the test set receive weaker VC embeddings, since their structural neighborhood in G_{train} is underrepresented. Analysis of embedding differences across anchor counts shows that the ratio of positive-to-negative sample dissimilarity stabilizes at approximately 85–88% regardless of anchor count, indicating a hard ceiling on discriminability that cannot be overcome by increasing m alone.

(4) Anchor scale vs. training stability (OGBL-PPA). On OGBL-PPA (576,289 nodes), sufficient embedding coverage requires large anchor counts ($m \geq 1,000$), but high-dimensional inputs destabilize training: gradient norms explode and the model degenerates to predicting a constant

class. At $m = 100$ anchors, training can converge on small held-out sets; at $m = 5,000$ anchors, training fails consistently regardless of activation function (ReLU, Leaky ReLU), gradient clipping, learning rate, or optimizer. These results suggest that the current training setup does not resolve the interaction between anchor-based embedding dimension, graph scale, and link-prediction optimization. Further experiments with alternative losses, sampling strategies, and edge-scoring architectures would be needed before attributing the failure to the representation alone.

6.8.2 Diagnostics and What to Log

To make error analysis concrete, we recommend logging the following per-epoch or per-evaluation-run diagnostics:

- **Rank histograms:** distribution of $\text{rank}(v)$ for test edges, highlighting tail behavior.
- **Score margins:** $s_\theta(u, v) - \max_{c \in \mathcal{N}(u)} s_\theta(u, c)$ for each query, to identify near-ties.
- **Hub breakdown:** metrics stratified by degree bins of source and target nodes (e.g., low/medium/high degree).
- **Collision rate:** fraction of proposed negatives rejected due to membership in E_{obs} ; high rates indicate inefficient sampling and potential bias.

These diagnostics help distinguish modeling limitations from pipeline artifacts.

6.8.3 Qualitative Error Patterns

Synthetic graphs: failures concentrate at large scale and low anchor count. On 10,000-node graphs, errors are not evenly distributed across configurations. Runs with fewer anchors ($m = 10$) fail substantially more often than those with $m = 20$ (e.g., Hits@1 of 0.17 vs. 0.34 on line-10000).

OGBL-DDI: collapse under large negative ratio. When the negative training ratio is increased to $\rho_{\text{neg}} = 6$, all tested configurations collapse to Hits@20 < 0.001 . Inspection of training curves shows that loss decreases but the model ceases to rank positives above negatives: the effective

gradient signal is dominated by the large negative mass, and the model assigns uniformly low scores to all pairs. This is consistent with the known instability of BCE-based training under high negative ratios without loss reweighting.

Pipeline validation on 1D synthetic data. To verify that the training loop itself is not the source of failure on real benchmarks, we tested the same pipeline on a fully controlled 1D signal classification task: positive samples are phase-shifted sinusoidal signals (with Gaussian noise) and negatives are random signals of matched length. The model converges to 100% test accuracy across all configurations, confirming that the training loop, loss, and evaluation logic are correct and that observed failures on OGBL-DDI and OGBL-PPA are attributable to dataset-specific structural factors rather than implementation errors.

Cold-start nodes (low-degree sources). Edges incident to low-degree nodes are difficult because the model has limited structural context for those nodes. If node attributes are weak or absent, structural encoders tend to underfit in this regime. Increasing anchor coverage, adding feature-based encoders, or using augmentation (e.g., label smoothing / dropout on neighborhoods) can help.

6.8.4 Discussion: Trade-offs and Practical Recommendations

We highlight several practical trade-offs revealed by error analysis.

Negative hardness vs. stability. Harder negatives (degree-biased or mined) tend to improve top- K ranking but can destabilize training if introduced too early. A robust strategy is a curriculum: begin with uniform or moderately degree-biased negatives, then increase hardness once the model separates easy negatives reliably.

Scalability vs. evaluation fidelity. Sampled evaluation is computationally feasible, but its difficulty depends heavily on how negatives are drawn. Reporting both (i) sampled metrics and (ii) a

limited all-candidates evaluation on a subset of queries (when feasible) can better reflect real-world retrieval behavior.

Pipeline reproducibility. For large-scale experiments, controlling randomness is essential. We recommend fixed evaluation candidate sets, fixed seeds for negative sampling, and logging the exact split files and sampler configuration. When nondeterministic GPU operations are unavoidable, multiple-seed reporting provides a more reliable picture than a single run.

Summary. Link prediction errors typically arise from structural ambiguity, hub dominance, train–test negative distribution shift, and pipeline artifacts such as false negatives or nondeterminism. Careful diagnostics (rank distributions, degree-stratified metrics, collision rates) help isolate these effects and guide improvements in both modeling and scalable training design.

Chapter 7

Train-on-One and Train-on-Many Transfer Across Anchor Sets and Graphs

This chapter investigates whether anchor-based structural representations can be reused across different anchor-induced views, subgraphs, and related graph datasets. The focus is on train-on-one, evaluate-on-many and train-on-many, evaluate-on-one settings, with particular attention to representation stability, alignment, and the limits of cross-view or cross-graph transfer.

7.1 Chapter Overview

This chapter investigates two closely related transfer settings that arise naturally in distance-based and anchor-based graph representation learning: *train-on-one*, *evaluate-on-many* and *train-on-many*, *evaluate-on-one*. In the first setting, a predictor is trained on a single reference graph or reference anchor-induced view and then reused on multiple target views or graphs without retraining. In the second setting, a predictor is trained jointly on several source graphs and then evaluated on a held-out target graph. Together, these settings let us test whether the learned representation captures reusable structure rather than overfitting to one particular graph/view realization.

The chapter is organized around three objectives. First, we formalize the evaluation protocol, including how anchor sets are generated, how representations are constructed per anchor set, and how transfer is measured across views and across graphs. Second, we analyze the main sources of variability that make this setting non-trivial, including anchor coverage effects, representation non-identifiability (e.g., sign flips and subspace rotations in PCA/SVD-based embeddings), node-feature dominance, and distribution shift across graphs. Third, we compare practical mitigation strategies that may improve reuse without full retraining, including Procrustes-style alignment, feature-only versus feature-plus-distance inputs, and multi-source training.

Empirically, this chapter reports real-graph results in both single-source and multi-source transfer settings. The primary cross-dataset real-graph experiments use the cross-attention model introduced later in the chapter. We report both train-on-one and train-on-many transfer results, summarize when alignment helps or hurts, and show that high-dimensional node attributes can dominate transfer behavior in some datasets. Synthetic graph generation remains an important extension, but finalized synthetic Train/Val/Test results are not included in this chapter.

7.2 Motivation and Use Cases

7.2.1 Why Train-on-One, Evaluate-on-Many?

Anchor-based representations introduce an implicit design choice: the learned features depend on the selected anchor set(s). In many pipelines, anchors are sampled randomly or selected via heuristics (e.g., degree-based choices), which means that two runs on the same graph can yield different representations even when the downstream task and split remain unchanged. If the downstream predictor must be retrained for every anchor set, then the final performance becomes tightly coupled to a particular anchor realization, and repeated training is required to obtain stable estimates.

The *train-on-one, evaluate-on-many* setting isolates a different question: can a predictor trained on one anchor-induced view of a graph generalize to other anchor-induced views without retraining? This question is closely aligned with the goal of representation stability. If a method produces features that preserve task-relevant information consistently across anchor choices, then a single trained predictor should remain effective under alternative anchor sets. Conversely, if performance collapses when anchors change, then the representation is likely encoding anchor-specific artifacts rather than stable structural signals.

Beyond anchor variability, train-on-one evaluation is also useful as a stress test for generalization across *graph instances*. For example, one may train on a canonical graph (or one snapshot) and evaluate on perturbed graphs, subgraphs, or synthetic graphs drawn from the same generator

family. This emphasizes whether the representation and predictor capture transferable patterns rather than overfitting to a particular graph realization.

7.2.2 Robustness to Anchor Choice and Graph Variations

Robustness in this chapter is framed as invariance (or controlled sensitivity) with respect to two interacting factors: (i) the anchor set used to construct the representation, and (ii) the specific graph instance on which the representation is computed.

Anchor-choice robustness. Different anchor sets can change coverage, locality, and resolution of distance profiles. For example, anchors concentrated in one community may yield representations that poorly discriminate nodes in other regions; similarly, small anchor sets can induce coarse distance signatures with many ties. Even when the representation is further compressed (e.g., via PCA/SVD), additional non-identifiability arises: principal components may flip sign, permute, or rotate within nearly-degenerate subspaces. A robust train-on-one model should tolerate these changes, either because the representation is intrinsically stable or because the pipeline includes normalization or alignment steps that reduce anchor-induced nuisance variation.

Graph-variation robustness. When moving across graphs (or across snapshots/perturbations of the same graph), distribution shift may occur in degree distributions, community structure, diameter, and connectivity patterns. Such shifts change distance statistics and may alter the scale of derived features. The train-on-one setting therefore acts as an inductive generalization probe: it tests whether the learned decision boundary depends on idiosyncratic global properties of the training graph, or on more transferable relational patterns. In addition, evaluating across multiple graphs provides a principled way to separate anchor variance from graph variance, and to diagnose which dominates in practice.

In summary, train-on-one, evaluate-on-many provides a unified robustness lens: it quantifies stability under anchor resampling on a fixed graph, and transferability under graph changes, while keeping the downstream predictor fixed.

7.2.3 Computational Benefits and Deployment Considerations

Training a downstream predictor repeatedly for many anchor sets is often the dominant cost in large-scale experiments, especially when hyperparameter tuning is performed per anchor realization. The train-on-one paradigm reduces this cost by amortizing training: the model is trained once and then evaluated across many anchor sets and/or graphs. This is particularly beneficial when anchor sets are used as a source of randomness to estimate variance, since traditional protocols require a full retrain for each replicate.

From a deployment perspective, the paradigm also matches realistic constraints. In many applications, a model is trained once and then applied repeatedly as data evolves. Anchor sets might change due to streaming updates, dynamic graph growth, privacy constraints (e.g., anchors cannot be fixed globally), or system-level resampling for load balancing. A predictor that remains accurate under such changes is easier to deploy and maintain. Moreover, if lightweight post-processing (e.g., feature standardization or subspace alignment) can restore compatibility across anchor sets, then the system can support representation refreshes without expensive retraining.

Finally, train-on-one evaluation provides a clean way to report *operational* performance. Instead of reporting a best-case result conditioned on a favorable anchor set, one can report distributions (mean/variance, quantiles) of performance under anchor resampling and graph variation, which more accurately reflects real-world uncertainty and robustness.

7.3 Problem Setting and Notation

7.3.1 Graphs, Tasks, and Splits

Let $G = (V, E)$ denote a graph with node set V and edge set E . We write $n := |V|$ and $m := |E|$. Unless otherwise stated, we consider a single graph with a fixed node identity space, and a downstream prediction task defined over nodes or node pairs.

Node-level tasks. For node classification, each node $v \in V$ is associated with a label $y_v \in \{1, \dots, C\}$. Let $\mathcal{V}_{\text{tr}}, \mathcal{V}_{\text{va}}, \mathcal{V}_{\text{te}} \subseteq V$ denote disjoint train/validation/test node sets. The training

objective is to learn parameters θ of a predictor f_θ such that $f_\theta(\phi(v)) \approx y_v$ for $v \in \mathcal{V}_{\text{tr}}$, where $\phi(v)$ is the representation of node v induced by the anchor-based pipeline.

Edge-/pair-level tasks. For link prediction, we consider node pairs (u, v) with binary labels $y_{uv} \in \{0, 1\}$ indicating the presence of an edge or an interaction. Let $\mathcal{E}_{\text{tr}}, \mathcal{E}_{\text{va}}, \mathcal{E}_{\text{te}}$ denote train/validation/test edge sets (with corresponding negative samples as needed). The predictor operates on a pairwise feature map $\psi(u, v)$ derived from node representations, e.g., $\psi(u, v) = \text{combine}(\phi(u), \phi(v))$ using concatenation, elementwise product, or other standard operators.

Splits and transductive setting. Throughout, we assume the graph structure is available at training time (transductive learning), while labels/edge targets are split into train/validation/test. The train-on-one, evaluate-on-many protocol keeps the task definition and splits fixed, while varying the anchor sets and potentially the graph instance used at evaluation.

7.3.2 Anchor Sets and Distance Constructions

An anchor set is a subset $A \subseteq V$ with $k := |A|$. In the train-on-one, evaluate-on-many setting, we distinguish a reference anchor set $A^{(0)}$, used for training, from a collection of evaluation anchor sets $\{A^{(t)}\}_{t=1}^T$.

Given an anchor set $A^{(t)}$, we construct an anchor-based distance matrix $D^{(t)} \in \mathbb{R}^{n \times k}$, where the (v, a) -entry is the shortest-path distance between node $v \in V$ and anchor $a \in A^{(t)}$:

$$D_{v,a}^{(t)} = \text{dist}_G(v, a).$$

Depending on the experiment, this matrix may be used directly, normalized, or transformed through PCA/SVD to obtain topology coordinates. In the Procrustes experiments, the main source of variation is the anchor set or the subgraph view, not a change from one graph kernel or similarity operator to another.

When evaluating across multiple graphs, we write $G^{(s)} = (V^{(s)}, E^{(s)})$ for the s -th graph instance, with its own anchor sets $\{A^{(s,t)}\}_t$ and distance matrices $\{D^{(s,t)}\}_t$. Depending on the exper-

imental design, graphs may share node identities, as in overlapping subgraph views, or be disjoint, as in cross-dataset transfer experiments.

7.3.3 Representations as Multiple Views

From each node-anchor matrix $D^{(t)}$, we derive a node representation matrix $X^{(t)} \in \mathbb{R}^{n \times d}$ through a deterministic (or seeded) pipeline:

$$X^{(t)} = \mathcal{T}(D^{(t)}),$$

where $\mathcal{T}$ may include normalization, nonlinear transformations, and dimensionality reduction (e.g., PCA/SVD). The node representation of v under anchor set $A^{(t)}$ is the row vector $\phi^{(t)}(v) \in \mathbb{R}^d$.

We interpret each $X^{(t)}$ as a **view** of the same underlying nodes: different anchor sets induce different coordinate systems and potentially different distortions. Under this view, train-on-one corresponds to training a predictor on $X^{(0)}$ and evaluating it on $X^{(t)}$ for $t \geq 1$ without updating θ .

7.4 Sources of Variation

Train-on-one, evaluate-on-many isolates a particular failure mode: the downstream predictor f_θ is fixed after training on a reference representation $X^{(0)}$, yet at evaluation time it is applied to representations $X^{(t)}$ induced by different anchors, different construction choices, or different graphs. The resulting performance gap reflects *nuisance variation* (changes in representation that should not affect the task) as well as *true distribution shift* (changes that legitimately alter the decision boundary). This section decomposes the main sources of variation considered in this chapter.

7.4.1 Anchor-Set Variability (Sampling, Size, Coverage)

Anchor sets $A^{(t)}$ typically arise from stochastic sampling or heuristics, and each choice can alter the statistics of node-anchor relations.

Sampling randomness. Even when anchors are sampled uniformly at random from V , the induced node-anchor distance profiles can vary substantially across realizations. This variation is amplified in graphs with heavy-tailed degree distributions or pronounced community structure, where a small number of anchors may land disproportionately in dense regions.

Anchor set size. The anchor count $k = |A^{(t)}|$ controls the resolution of the representation. Small k yields coarse signatures with limited discriminability, while large k increases coverage and expressiveness but may introduce redundancy and computational cost. Importantly, train-on-one is sensitive to *mismatch* in k : a predictor trained on embeddings derived from one k may not transfer cleanly to embeddings derived from another k if the embedding dimension, scaling, or variance structure changes.

Coverage and locality. Anchors that concentrate within a subset of communities can produce distance matrices where many nodes are similarly far from all anchors, collapsing separability. Conversely, anchors spread across communities improve coverage and yield more informative distance profiles. Coverage interacts with graph diameter and connectivity: in graphs with large diameter or weak inter-community links, distances may saturate (many large values), reducing effective signal.

Disconnected or weakly connected regions. If G contains multiple components (or nearly-disconnected regions), shortest-path distances to anchors outside a component may be infinite or truncated, producing structured missingness. The resulting representation can shift abruptly depending on whether anchors fall within each component, creating a strong dependence on anchor placement.

7.4.2 Representation Variability

Even with a fixed anchor set size, representations can vary because different anchor selections produce different node-to-anchor distance matrices, and because the downstream embedding depends on preprocessing choices applied to those distance matrices.

Anchor-induced distance variation. In this dissertation, the structural signal is derived primarily from node-to-anchor shortest-path distances. Different anchor sets emphasize different regions of the same graph and can therefore produce distance matrices with different distributions, coverage patterns, and levels of redundancy. In graphs with strong community structure or heterogeneous degree distributions, this variation can be substantial even when the anchor budget is fixed.

Transformations and scaling. The raw distance matrix may be used directly or further processed through truncation, standardization, or low-dimensional projection. Such choices alter the relative magnitudes and geometry of the resulting feature space. Since many downstream predictors are not inherently scale-invariant, inconsistent scaling across anchor sets or graph layers can directly affect transfer performance.

Normalization and centering. Row-wise normalization, column-wise normalization, and global normalization can each stabilize different aspects of the distance matrix $D^{(t)}$. For example, column-wise standardization can reduce the effect of anchors whose distance distributions differ systematically from others, while row-wise normalization can reduce magnitude effects associated with node centrality or graph position. At the same time, normalization can suppress task-relevant information if labels are correlated with absolute distance scale.

Dimensionality reduction non-identifiability. When $X^{(t)}$ is obtained by applying SVD- or PCA-based dimensionality reduction to $D^{(t)}$, the resulting embedding is defined only up to sign flips, permutations of components with similar singular values, and orthogonal rotations within nearly degenerate subspaces. As a result, even if two anchor sets produce distance matrices with similar structural information, their low-dimensional embeddings may still be misaligned. This is one of the main reasons that a fixed predictor may underperform across views unless an alignment step, such as orthogonal Procrustes, is applied.

7.4.3 Graph Variability (Perturbations, Subgraphs, Synthetic Families)

Train-on-one can also be evaluated across multiple graph instances, where variability arises from changes in the underlying topology.

Graph perturbations. Small edge perturbations (random deletions/additions, rewiring) can change shortest-path distances discontinuously, especially near bridges and cut edges. Even when the same distance construction is used, the resulting node-to-anchor distance distributions can shift in ways that affect downstream feature distributions.

Subgraphs and induced sampling. Evaluating on subgraphs (e.g., extracted partitions, ego-nets, or sampled induced subgraphs) changes global properties such as diameter, average distance, and community mixing. Node-anchor distances measured within a subgraph can systematically differ from those in the full graph, even for the same anchors, creating a structured domain shift.

Synthetic graph families. When graphs are drawn from a generator family (e.g., stochastic block models, configuration-style models), variability reflects both the randomness of the generator and parameter changes (e.g., community strength, degree heterogeneity). This setting is useful for controlled stress testing: one can vary a single property (e.g., mixing parameter) to study how transfer degrades and which representation choices are most stable.

Node identity alignment. In cross-graph settings, a further distinction is whether node identities are shared (temporal snapshots) or disjoint (different sampled graphs). Shared identities permit direct evaluation on the same node set, while disjoint identities typically require inductive predictors that operate on features only, making feature distribution alignment and normalization more critical.

7.4.4 When Train-on-One is Expected to Fail

Train-on-one is not universally achievable; it is expected to fail when the induced representation changes in ways that alter the effective decision boundary.

Severe anchor mismatch or poor coverage. If the reference anchor set provides informative distances for all classes/regions but an evaluation anchor set concentrates in a subset of the

graph, then the resulting features may collapse for many nodes. No fixed predictor can recover information that is absent from the evaluation representation.

Uncorrected embedding misalignment. When dimensionality reduction introduces arbitrary sign/permutation/rotation changes, a predictor trained in one coordinate system may not transfer unless it is invariant to these transformations (e.g., uses norms or pairwise distances in embedding space) or an explicit alignment step is applied.

Strong distribution shift across graphs. If evaluation graphs differ substantially in topology (degree distribution, community structure, connectivity), then distance and coordinate distributions shift. In such cases, feature normalization alone may be insufficient, and retraining or domain adaptation may be necessary.

Label mechanisms tied to absolute scales. If labels depend on absolute distance scales (e.g., global centrality thresholds) and those scales change across anchor sets or graphs, then invariance is not appropriate: a model trained under one scale may systematically misclassify under another.

Non-stationary targets. In temporal graphs or evolving interaction networks, the label function itself may change over time. In this case, performance degradation reflects target shift rather than representation instability, and train-on-one should be interpreted as a lower bound on achievable accuracy without continual retraining.

These failure conditions motivate the use of normalization, careful reporting, and alignment to reduce nuisance variation while preserving task-relevant structure.

7.5 Train-on-One, Evaluate-on-Many Protocol

This section specifies the experimental protocol used throughout the chapter. The core principle is to *freeze* the downstream predictor after training on a single *reference* representation, and then measure how performance changes when the representation is recomputed under alternative anchor sets and/or alternative graph instances.

Figure 7.1 summarizes this protocol. A reference anchor set and coordinate construction first define the training representation $X^{(0)}$. A downstream predictor f_θ is trained on this reference view

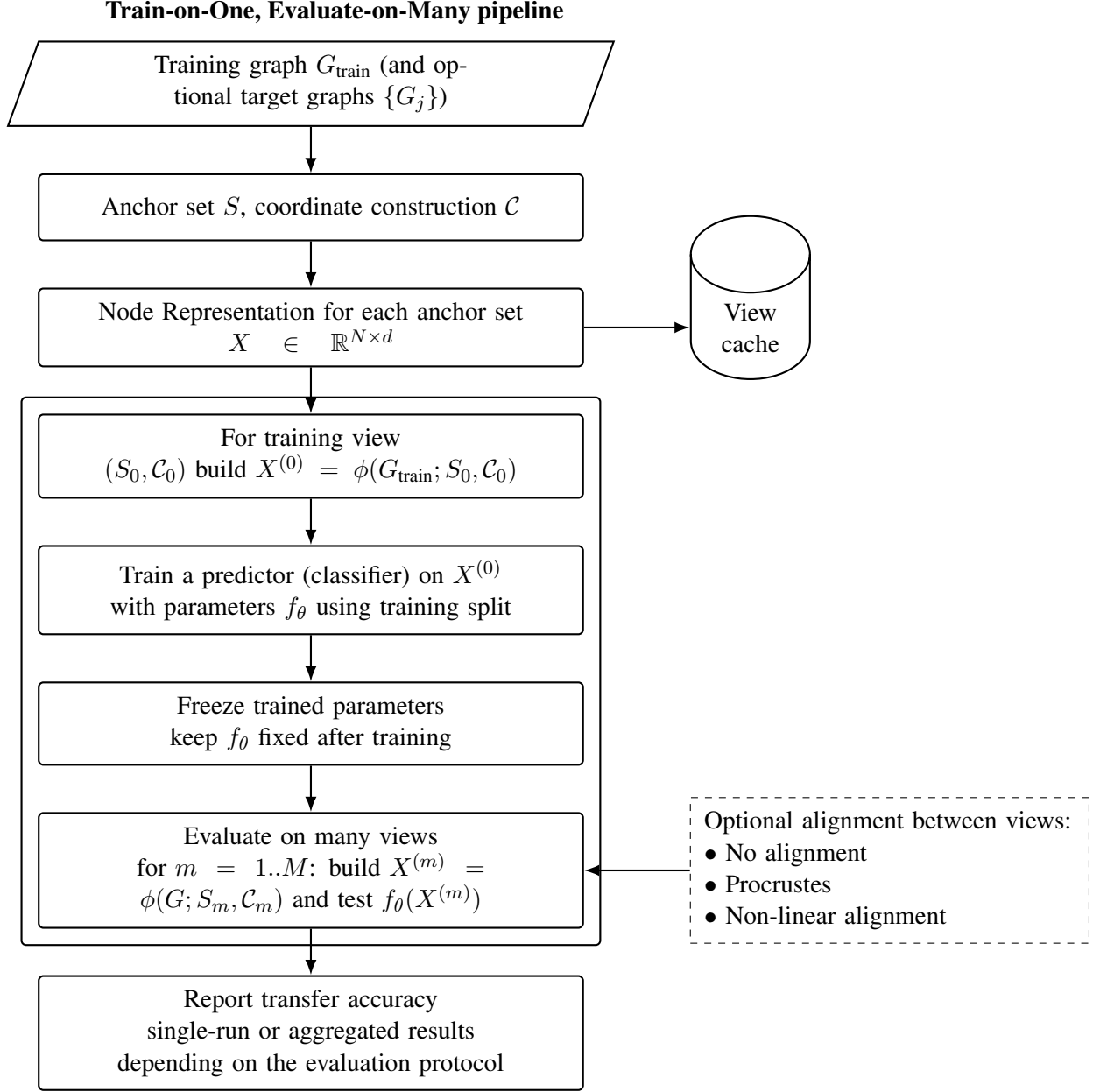


Figure 7.1: Train-on-One, Evaluate-on-Many pipeline. A predictor f_θ is trained on a reference representation view defined by an anchor set and coordinate construction, then evaluated without retraining on alternative anchor-induced views and, when applicable, related graph instances.

and then frozen. At evaluation time, new representations $X^{(m)}$ are constructed from alternative anchor sets, subgraphs, or related graph instances. The frozen predictor is applied to these target views without retraining, optionally after an alignment step such as Procrustes alignment. This

procedure separates representation variability from predictor retraining and allows transfer across views to be measured directly.

7.5.1 Reference Anchor Set Selection

Let $A^{(0)} \subseteq V$ denote the reference anchor set with $k = |A^{(0)}|$. The reference anchors must be selected in a way that is reproducible and representative, since they define the feature space on which the predictor is trained.

Selection rule. In the reported experiments, $A^{(0)}$ is selected using uniform random sampling with a fixed random seed unless otherwise stated. In experiments where anchor size varies, k is treated as a controlled parameter; otherwise, k is fixed across all anchor sets to isolate the effect of anchor *identity* rather than dimensionality.

Reference adequacy. To avoid pathological reference choices, the reference anchor set should satisfy basic coverage constraints, such as including anchors across major connected components (when applicable) or achieving a minimum dispersion (e.g., average pairwise distance among anchors). In practice, this can be enforced by rejecting anchor sets that yield excessive truncation/infinite distances, or, in future work, by using a spread-out selection heuristic such as farthest-point sampling.

Separation from evaluation anchors. Evaluation anchor sets $\{A^{(t)}\}_{t=1}^T$ are generated independently of $A^{(0)}$ using the same selection strategy (unless explicitly studying strategy mismatch). This ensures that train-on-one performance reflects generalization under anchor resampling rather than reuse of identical anchors.

7.5.2 Training Procedure on the Reference View

Given $A^{(0)}$, we construct the reference node-anchor matrix $D^{(0)} \in \mathbb{R}^{n \times k}$ and apply the representation pipeline to obtain $X^{(0)} = \mathcal{T}(D^{(0)}) \in \mathbb{R}^{n \times d}$.

Model training. A downstream predictor f_θ (e.g., logistic regression, MLP, or other classifier) is trained using only $(X^{(0)}, \mathcal{V}_{\text{tr}})$ for node tasks (or $(X^{(0)}, \mathcal{E}_{\text{tr}})$ for link tasks). Hyperparameters are

selected using validation performance computed on the same reference view $X^{(0)}$. After selection, the final model parameters θ^* are fixed and reused for all subsequent evaluations.

No adaptation constraint. In the strict protocol, *no* model parameters are updated when moving from $X^{(0)}$ to any other representation $X^{(t)}$ or to any other graph $G^{(s)}$. Any additional processing applied at evaluation time must be either (i) purely deterministic given the evaluation representation (e.g., per-view standardization computed on evaluation features only), or (ii) fit using source statistics computed during training and then applied to the target (e.g., z-scoring with source mean/variance). Optional calibration layers (if used) are reported separately as an explicit form of lightweight adaptation.

7.5.3 Evaluation on Multiple Anchor Sets for the Same Graph

To evaluate cross-anchor generalization on a fixed graph G , we sample T evaluation anchor sets $\{A^{(t)}\}_{t=1}^T$ and compute $X^{(t)} = \mathcal{T}(D^{(t)})$ for each.

Fixed task split. The train/validation/test splits are held fixed across all t . In particular, test performance is measured on $\mathcal{V}_{\text{te}}$ (or $\mathcal{E}_{\text{te}}$) using the same target set, ensuring that only the representation changes.

Evaluation metric. For each t , we evaluate the frozen model f_{θ^*} on $X^{(t)}$ to obtain a score $s^{(t)}$ (e.g., accuracy, F1, ROC-AUC). The collection $\{s^{(t)}\}_{t=1}^T$ defines the *train-on-one, evaluate-on-many* distribution for the graph.

Optional alignment variants. If an alignment method is applied, such as aligning $X^{(t)}$ to the reference basis before inference, it is applied consistently for all t and reported as a separate method variant under the same evaluation anchor sets.

7.5.4 Evaluation Across Multiple Graphs

For cross-graph transfer, we train on a source graph $G^{(\text{src})}$ using its reference anchors $A^{(\text{src},0)}$ and representation $X^{(\text{src},0)}$. We then evaluate on one or more target graphs $\{G^{(\text{tgt},s)}\}_{s=1}^S$.

Per-graph anchors. For each target graph $G^{(\text{tgt},s)}$, we first sample evaluation anchor sets $\{A^{(\text{tgt},s,t)}\}_{t=1}^{T_s}$ and compute representations $X^{(\text{tgt},s,t)}$. In the disjoint-identity setting, evaluation is

conducted on the target’s own labeled split, and the predictor is applied purely as a function of features.

Consistent preprocessing contract. The representation pipeline $\mathcal{T}$ is fixed across graphs. When $\mathcal{T}$ includes operations that are inherently data-dependent (e.g., PCA), we must specify whether the basis is fit per graph/view (requiring alignment) or fixed from the source (requiring a mapping from target features into the source basis). The chosen convention is treated as part of the method definition and kept constant for all target graphs.

Per-graph scoring. For each target graph and anchor set, we obtain a score $s^{(\text{tgt},s,t)}$. We report both within-graph variability (across t) and across-graph variability (across s).

7.6 Methods for Cross-Graph Transfer

This section focuses on transferring a predictor trained on a source graph $G^{(\text{src})}$ to one or more target graphs $\{G^{(\text{tgt})}\}$ without retraining. Compared to cross-anchor evaluation on a fixed graph, cross-graph transfer introduces additional distribution shift in the underlying topology and, in many cases, a mismatch in node identity spaces. We describe assumptions that make transfer feasible, anchor selection strategies that minimize graph-specific coupling, and practical diagnostics/normalizations that improve robustness.

7.6.1 Inductive vs Transductive Assumptions

A central distinction is whether node identities are shared between training and evaluation graphs.

Transductive cross-graph evaluation (shared identities). In temporal snapshots or evolving graphs, we may have a sequence $\{G_t\}$ over a common node set V . In this case, node indices are aligned across time, and it is meaningful to train on G_{t_0} and evaluate on G_{t_1} . The representation $\phi_t(v)$ is computed from each snapshot, but the predictor can be applied to the same node IDs. Transfer is primarily challenged by *feature shift* induced by topology changes and anchor-set changes.

Inductive cross-graph transfer (disjoint identities). When graphs have disjoint node sets (e.g., different sampled synthetic graphs or different datasets), there is no node-level correspondence. Transfer requires a predictor that operates purely on features and does not rely on graph-specific learned embeddings tied to node IDs. In this setting, the representation pipeline must produce *comparable feature distributions* across graphs, or include an explicit normalization/alignment mechanism. Inductive transfer is therefore more sensitive to the choice of distance-based representation, coordinate construction, and scaling.

Task compatibility. Cross-graph transfer additionally assumes the downstream task is consistent across graphs: the label space, label semantics, and evaluation metrics must match. For node classification, this typically means the same set of classes C and a comparable labeling mechanism. For link prediction, it means the same definition of positive/negative edges (or interactions) and a comparable negative sampling policy.

7.6.2 Potential Anchor Strategies for Cross-Graph Transfer

Anchors can either be chosen by graph-independent rules (favoring invariance and portability) or by graph-specific rules that are designed to be comparable through a shared interface. The reported experiments primarily use random anchors; the strategies below are included to clarify possible extensions rather than to claim that all were evaluated in this dissertation.

Graph-Independent Anchor Rules (e.g., degree/centrality sampling)

Graph-independent anchor selection aims to reduce coupling to idiosyncratic structure in any particular graph while still producing anchors that cover the graph and provide informative distance profiles.

Uniform random node sampling. Select anchors by sampling nodes uniformly at random: $A \sim \text{Unif}(V, k)$. This is simple, scalable, and makes anchor statistics analyzable in expectation. Across graphs drawn from a common family, uniform sampling yields anchor sets whose distribution is comparable, although coverage can still vary with graph size and community structure.

Degree-biased sampling. Select anchors with probability proportional to node degree (or a capped degree), favoring hubs that tend to have small distances to many nodes. This often improves coverage and reduces the frequency of extreme distances, but can also over-emphasize dense regions and reduce discriminability in peripheral regions if the graph is highly heterogeneous.

Centrality-based heuristics. Anchors can be chosen as top- k nodes under a centrality score (approximate betweenness, closeness, PageRank). These tend to stabilize distance scales and improve coverage, which is beneficial for transfer. However, the ranking and scale of centrality measures can change markedly across graphs, so care is needed to ensure the selection rule is implemented consistently and efficiently (often via approximations).

Stratified or community-aware sampling without explicit matching. If lightweight community detection is feasible, anchors can be sampled proportionally within detected communities to improve coverage. Importantly, this does not require communities to correspond across graphs; it only enforces within-graph diversity, which improves robustness of the induced representations.

Overall, graph-independent rules are most suitable when graphs are different instances of a broadly similar distribution, and when a stable, portable selection mechanism is preferred over graph-specific optimization.

Graph-Specific Anchors with Shared Representation Interfaces

Graph-specific strategies choose anchors tailored to each graph, but enforce a shared *interface* so that the downstream predictor can consume features consistently across graphs.

Anchor interfaces as “roles” rather than identities. When graphs are disjoint, individual anchor identities are meaningless across graphs. Instead, anchors can be treated as exchangeable, and the representation pipeline should be invariant (or approximately invariant) to anchor permutations. Practical ways to achieve this include: (i) using summary statistics over anchors (e.g., histograms or quantiles of distances), (ii) applying permutation-invariant pooling over anchor dimensions, or (iii) using an alignment step that maps each graph’s anchor-induced feature space into a shared latent space.

Prototype-based anchors. Anchors can be selected as graph-specific prototypes (e.g., via k -medoids over a structural similarity space), yielding anchors that represent diverse “structural types” within each graph. While prototypes differ across graphs, the induced representations may become more comparable because each anchor set covers a similar range of structural roles.

Landmarks via iterative farthest-point sampling. Selecting anchors by maximizing coverage (e.g., greedy farthest-point sampling under shortest-path distances) yields well-spread landmarks per graph. This often stabilizes distance signatures and reduces redundancy. Cross-graph comparability then relies on the fact that both graphs are covered by similarly distributed landmarks, even though the landmark identities differ.

Shared dimensionality and preprocessing contract. Regardless of how anchors are chosen, transfer requires a consistent contract: fixed k (or a mapping from variable k to fixed d), identical transformations/normalizations, and a consistent dimensionality reduction procedure. For example, using a fixed embedding dimension d with per-graph PCA requires additional care because PCA bases are graph-specific; in such cases, alignment (e.g., Procrustes) or basis-invariant features become necessary.

7.7 Implementation Details

This section summarizes the practical implementation choices used to instantiate the train-on-one, evaluate-on-many protocol. The emphasis is on reproducibility and on isolating variability sources (anchors, graph instances, and training randomness) through careful control of seeds and consistent preprocessing contracts.

Figure 7.2 shows the implementation-level representation pipeline used in the transfer experiments. The model can consume node features, node-to-anchor distance features, anchor-to-anchor information, or combinations of these inputs. Node features are passed through a linear projection, while distance-derived information is encoded through a positional or structural feature branch. Anchor information is also pooled into a context vector and fused with the node representation before prediction. This implementation supports several ablation settings, including node-feature-

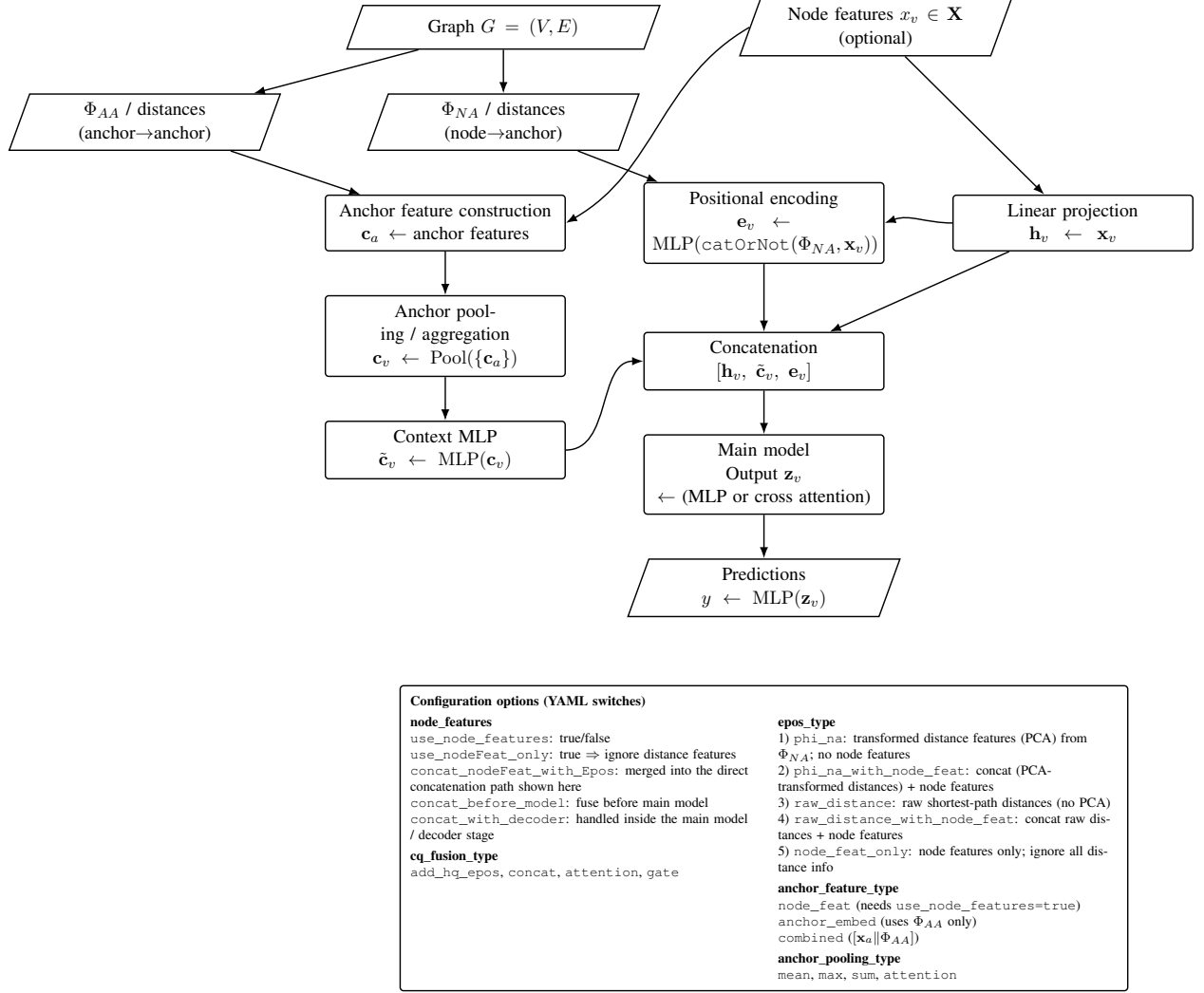


Figure 7.2: Implementation-level pipeline used in the transfer experiments. Node features feed both the linear projection and the positional encoding, while anchor-derived information is aggregated into a context vector before fusion in the main model.

only inputs, distance-only inputs, raw-distance inputs, PCA-transformed distance inputs, and combined node-feature-plus-distance inputs.

7.7.1 Data Pipeline

Graph loading and preprocessing. Each dataset is loaded into a sparse adjacency format (CSR/CSC) and converted into an undirected graph. Self-loops are removed to avoid implicit changes in distance computations. For multi-graph experiments, we store each graph instance $G^{(s)}$ with its own metadata (number of nodes/edges, component statistics) for later diagnostics.

Splits and label access. Train/validation/test splits are fixed once per dataset and reused across all anchor sets and seeds. For node classification, we keep $\mathcal{V}_{\text{tr}}, \mathcal{V}_{\text{va}}, \mathcal{V}_{\text{te}}$ constant. For link prediction, we keep $\mathcal{E}_{\text{tr}}, \mathcal{E}_{\text{va}}, \mathcal{E}_{\text{te}}$ constant, and fix the negative sampling policy (including the random seed) to ensure comparability across anchor-induced views.

Anchor set generation. For each graph, anchor sets are generated using uniform random sampling unless otherwise stated.

The reference anchor set $A^{(0)}$ is generated once per training seed (or fixed globally, depending on the experiment), while evaluation anchor sets $\{A^{(t)}\}_{t=1}^T$ are generated independently.

Distance construction. For shortest-path distances, we compute multi-source BFS on unweighted graphs from all anchors in a batch, producing $D \in \mathbb{R}^{n \times k}$. For large graphs, we support chunked anchor batches to cap peak memory. When truncation is enabled, distances beyond a hop limit are clipped to a constant. The experiments reported in this dissertation use distance-based coordinate constructions rather than kernel-based node-to-anchor scores.

Normalization and feature preparation. We apply a fixed normalization contract across all views: common options include column-wise z-scoring of D (per view), row-wise normalization, or global scaling. When using z-scoring, small-variance columns are handled by adding an ϵ in the denominator or by dropping columns below a variance threshold. The same choice is used for reference and evaluation views.

Dimensionality reduction and caching. If PCA/SVD is used, we compute a rank- d truncated SVD (or randomized SVD for efficiency) on the normalized matrix and store the resulting node embedding $X \in \mathbb{R}^{n \times d}$. Intermediate products (e.g., normalized D , SVD factors, per-view statistics) are cached to disk keyed by (dataset, graph instance, anchor set id, layer id, preprocessing config) to avoid recomputation and to enable exact reproduction.

7.7.2 Model Architectures and Training Hyperparameters

Predictor families. To separate representation effects from downstream model-capacity effects, we consider several predictor families, including linear classifiers, shallow MLP-style pre-

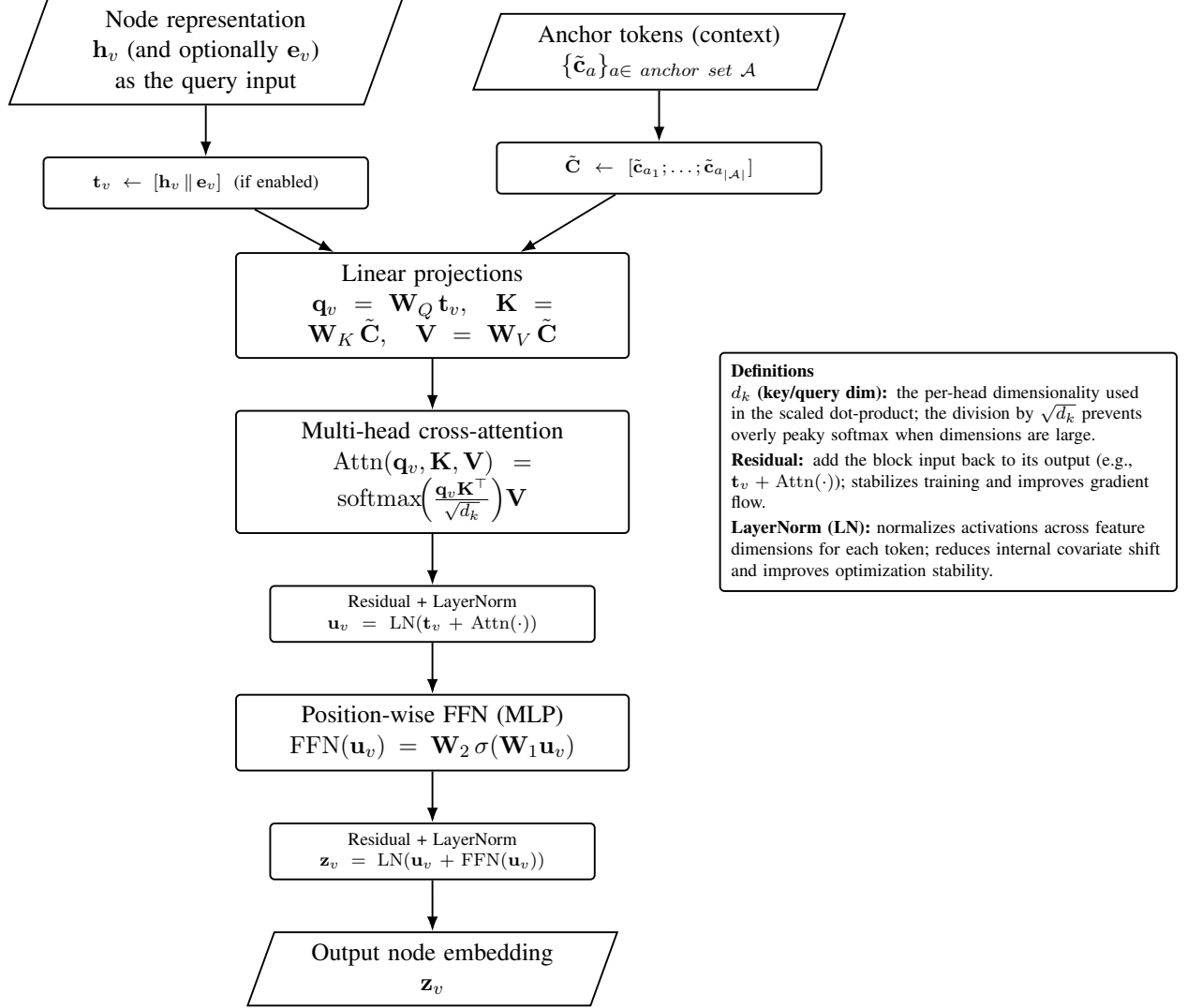


Figure 7.3: Cross-attention variant of the main model. Each node forms a query from $\mathbf{h}_v$ (optionally concatenated with $\mathbf{e}_v$), and attends over anchor tokens $\{\mathbf{c}_a\}_{a \in \mathcal{A}}$ to produce a context-aware embedding $\mathbf{z}_v$.

dictors, and a cross-attention-based architecture for combining node features with anchor-derived structural context. Linear and shallow MLP predictors provide lower-capacity references for evaluating whether the structural representation itself is informative, while the cross-attention model provides a more flexible mechanism for node-anchor interaction. In the reported cross-dataset real-graph transfer experiments, cross-attention is used for selected higher-capacity transfer runs, but the broader goal of the chapter is to evaluate representation reuse across anchor-induced views and related graphs rather than to establish a universally best downstream architecture.

The cross-attention variant used in selected transfer experiments is shown in Figure 7.3. In this model, each node forms a query from its node representation h_v , optionally concatenated with its positional encoding e_v . The anchor-derived context vectors serve as key and value tokens. Multi-head cross-attention then allows each node to attend to the anchor context most relevant to its representation. The attention output is combined with the original node representation through residual connections, layer normalization, and a feed-forward block to produce the final embedding z_v . This design is intended to make the node representation depend not only on its own features, but also on a learned interaction with anchor-derived structural context.

Training on the reference view. All predictors are trained *only* on the reference representation $X^{(0)}$ (and corresponding training labels/edges). Hyperparameters are tuned on the reference validation set computed on $X^{(0)}$ only. Typical hyperparameters include learning rate, weight decay, dropout (MLP), hidden width, and number of epochs. Early stopping is based on reference validation performance with a fixed patience. After selection, the best-performing checkpoint defines θ^* , which is frozen for all evaluations.

Seed control. We separate randomness into: (i) *training seeds* (initialization, minibatch order), (ii) *anchor seeds* (sampling anchors), and (iii) *negative sampling seeds* (for link prediction). Each is logged independently to enable variance decomposition. When reporting, we specify whether $A^{(0)}$ is fixed across training seeds or regenerated per training seed, since this affects interpretation.

Evaluation without retraining. At evaluation time, we compute representations $X^{(t)}$ for each anchor set (and graph instance), and run a forward pass of the frozen predictor. No gradients are computed and no parameters are updated. If probability outputs are needed (e.g., for ROC-AUC), we use the same output activation as during reference validation.

7.7.3 Alignment Implementation Notes

When PCA/SVD is performed independently per anchor set or per graph, the resulting embeddings are not directly comparable because the retained coordinates are only defined up to sign flips, permutations of nearly tied components, and more general orthogonal changes of basis. In

this chapter, alignment is treated as a deterministic evaluation-time post-processing step applied to each target embedding before inference.

Same-graph multi-view setting. Let $A^{(0)}$ denote a reference anchor set on a graph $G = (V, E)$, and let $A^{(t)}$ denote an alternative anchor set on the *same* graph. For each view we construct a node-anchor matrix $D^{(t)}$, apply the same preprocessing pipeline, and obtain an embedding $X^{(t)} \in \mathbb{R}^{n \times d}$ whose v -th row is the representation of node v . A classifier f_{θ^*} is trained once on the reference view $X^{(0)}$ and then reused on $X^{(t)}$ without retraining. In this setting, the node identity space is shared across views, so alignment is well motivated.

Why orthogonal alignment is reasonable. Different anchor sets can be viewed as different projections of the same latent graph structure. Empirically, when the underlying graph is fixed and only the anchors change, the retained singular spectra and principal components are very similar across views. The dominant nuisance factor is therefore not a semantic change in the embedding, but an approximate rotation or reflection of the retained subspace. This makes orthogonal Procrustes a natural first correction: it removes orientation mismatch while preserving distances and relative geometry inside the retained coordinate system.

Reference-aligned orthogonal Procrustes. Rather than using all nodes, we estimate the alignment from a small correspondence set $V_{\text{ref}} \subseteq V$, which may be unlabeled. Let $X_{\text{ref}}^{(0)} \in \mathbb{R}^{|V_{\text{ref}}| \times d}$ and $X_{\text{ref}}^{(t)} \in \mathbb{R}^{|V_{\text{ref}}| \times d}$ denote the reference-node embeddings in the source and target views. We solve

$$R^{(t)} = \arg \min_{R \in \mathbb{R}^{d \times d}, R^\top R = I} \left\| X_{\text{ref}}^{(t)} R - X_{\text{ref}}^{(0)} \right\|_F.$$

If $(X_{\text{ref}}^{(t)})^\top X_{\text{ref}}^{(0)} = U \Sigma V^\top$ is the singular value decomposition of the cross-covariance, then the closed-form solution is

$$R^{(t)} = UV^\top.$$

The aligned evaluation embedding is obtained by

$$\tilde{X}^{(t)} = X^{(t)} R^{(t)},$$

and the frozen predictor is applied to $\tilde{X}^{(t)}$.

Choice of correspondence nodes. When the full node set is shared across views, V_{ref} can be sampled uniformly at random from the common node set. If one wishes to avoid using test nodes even for an unlabeled geometric alignment step, then V_{ref} can be restricted to training or training-plus-validation nodes. In the overlapping-subgraph setting, V_{ref} is sampled from the nodes shared by the source and target subgraphs. In all cases, the correspondence set is much smaller than the full graph, so the alignment cost is negligible compared with feature construction.

Cross-graph alignment limitations. When node identities are disjoint across graphs, direct Procrustes alignment is not justified in the same way as same-graph alignment. In principle, Procrustes requires either a shared node set or an explicit correspondence subset. The cross-dataset Cornell/Texas/acmv9 experiments later in the chapter therefore treat with/without-Procrustes comparisons as *heuristic diagnostics*, not as correspondence-grounded alignment results. They are useful for probing representation mismatch, but should not be interpreted as establishing a general solution for disjoint-graph transfer.

Numerical considerations. We center embeddings before alignment when appropriate (subtract per-dimension means), and optionally normalize by Frobenius norm to prevent scale drift from dominating the objective. When d is large or the retained spectrum is nearly degenerate, Procrustes can become sensitive; we therefore align the full retained subspace rather than attempting to match individual components, and we avoid choosing d at obvious spectral plateaus when possible. All alignment steps are logged and deterministic given the stored embeddings and the selected reference-node subset.

7.8 Experiments

This section reports the empirical evaluation of cross-view and cross-graph reuse in two settings: (i) *single-source train-on-one, evaluate-on-many*, where a model is trained on one source graph/view and evaluated on one or more target graphs/views without retraining, and (ii) *multi-source train-on-many, evaluate-on-one*, where a model is trained jointly on several source graphs

and then evaluated on one target graph. Throughout, the downstream predictor is kept fixed after training, and alignment is treated as an evaluation-time operation.

7.8.1 Datasets and Experimental Setup

The reported experiments focus on real benchmark graphs that share task semantics closely enough to make transfer meaningful. The main source/target combinations are: (i) Cornell and Texas for single-source transfer, (ii) Cornell and Wisconsin as multi-source inputs with Texas as target, and (iii) citationv1 and dblpv7 as source graphs with acmv9 as target.

Single-Source Train-on-One, Evaluate-on-Many

In the single-source setting, we train on one source graph using a single reference view, freeze the predictor, and evaluate the trained model on source and target graphs. In the reported real-graph experiments, this setting is instantiated most clearly by the Cornell $\rightarrow$ Texas and Texas $\rightarrow$ Cornell transfers. These experiments test whether a single learned representation and classifier can be reused across related but non-identical datasets.

Multi-Source Train-on-Many, Evaluate-on-One

In the multi-source setting, we train on multiple source graphs and evaluate on a held-out target graph. The reported experiments include two main examples: (i) Cornell + Wisconsin $\rightarrow$ Texas, and (ii) citationv1 + dblpv7 $\rightarrow$ acmv9. During multi-source training, the validation curves are averaged across the source graphs, while final target evaluation is reported as a single Train/Val/Test triple on the target graph.

7.8.2 Baselines and Comparisons

The experiments emphasize a small number of direct comparisons rather than a broad baseline sweep.

No alignment vs. Procrustes. For the same source/target pair, we compare evaluation with and without Procrustes-style alignment. In same-graph settings this serves as a test of coordinate

non-identifiability. In cross-dataset settings it should be interpreted more cautiously, as discussed in the implementation notes above.

Node-feature-only vs. node-feature-plus-distance. For the acmv9 transfer experiments, we compare models that use only node attributes against models that combine node attributes with distance-based information. This comparison is especially important because the bag-of-words node features are high-dimensional (about 1400 dimensions), whereas the distance-based GC representation is only 32 dimensions.

7.8.3 Metrics and Reporting Protocol

The reported real-graph results are given as single-run Train/Val/Test scores. This is the reporting granularity available from the completed runs used in this chapter. Accordingly, the chapter reports those values directly rather than claiming mean $\pm$ std across repeated anchor sets or seeds that are not available in finalized repeated-run form.

For multi-source training runs, validation accuracy during training is averaged across source graphs. The target-graph results reported below are still single Train/Val/Test triples for the transferred model. Repeated-view statistics, repeated-seed aggregation, and confidence intervals remain natural extensions for future work.

7.8.4 Results and Analysis

We organize the reported results around the two transfer settings and the main ablations available from the experimental logs.

Phase 1: Full Graph with Different Anchor Sets

We begin with phase 1 to study the simplest controlled setting where the complete graph is available, anchors are randomly chosen, and embeddings from different anchor views can be aligned with Procrustes using a small set of randomly chosen reference nodes. This setting isolates the effect of anchor choice, verifies that views are seeing a nearly common coordinate frame, and establishes baseline transfer when training on one view and evaluating on another without re-

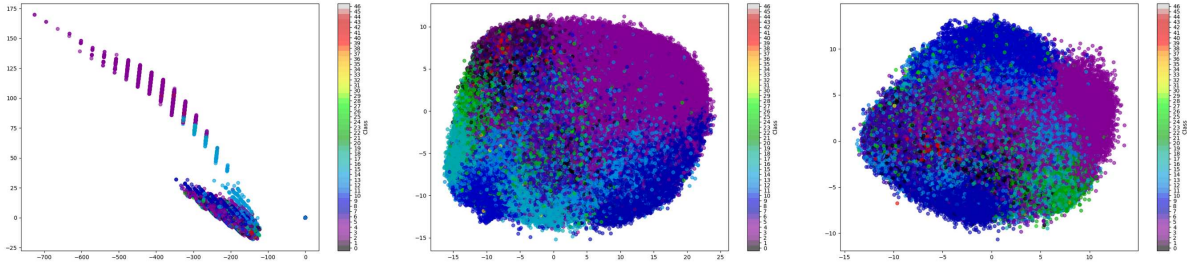


Figure 7.4: TC plots of anchor set 1 (baseline).

training. Phase 2 then relaxes the full coverage assumption by moving to partial views such as subgraphs, which introduces missing nodes and structural variation. The goal is to test whether the alignment procedure and the model trained in partial view remain effective under practical constraints, thereby tracing a clear path from feasibility to robustness.

In the initial phase of our experiments on the OGBN-Products dataset, we worked exclusively on the full graph without applying any subgraphing or partitioning. The objective was to evaluate the effect of different anchor sets on the resulting node representations.

We randomly sampled different sets of anchor nodes from the full graph. For each anchor set, we computed the distance matrix from all nodes in the graph to the selected anchors. These distance matrices were then transformed into node embeddings (topology coordinates, TCs) using Principal Component Analysis (PCA). Let $TC \in \mathbb{R}^{N \times d}$ denote the d -dimensional TC for N nodes. Coordinates columns are ordered by decreasing explained variance.

To inspect how geometry and label information unfold across dimensions, we adopt a visualization: for each $i \in \{1, \dots, k-1\}$ we plot the 2D projection (TC_i, TC_{i+1}) , where TC_j denotes the j -th coordinate column of TC . As shown in Fig. 7.4, we visualized scatter plots of (TC_i, TC_{i+1}) , for $i = 1, 3, 5$ for anchor sets 1 (baseline). Each point is a node; color encodes its class label using a single colormap shared across subplots.

To quantify and correct the rotational discrepancy between embeddings, we employed Procrustes analysis. We selected one anchor set as the baseline and used its corresponding TCs as the reference representation.

For each new anchor set, we randomly sampled 1,000 nodes and extracted their TC embeddings. We then applied Procrustes analysis to compute the optimal orthogonal transformation (rotation matrix) that aligns the sampled embeddings from the current anchor set to the reference embeddings from the baseline. This transformation was subsequently applied to the entire set of TCs from the current anchor set to bring it into alignment with the baseline space.

Let $X \in \mathbb{R}^{N \times d}$ denote the matrix of TCs for a set of N nodes from a new anchor set, and let $Y \in \mathbb{R}^{N \times d}$ denote the corresponding TCs of the same nodes derived from a fixed baseline anchor set. The goal of Procrustes analysis [96] is to find an orthogonal matrix $R \in \mathbb{R}^{d \times d}$ that best aligns X to Y by minimizing the Frobenius norm:

$$R^* = \arg \min_{R \in \mathbb{R}^{d \times d}, R^\top R = I} \|XR - Y\|_F^2 \quad (7.1)$$

This optimization has a closed-form solution. First, compute the cross-covariance matrix $C = X^\top Y$, then take SVD:

$$C = U \Sigma V^\top \quad (7.2)$$

The optimal rotation matrix is then given by:

$$R^* = UV^\top \quad (7.3)$$

Once R^* is obtained, we apply it to all topology coordinates X_{full} of the current anchor set:

$$X_{\text{aligned}} = X_{\text{full}} R^* \quad (7.4)$$

This transformation brings the embeddings into the coordinate space of the baseline anchor set, enabling direct comparison and cross-anchor evaluation.

We then plot the TCs (before applying Procrustes) of anchor set 2 in Fig. 7.5, compared to what colors are distributed in Fig. 7.4, we notice that their shapes are similar, but their orientations are different.

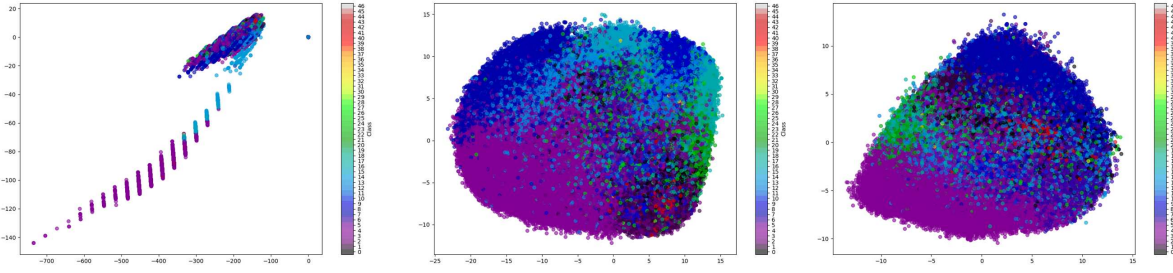


Figure 7.5: TC plots of anchor set 2, before applying Procrustes.

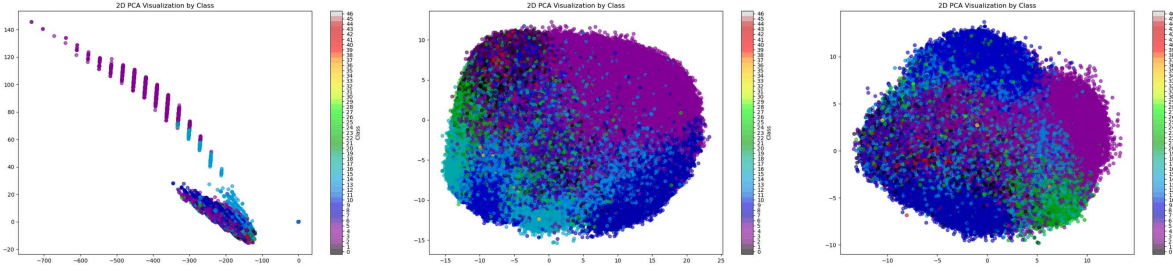


Figure 7.6: TC plots of anchor set 2, after applying Procrustes.

As shown in Fig. 7.6, after applying Procrustes based on the TCs of a small set of reference nodes, even with different anchor sets, the rotated TCs exhibit strong alignment, indicating the effectiveness of the TC and Procrustes transformation. We also conduct an experiment on a third anchor set, where we have similar results, as shown in Fig. 7.7 and Fig. 7.8. A key observation from this phase was that the PCA embeddings derived from different anchor sets were remarkably consistent, differing only up to an orthogonal transformation (e.g., sign flips or rotations). This invariance suggests a latent structural alignment between representations across different anchor sets, despite the randomness and lack of overlap among the anchor nodes.

We then conduct an experiment in which a model is trained on one anchor set, and then evaluate this trained model (without retraining or fine-tuning) on other anchor sets. This test aimed to evaluate the generalizability of the learned representations across anchor sets. As shown in Table 7.1, when training on anchor set 1a (baseline), the accuracy on valid set is 0.8980 and accuracy on test set is 0.7714; when evaluating on anchor set 2a, the accuracy on valid set is 0.8800 and accuracy on test set is 0.7599; when evaluating on anchor set 3a, the accuracy on valid set is 0.8812 and ac-

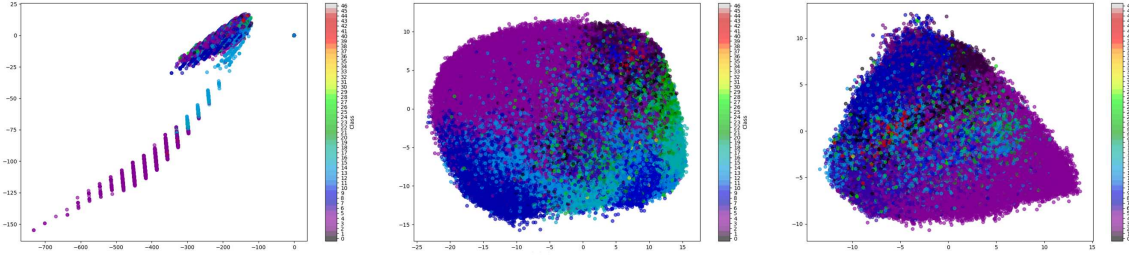


Figure 7.7: TC plots of anchor set 3, before applying Procrustes.

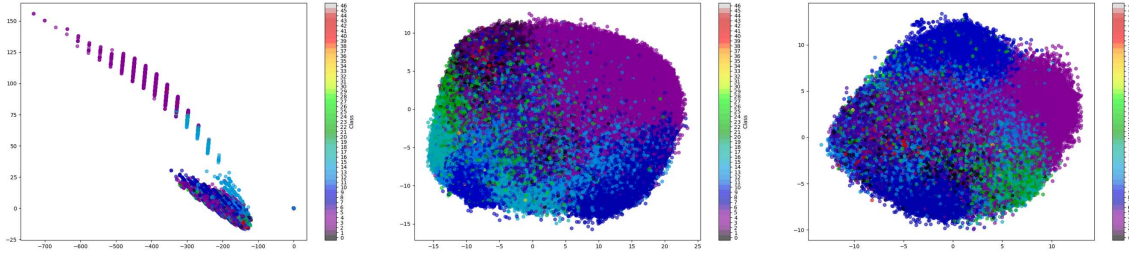


Figure 7.8: TC plots of anchor set 3, after applying Procrustes.

accuracy on test set is 0.7580. We also compare with results of not applying Procrustes on anchor set 2a and 3a, the validation accuracy and test accuracy are lower than 0.2. These results indicate that a single Procrustes rotation estimated from a small reference is sufficient to place distinct anchor views in a shared coordinate frame, preserving most of the model’s predictive power across views.

For TC-based embeddings of the same underlying graph, reference-set Procrustes is a practical, supervision-free step to achieve cross-anchor comparability and reuse trained models.

We also experiment with different numbers of anchors overlapping with anchors from the baseline anchor set. The results are shown in Table 7.2. This test aimed to evaluate the robustness of the anchor selection across anchor sets. As shown in Table 7.2, when training on anchor set 1c (baseline) which has 1000 anchors, the accuracy on valid set is 0.8977 and accuracy on test set is 0.7686; when evaluating on anchor set 2 with 0% nodes from anchor set 1c used as anchors, the accuracy on valid set is 0.8815 and accuracy on test set is 0.7521; when evaluating on anchor set 3 with 50% nodes from anchor set 1c used as anchors, the accuracy on valid set is 0.8873 and accuracy on test set is 0.7609. The experiments with 100 anchors show similar results. Increasing

Table 7.1: Train-on-one, evaluate-on-another across anchor sets. The model is trained on anchor set 1 and then evaluated (with no retraining) on other anchor views after Procrustes alignment to the baseline frame.

Anchor Set	#Anchor	Valid Acc	Test Acc
Anchor set 1a (baseline, train set)	1000	0.8980	0.7714
Anchor set 2a (TCs aligned to 1a)	1000	0.8800	0.7599
Anchor set 2a (TCs NOT aligned to 1a)	1000	0.1237	0.1017
Anchor set 3a (TCs aligned to 1a)	1000	0.8812	0.7580
Anchor set 3a (TCs NOT aligned to 1a)	1000	0.1179	0.0900
Anchor set 1b (baseline, train set)	100	0.8702	0.7207
Anchor set 2b (TCs aligned to 1b)	100	0.8537	0.7125
Anchor set 3b (TCs aligned to 1b)	100	0.8489	0.7111

Table 7.2: Train-on-one, evaluate-on-another across anchor sets. The model is trained on anchor set 1 and then evaluated (with no retraining) on other anchor views after Procrustes alignment to the baseline frame, with #Oanc anchors overlapping with anchors in set 1.

Anchor Set	#Anchor	#Oanc	Valid Acc	Test Acc
1c (baseline, train set)	1000		0.8977	0.7686
2c (TCs aligned to 1c)	1000	0	0.8815	0.7521
3c (TCs aligned to 1c)	1000	500	0.8873	0.7609
1d (baseline, train set)	100		0.8702	0.7207
2d (TCs aligned to 1d)	100	0	0.8256	0.6813
3d (TCs aligned to 1d)	100	50	0.8412	0.6928

the percentage of baseline anchors used gives slightly better performance, though the gains are modest. Thus, we can randomly choose anchors when evaluating a trained model on the same graph.

We also explore having some reference nodes overlapping with the anchor in the baseline anchor set. As shown in Table 7.3, training on the baseline view and evaluating on other anchor views after Procrustes alignment is feasible. For the 1000 anchor setting, aligning from sets 2e, 3e, 4e to set 1e yields a small drop from the baseline (valid 0.8977 and test 0.7719) to about 0.876–0.879 on validation and 0.756–0.760 on test, and using overlapping reference anchors provides only modest gains as #Oref increases from 0 to 1000. In contrast, with only 100 anchors, transfer from set 1f is noticeably harder: aligned views 2f, 3f, 4f remain well below the baseline 1f on both valida-

Table 7.3: Train-on-one, evaluate-on-another across anchor sets. The model is trained on anchor set 1 and then evaluated (without retraining) on other anchor views after Procrustes alignment to the baseline frame, with #Anchor nodes as reference nodes, where #Oref reference nodes overlapping with anchors in set 1.

Anchor Set	#Anchor	#Oref	Valid Acc	Test Acc
1e (baseline, train set)	1000		0.8977	0.7719
2e (TCs aligned to 1e)	1000	0	0.8764	0.7560
3e (TCs aligned to 1e)	1000	500	0.8785	0.7580
4e (TCs aligned to 1e)	1000	1000	0.8786	0.7598
1f (baseline, train set)	100		0.8702	0.7207
2f (TCs aligned to 1f)	100	0	0.7319	0.6077
3f (TCs aligned to 1f)	100	50	0.7460	0.6164
4f (TCs aligned to 1f)	100	100	0.7523	0.6121

tion and test, though adding overlapping anchors to the reference set improves validation accuracy somewhat. Overall, alignment supports cross-view transfer; the benefit of overlapping reference anchors is positive but small, and the performance gap is primarily driven by the anchor budget rather than the exact size of the overlapping reference set.

Phase 2: Subgraph Generation by Node Removal

In phase 2, we move from the complete graph in phase 1 to subgraphs obtained by removing nodes to mirror the conditions that arise in practice, where coverage of the view is incomplete due to scale, privacy, or distributed storage. Node removal introduces missing vertices and paths, which changes the graph structure and distance-based embeddings, therefore providing a direct stress test for the alignment procedure validated in phase 1. By working with subgraphs formed from random node removal, we model unstructured missingness while keeping the setup simple. With randomly chosen anchors and a small random set of reference nodes for Procrustes alignment, we test whether embeddings from distinct partial views can be placed in a common coordinate frame and whether a model learned under one pattern of missing nodes generalizes to another pattern.

In this phase, we generated subgraphs by randomly removing a subset of nodes from the full OGBN-Products graph. The goal was to simulate scenarios where different parts of the graph

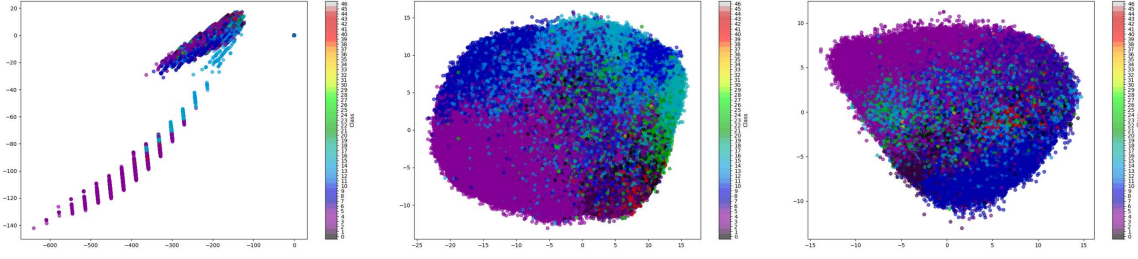


Figure 7.9: TC plots of subgraph 1 (baseline).

are independently accessible, such as in federated or distributed learning settings. Each subgraph retained most of the original structure but had different coverage due to the random node removals.

For efficiency, we remove a fixed percentage of nodes. In the reported experiments, we randomly remove 30% of the nodes, extract the largest remaining connected component, and use it as the subgraph view.

For each subgraph, we recomputed distance matrices based on independently sampled anchor sets and applied PCA to obtain TCs. As shown in Fig. 7.9, we first visualized scatter plots of (TC_i, TC_{i+1}) , for $i = 1, 3, 5$ for subgraph 1 (baseline). Different colors represent different labels of each node.

We then randomly remove 30% nodes from the original OGBN-Products graph to create subgraph 2, then plot the TCs (before applying Procrustes) of subgraph 2 in Fig. 7.10, compared to what colors are distributed in Fig. 7.9, we notice that their shapes are slightly different.

To allow comparison and alignment between these embeddings, we again used Procrustes analysis to align the coordinates of nodes shared between subgraphs. We randomly sampled TCs of 10,000 nodes as Procrustes references. When applying Procrustes, we use part of these nodes (if they exist in the new subgraph) as references to derive the rotation matrix.

As shown in Fig. 7.11, after applying Procrustes, although the shapes are still slightly different from the shapes in Fig. 7.9, the orientations in Fig. 7.11 are more similar to the orientations in Fig. 7.9.

We also conduct the same comparison for subgraph 3, as shown in Fig. 7.12 and Fig. 7.13.

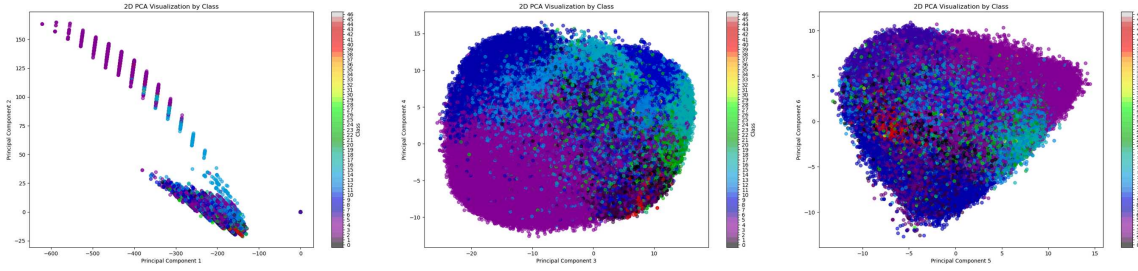


Figure 7.10: TC plots of subgraph 2, before applying Procrustes.

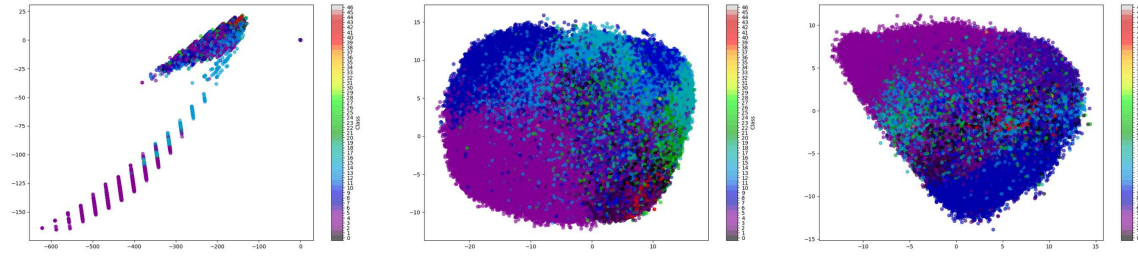


Figure 7.11: TC plots of subgraph 2, after applying Procrustes.

This setup allowed us to investigate further whether the learned representations are transferable not only across anchor sets but also across distinct graph views with partial node overlap.

In our experiments, we randomly remove 30% of the nodes to generate subgraphs, and list the experiment results in Table 7.4. When training on subgraph 1 (baseline), the accuracy on the valid and test sets is 0.8551 and 0.7106. When evaluating the trained model on other subgraphs, the valid and test accuracy are 0.8387 and 0.6983 (subgraph 2), and 0.8383 and 0.6975 (subgraph 3). The results of this phase laid the foundation for more structured partitioning and multi-subgraph training strategies in future work.

These results suggest that using Procrustes as a lightweight, supervision-free commensuration step for train-on-one, evaluate-on-another across subgraphs. These also motivate more structured partitioning (e.g., community splits) and multi-subgraph training in future work.

Single-Source Cross-Graph Transfer: Cornell and Texas

The Cornell/Texas single-source cross-graph transfer results reported in Table 7.5 use the cross-attention model as a higher-capacity predictor for combining node features and anchor-derived

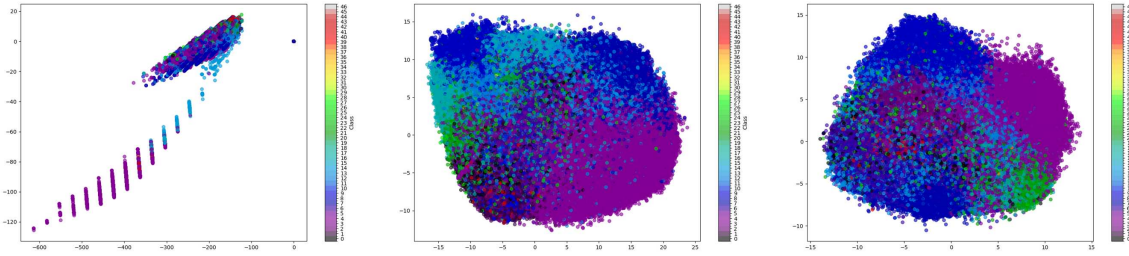


Figure 7.12: TC plots of subgraph 3, before applying Procrustes.

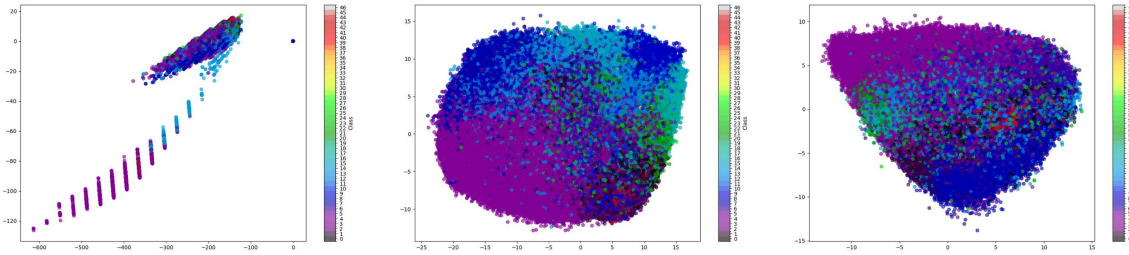


Figure 7.13: TC plots of subgraph 3, after applying Procrustes.

structural information. These results should be interpreted as one evaluated transfer configuration rather than as a complete comparison across all possible downstream predictor families.

These results show that Procrustes is *not* uniformly beneficial. It improves Cornell $\rightarrow$ Texas, but degrades Texas $\rightarrow$ Cornell and also hurts same-graph Texas evaluation. The evidence therefore supports a limited conclusion: alignment can sometimes reduce transfer mismatch, but its effect is strongly direction-dependent and may distort discriminative structure when the source/target difference is not well modeled by a simple correctable subspace transformation.

Multi-Source Transfer to acmv9

The acmv9 experiments provide the clearest evidence about feature composition. Table 7.6 compares node-feature-only transfer against node-feature-plus-distance transfer for citationv1 + dblpv7 $\rightarrow$ acmv9.

In the reported runs, node-feature-only transfer clearly outperforms the node-feature-plus-distance setting on acmv9. This is consistent with the concern that the high-dimensional bag-

Table 7.4: Train-on-one, evaluate-on-another across subgraphs. The model is trained on subgraph 1 and then evaluated (without retraining) on other subgraphs after Procrustes alignment to the baseline frame.

Subgraph	#Anchors	Valid Acc	Test Acc
Subgraph 1a (baseline, train set)	1000	0.8551	0.7106
Subgraph 2a (TCs aligned to 1a)	1000	0.8387	0.6983
Subgraph 3a (TCs aligned to 1a)	1000	0.8383	0.6975
Subgraph 1b (baseline, train set)	100	0.8362	0.6893
Subgraph 2b (TCs aligned to 1b)	100	0.8152	0.6901
Subgraph 3b (TCs aligned to 1b)	100	0.8147	0.6888

Table 7.5: Single-source cross-dataset transfer results for the cross-attention model on Cornell and Texas. Values are single-run test accuracy and should be interpreted as exploratory rather than final.

Train → Evaluate	No Alignment	Procrustes
Cornell → Cornell	0.6216	0.5946
Cornell → Texas	0.6486	0.7027
Texas → Cornell	0.5405	0.4324
Texas → Texas	0.6757	0.5405

of-words node attributes may dominate the transferred signal, while the distance-based GC representation is much lower-dimensional.

Cornell + Wisconsin → Texas Results

Table 7.7 reports the Cornell + Wisconsin → Texas diagnostic results. The multi-source Cornell + Wisconsin → Texas experiments are informative but should be treated cautiously because implementation-level inconsistencies were noted during experimentation. We therefore retain them as diagnostic observations rather than headline evidence.

Because these runs were obtained before the implementation was fully stabilized, we do not use them to support strong claims about the relative value of train-on-many versus train-on-one on the Cornell/Texas family. Their main value is diagnostic: they show that the combined node-feature-plus-distance pipeline can be unstable enough that this limitation should be stated explicitly.

Table 7.6: Multi-source transfer results to acmv9. Values are single-run test accuracy.

Source Setting	Configuration	Test Accuracy
citationv1 + dblpv7 $\rightarrow$ acmv9	node features only	0.4594
citationv1 + dblpv7 $\rightarrow$ acmv9	node features + distance	0.2468
citationv1 + dblpv7 $\rightarrow$ acmv9	node features + distance + Procrustes	0.2548

Table 7.7: Cornell + Wisconsin $\rightarrow$ Texas results. These numbers are retained as diagnostic observations because the corresponding implementation was not fully stabilized.

Configuration	Test Accuracy
node features only	0.4865
node features + distance	0.1622
node features + distance + Procrustes	0.5946

Status of Synthetic Experiments

Synthetic graph generation is a natural next step because it would allow controlled tests of transfer under parameterized structural shifts. However, finalized synthetic Train/Val/Test results are not included in this chapter. For that reason, the synthetic setting is discussed as future work rather than as completed evidence here.

7.9 Discussion

The reported results provide a more cautious picture of cross-graph reuse than a simple success narrative would suggest. Transfer is clearly possible in some settings, but the available evidence does not support a broad claim that alignment or multi-source training reliably solves cross-view and cross-graph mismatch. Instead, the results point to three interacting factors: source-target direction, feature composition, and implementation stability.

7.9.1 When Train-on-One Appears Promising

Single-source transfer is most promising when the source and target graphs are similar enough that the learned decision rule remains meaningful after only mild feature-space distortion. The Cornell $\rightarrow$ Texas result is the clearest positive example in the reported runs: aligned evaluation reaches 0.7027 test accuracy, which is higher than the corresponding no-alignment score of 0.6486.

These results also suggest that train-on-one can be useful as a practical probe of representation reuse. When it succeeds, it shows that the learned representation and classifier have captured some transferable regularity rather than memorizing one particular anchor realization or graph instance.

7.9.2 Why Alignment Must Be Interpreted Carefully

The real-graph evidence shows that alignment is *mixed*: it can help, but it can also hurt substantially. For example, Procrustes helps Cornell $\rightarrow$ Texas, but it hurts Texas $\rightarrow$ Cornell and even reduces same-graph Texas performance. This asymmetry indicates that alignment is not a uniformly reliable correction.

A second caution is methodological. In same-graph settings, Procrustes is naturally justified because the source and target embeddings are defined on the same node set. In cross-dataset settings with disjoint node identities, the with/without-Procrustes comparisons should be interpreted as heuristic diagnostics unless explicit correspondences are defined.

7.9.3 Node Features May Currently Dominate Transfer

One of the most important findings in this chapter is that node attributes may dominate transfer behavior. The bag-of-words node features are about 1400 dimensions, whereas the distance-based GC representation is only 32 dimensions. Consistent with that concern, the strongest acmv9 result is the node-feature-only multi-source baseline (0.4594), which outperforms the node-feature-plus-distance variants by a wide margin.

This does not mean that distance information is useless. Instead, it suggests that the present way of combining node features and distance-based features may be underpowered, miscalibrated, or insufficiently regularized. It also means that claims about anchor-based transfer should be framed carefully until the node-feature contribution and distance-based contribution are disentangled more systematically.

7.9.4 What Multi-Source Training Currently Tells Us

The multi-source experiments are useful, but they should not be overinterpreted. On acmv9, training on citationv1 + dblpv7 helps relative to some single-source feature-plus-distance runs, especially when compared to the weaker citationv1-only results. However, on the Cornell/ Wisconsin/Texas family, the available multi-source distance-based experiment comes from a run with implementation-level inconsistencies. As a result, the chapter reports multi-source training as an explored and partially promising direction, not as a settled improvement over train-on-one.

7.9.5 Updated Reporting Guidance

The reporting protocol in this chapter matches the available evidence: the real-graph results are single-run Train/Val/Test numbers. Once repeated anchor sets, repeated seeds, or repeated graph draws are available, the chapter can be extended to mean $\pm$ std reporting and confidence intervals. Until then, claims should remain correspondingly modest.

7.9.6 Limitations

Several limitations remain important.

Empirical coverage. The chapter is based on a limited number of real-graph runs. Some experimental branches were not fully stabilized, and repeated-view statistics are not available.

Alignment assumptions. In disjoint-graph settings, Procrustes-style comparisons should be interpreted heuristically unless explicit node correspondences or shared landmarks are provided.

Node-feature dominance. Because the node attributes are high-dimensional and highly informative, they may mask the contribution of the distance-based structural representation, making it harder to isolate the specific benefits of anchor-induced representations.

Synthetic experiments. Although synthetic graph generation is a key motivation for this chapter, finalized synthetic results are not included here. Consequently, the empirical evidence is restricted to a limited set of real-dataset shifts.

Overall, the most defensible conclusion is that cross-graph reuse is feasible but uneven. It is best analyzed as a diagnostic problem: when transfer works, it reveals reusable structure; when it fails, the failure can come from direction-specific mismatch, feature imbalance, or implementation instability rather than from a single universal cause.

7.10 Summary

This chapter now reports two related transfer settings for anchor-based graph representations: single-source *train-on-one, evaluate-on-many* transfer and multi-source *train-on-many, evaluate-on-one* transfer. The primary reported cross-dataset real-graph experiments use the cross-attention model and focus on Cornell, Texas, citationv1, dblpv7, and acmv9.

The empirical evidence in this chapter is limited but already informative. First, Procrustes alignment is not uniformly beneficial: it improves some transfers, such as Cornell $\rightarrow$ Texas, but hurts others, such as Texas $\rightarrow$ Cornell. Second, the strongest acmv9 result comes from the node-feature-only multi-source baseline, suggesting that high-dimensional node attributes may dominate transfer more than the distance-based structural representation. Third, multi-source training is promising in some settings but is not consistently established as superior in the available runs, especially where implementation stability remains a concern.

Accordingly, the main contribution of the chapter is not a universal transfer recipe, but a clearer characterization of what works, what remains unstable, and what should be tested next. Synthetic experiments, repeated-view statistics, and more systematic ablations remain important next steps for extending the chapter.

Chapter 8

Discussion, Limitations and Future Work

8.1 Summary of Findings

This dissertation develops and evaluates a novel anchor-based approach to graph learning that does not rely exclusively on iterative message passing. Graph neural networks have been highly successful, but many practical settings expose limitations of standard GNN workflows: repeated neighborhood aggregation can be computationally expensive on large graphs, deeper propagation can introduce over-smoothing in deeper models and over-squashing of long-range information, and learned models are often tied to a particular graph, feature space, or training protocol. These limitations become especially important when node attributes are weak, unavailable, or incompatible across datasets, or when the goal is to reuse learned models across similar graph datasets or even on different parts of the same dataset. Reliable transfer across heterogeneous graph domains remains challenging, especially when source and target graphs differ in feature distributions, neighborhood structure, label semantics, or training protocols. This dissertation addresses these challenges by shifting part of the graph-learning problem from repeated message propagation to reusable structural representation construction.

The central idea of this dissertation is that graph topology itself contains reusable structural information that can be sampled, compressed, and used before downstream prediction. Instead of repeatedly propagating messages through local neighborhoods during training, the proposed framework samples global graph structure through anchor-based relationships, compresses this structure into a lower-dimensional coordinate space, and uses the resulting representations for downstream prediction tasks. In this way, graph structure is transformed into a reusable representation that can be coupled with available attributes, aligned across views, and used by conventional machine-learning models or lightweight neural networks.

The main contribution of this dissertation is to turn this structural perspective into a scalable graph-learning framework. The dissertation formulates anchor-based structural encodings as reusable graph-derived features, develops and evaluates several coordinate constructions including VC, TC, and DVC, and shows how sampled node-to-anchor distance information can be compressed, stored, aligned, and used in downstream prediction. Across node classification, link prediction, and transfer-oriented experiments, the work demonstrates that anchor-based coordinates can preserve useful local and non-local topological information while reducing dependence on deep message-passing architectures and dense all-pairs graph computations. The contribution is therefore not a single model intended to dominate all baselines, but a resource-adaptable methodology for representing graph topology, evaluating accuracy and efficiency trade-offs, and identifying when compact structural coordinates provide a practical alternative or complement to conventional GNN pipelines.

Across the preceding chapters, this dissertation demonstrates that anchor-based structural coordinates provide a practical basis for node classification, link prediction, multi-view representation learning, and transfer-oriented evaluation across anchor-induced views and related graph settings. The contribution is therefore not merely the use of anchors, but the decoupling of structural representation from downstream learning. Once computed, anchor-based coordinates can be reused, compared, aligned, and combined with different feature sources. This provides a resource-adaptable path for graph learning when full GNN training is expensive, when graph domains differ, or when raw features alone are insufficient.

The later empirical chapters evaluated this framework under both train-on-one and train-on-many settings. In the train-on-one setting, the results showed that the proposed structural representation can support competitive downstream prediction when the model is trained using data from a single source. In the train-on-many setting, the results further showed that aggregating information across multiple sources can be beneficial, although the gains were not uniform across all datasets and experimental configurations. This indicates that multi-source training is promising, but its ef-

fectiveness depends on the degree of shared structure across the participating graphs and on the model used to integrate that information.

A central motivation for the transfer experiments is that direct transfer of a conventional message-passing GNN across heterogeneous graphs remains an unsolved challenge. A GNN trained on one graph is often coupled to that graph’s feature distribution, neighborhood structure, label semantics, and training protocol; therefore, a model that performs well on one graph does not automatically provide evidence of reliable transfer to another graph. This dissertation addresses this issue by evaluating whether anchor-based structural representations provide a more reusable basis for transfer across anchor-induced views, overlapping subgraphs, and related graph datasets. Its results suggest that a more flexible mechanism for relating information across structural inputs can provide stronger performance than simpler alternatives in realistic settings. At the same time, the findings also show that improved performance should not be attributed to any single component in isolation; rather, the empirical gains come from the interaction between the anchor-based representation, the training protocol, and the final predictive architecture.

The transfer experiments further show that anchor-based structural representations can support reuse beyond a single fixed coordinate construction. Across different anchor-induced views of the same graph and across overlapping subgraphs, the dissertation demonstrates that structural coordinates derived from sampled node-to-anchor distances often preserve a stable dominant representation space. Lightweight alignment methods, especially orthogonal Procrustes alignment, can reduce view-to-view mismatch and allow a predictor trained on one structural view to be evaluated on related views without full retraining. These results strengthen the central claim of the dissertation: graph topology can be converted into compact, reusable coordinates that support downstream learning while reducing dependence on repeated message passing. At the same time, the cross-dataset experiments clarify the boundary of the current framework. When source and target graphs have substantially different structural and feature distributions, alignment alone does not produce sufficiently strong or consistent gains to justify a broad claim that cross-dataset transfer is solved. Thus, the contribution is twofold: the dissertation provides positive evidence for reusable anchor-

based representations across related structural views, and it identifies cross-dataset transfer as an important but more difficult direction for future work.

Taken together, the findings across chapters support two main conclusions. First, anchor-based graph embeddings provide a viable and resource-efficient basis for graph learning. By representing nodes through sampled relationships to anchor nodes and transforming these relationships into VC, TC, or DVC coordinates, the dissertation demonstrates that graph topology can be encoded as reusable structural features rather than used only through iterative message passing. These embeddings are compatible with conventional graph-learning pipelines: they can be used as standalone structural features, combined with node attributes, or consumed by lightweight neural predictors. In the node-classification experiments, the resulting models achieve performance comparable to selected GNN baselines while using substantially fewer trainable parameters, showing that compact graph-coordinate representations can provide a practical alternative or complement to message-passing GNNs when efficiency and deployability are important.

Second, the transfer experiments show both the promise and the difficulty of reusing structural representations across views and graphs. The strongest evidence is for transfer across related structural views, especially different anchor-induced views of the same graph and overlapping sub-graphs, where alignment methods such as orthogonal Procrustes can reduce coordinate mismatch and support reuse without full retraining. Train-on-many settings and cross-attention-based models are useful tools for exploring more flexible transfer, but the results do not show that transfer is automatically solved across heterogeneous graph domains. Thus, the dissertation establishes anchor-based representations as reusable structural embeddings in related-view settings, while identifying broad cross-dataset transfer as an important open problem requiring richer alignment, stronger source-target compatibility analysis, or more adaptive transfer models.

8.2 Guidelines for Applying Anchor-Based Structural Representations

The results of this dissertation suggest that anchor-based structural graph learning should not be viewed as a single fixed method to be used uniformly in every setting. Rather, the dissertation evaluated a family of related choices: how to construct node representations from distances or coordinate relationships to selected anchors, whether to use VC, TC, or DVC as the structural encoding, whether to combine these encodings with node attributes, whether to use a shallow predictor or a more expressive model such as cross-attention, and whether to train on one graph, multiple graphs, or aligned structural views. The practical value of these choices depends on the structure of the available data, the strength of the node features, the number and compatibility of source graphs, and the computational complexity that can be supported in the downstream model. This section summarizes the experimental findings as practical guidance for choosing among these representation, training, and modeling options.

A first consideration is whether an anchor-based structural representation is appropriate for the problem at hand. The methods studied in this dissertation are most useful when graph structure carries important task-relevant information and when direct reliance on raw node attributes is either insufficient or undesirable. In such cases, anchor-based structural relationships provide a way to encode positional or relational information in a form that can be used across different graphs and learning settings. When node features are already highly informative, however, anchor-based structural coordinates should be viewed primarily as a complementary source of topological information rather than as a replacement for feature-based or message-passing models.

From an architectural perspective, the methods in this dissertation are most appropriate when one wants to decouple topology extraction from downstream prediction. Message-passing GNNs are preferable when end-to-end neighborhood aggregation and feature propagation are central to the task. Graph-coordinate neural networks, such as VCNN, TCNN, and DVCNN, are preferable when one wants a reusable structural embedding, a shallow downstream predictor, or a lower-

parameter alternative that can operate when node features are limited. They are also attractive in resource-constrained settings, such as IoT devices, mobile or edge systems, sensor networks, and distributed graph-processing environments, where memory, latency, communication cost, and energy budget may limit repeated multi-hop message passing. In such cases, anchor-based coordinates can be computed, compressed, and reused so that downstream training or prediction can proceed with less direct graph access and smaller predictive models.

A second consideration is whether to use the train-on-one or train-on-many setting. The train-on-one setting is the more appropriate choice when only one source graph is labeled, when the target deployment condition is close to a single known distribution, or when the goal is to evaluate the quality of the representation under the clearest and most controlled conditions. It is also the more conservative option when there is uncertainty about whether multiple source graphs are sufficiently related to support joint learning. In these cases, train-on-one provides a simpler experimental protocol and avoids the possibility of negative transfer from mismatched sources.

The train-on-many setting becomes more attractive when multiple related graphs are available and there is reason to expect that they share transferable structural patterns. In such cases, training on multiple sources can improve robustness and can sometimes improve downstream performance by exposing the predictor to a broader range of graph configurations. However, the empirical results also indicate that train-on-many should not be assumed to dominate train-on-one automatically. Its usefulness depends on the compatibility of the participating graphs, the consistency of the prediction task across them, and the model’s ability to integrate cross-source information without overfitting to source-specific artifacts. For this reason, train-on-many is best treated as a powerful but conditional option rather than as a universal default.

With respect to the downstream predictor, the cross-dataset real-graph transfer experiments in this dissertation support using the cross-attention model as a strong higher-capacity choice when performance is the main objective and sufficient computational resources are available. Among the evaluated cross-dataset configurations, the cross-attention model provided the strongest main results, suggesting that a flexible node-anchor interaction mechanism can be helpful when learn-

ing from anchor-based structural representations across related datasets. In practical terms, this makes cross-attention a reasonable choice for the cross-dataset real-graph transfer setting considered here, especially when the goal is to maximize empirical performance. At the same time, this recommendation should be interpreted in light of cost, complexity, and the scope of the completed experiments. Simpler predictive architectures may still be preferable in same-graph anchor-set experiments, overlapping-subgraph experiments, smaller-scale studies, ablation studies, or settings where interpretability, implementation simplicity, or computational cost is more important than obtaining the strongest possible empirical result.

The experiments also provide guidance on how to choose representation hyperparameters before worrying about downstream model complexity. Across the GCNN and VCNN studies, a useful practical rule is to tune the pipeline in three stages. First, choose the number of anchors, since this is the primary knob controlling structural coverage and the computational cost of representation construction. The results suggest that only a small fraction of nodes needs to be selected as anchors in the reported large-graph experiments, and that uniform random anchor selection is a reasonable practical default. Second, once a promising anchor budget has been identified, choose the number of retained graph coordinates or virtual coordinates by looking for a saturation point in validation performance. Third, only after these representation choices have stabilized should the neural-network configuration be tuned in detail. In practice, this ordering is more efficient than jointly tuning everything at once because it first determines how much structural information is available and then determines how much model capacity is actually needed to use it.

The empirical results suggest several concrete implications of this three-step procedure. On OGBN-Products, performance improved substantially as the number of coordinates increased from very small values, but gains diminished once the representation became sufficiently expressive. On OGBN-Proteins, the same pattern appeared across both TC-based and VC-based experiments: once a moderate number of coordinates had been reached, deeper or wider neural networks did not reliably improve performance. These findings indicate that representation quality should be tuned

before network complexity, and that overly complex downstream models can add cost without adding much predictive value.

The comparison among VC, TC, and DVC also supports a simple selection heuristic. VC is the preferred choice when preprocessing simplicity and ease of deployment matter most, since it avoids feature engineering and often works well with shallow predictors. TC is the preferred choice when stronger predictive performance is desired and an SVD-style compression step is acceptable. DVC is most attractive when one wants to reduce the number of anchors and avoid SVD, even at some cost in predictive performance. In other words, the choice among these representations should be made in light of the desired balance among preprocessing effort, parameter count, and final accuracy rather than by assuming one representation is universally superior.

The experiments also provide guidance on how to treat cross-dataset transfer. Although the dissertation explored this setting directly, the results suggest that transfer across heterogeneous graph domains remains difficult. Alignment-based approaches, including Procrustes-style alignment, did not produce gains that were sufficiently strong or consistent to justify routine use as a dependable solution. As a result, cross-dataset transfer should currently be viewed as exploratory. It is most reasonable to attempt when there is substantive evidence that the source and target graphs share similar structural organization or task semantics, and even then it should be validated carefully rather than assumed to work by default.

Overall, the practical guidance from this dissertation can be summarized as follows. Anchor-based structural representations are most useful when structure itself is a key source of information. Train-on-one is the safer and cleaner choice when data come from a single source or when cross-source compatibility is uncertain. Train-on-many is worth using when multiple related sources are available, but it should be adopted with attention to the risk of source mismatch. For the downstream predictor, cross-attention is a strong higher-capacity option for the cross-dataset real-graph transfer setting studied here, but it should not be interpreted as necessary for the same-graph anchor-set or overlapping-subgraph transfer experiments. For representation tuning, it is usually best to choose anchors first, then coordinates, and only then the neural network configuration.

Cross-dataset transfer, however, should be approached cautiously and regarded as an open research problem rather than a solved deployment strategy.

8.3 Contributions and Advantages

Before discussing the limitations of the proposed framework, it is useful to summarize what the anchor-based structural learning approach achieves. The main contribution of this dissertation is a graph-learning framework that converts topology into reusable structural coordinates before downstream prediction, rather than requiring graph structure to be consumed only through iterative message passing. By selecting anchor nodes, computing node-to-anchor structural relationships, and transforming these relationships into VC, TC, and DVC representations, the dissertation provides a practical way to encode local and non-local graph structure without constructing dense all-pairs distance matrices. The contribution is therefore not merely the use of anchors, but the systematic development of anchor-based coordinates into a reusable representation framework for graph learning.

This representation strategy provides several practical advantages. First, it decouples topology extraction from downstream learning. Once the anchor-based coordinates are computed, they can be stored, compressed, reused, concatenated with node attributes, or provided to conventional neural predictors such as MLPs and lightweight classifiers. This makes the framework useful in settings where repeated neighborhood aggregation is expensive, where node features are weak or unavailable, or where the same structural representation may be reused across multiple prediction tasks or experimental settings. Second, the framework gives explicit control over the resource and accuracy trade-off through the number of anchors, the coordinate construction, the compressed dimension, and the downstream model size. These choices make the method resource-adaptable rather than tied to a single fixed architecture.

The node-classification experiments demonstrate that anchor-based graph coordinates can preserve substantial predictive information while using much smaller downstream models than many message-passing or leaderboard-scale baselines. In particular, TC-based models on OGBN-Products

and OGBN-Proteins show favorable accuracy and efficiency trade-offs: they achieve performance comparable to or stronger than selected GCN-style baselines while using substantially fewer trainable parameters. These results do not claim to replace the highest-accuracy leaderboard models. Instead, they establish that sampled and compressed graph-distance structure can provide a competitive alternative when parameter count, memory use, computation time, or deployability are important.

The link-prediction experiments extend the same structural-coordinate idea from node-level prediction to edge-level prediction. By constructing edge representations from pairs of graph-coordinate embeddings and evaluating different coordinate variants and edge-combination functions, the dissertation shows that anchor-based coordinates can support link prediction when graph geometry is clear and structurally informative. The synthetic experiments demonstrate strong performance in controlled settings, while the real-world experiments identify where additional modeling capacity, richer structural signals, or task-specific scoring functions are needed. This is a useful contribution because it clarifies not only that graph-coordinate methods can succeed, but also the structural regimes in which their current form begins to degrade.

The transfer experiments contribute a further perspective: anchor-based representations are not only features for a single fixed task, but also structural views that can be compared, aligned, and reused. Across different anchor-induced views of the same graph and overlapping subgraphs, the results show that sampled node-to-anchor distances can preserve a stable dominant structural space. Lightweight alignment methods such as orthogonal Procrustes alignment provide a practical mechanism for reducing view-to-view mismatch and evaluating whether a predictor trained on one structural view can be reused on another. Although broad cross-dataset transfer remains difficult, the dissertation provides positive evidence that anchor-based coordinates are reusable across related structural views and gives a concrete protocol for analyzing when such reuse is feasible. Overall, the dissertation contributes a resource-adaptable structural learning perspective that connects anchor sampling, distance-based structural encoding, dimensionality reduction, neural prediction, and transfer analysis under a common methodology.

8.4 Limitations

Although the results of this dissertation provide encouraging evidence for anchor-based structural representations and their use in different training settings, several limitations should be acknowledged. These limitations define the scope of the conclusions that can be drawn and clarify which aspects of the proposed framework remain open for further research.

The first limitation concerns the scope of the empirical evaluation. The experiments were conducted on a selected set of datasets and task settings rather than on the full range of graph types encountered in practice. As a result, the findings demonstrate that the proposed approach can work under the evaluated conditions, but they do not establish that it will perform equally well across all graph domains, graph sizes, or application scenarios. In particular, graphs with very different structural properties, feature distributions, or label semantics may present challenges that are not fully captured by the benchmarks used here.

The second limitation arises from the design choices embedded in the representation pipeline. The performance of the proposed methods can depend on factors such as anchor selection, the construction of the structural representation, dimensionality reduction choices, and other implementation decisions. Many of these choices were motivated by empirical effectiveness and computational practicality rather than by a complete theoretical understanding of optimality. As a result, different design choices could potentially lead to different outcomes, and the present dissertation does not exhaustively isolate the contribution of every component or establish principled rules for choosing among all possible configurations.

The third limitation is that the reported results depend not only on the anchor-based representation itself, but also on the downstream predictive architecture used with that representation. In the cross-dataset real-graph transfer experiments, the cross-attention model served as the main higher-capacity predictor because it produced the strongest main results among the evaluated alternatives. This means that some of the empirical gains observed in the dissertation should be understood as arising from the interaction between the representation and the predictor, rather than from the representation alone. Accordingly, the contribution should be understood as an anchor-based

structural learning pipeline rather than as a standalone representation guarantee: the dissertation shows that structural coordinates can support useful prediction in the evaluated settings, while a systematic characterization of how different downstream predictors interact with these coordinates remains an open direction.

The fourth limitation concerns the train-on-many setting. Although the experiments showed that training on multiple sources can be beneficial in some cases, these gains were not uniform across all settings. This suggests that joint training across graphs is sensitive to the degree of compatibility among the source datasets and to the model’s ability to extract shared structure without being dominated by source-specific patterns. Consequently, train-on-many should not be interpreted as universally superior to train-on-one, and the present work does not provide a complete characterization of when multi-source training will help or when it may instead introduce negative transfer.

The fifth limitation is the difficulty of cross-dataset transfer. The dissertation explored transfer more directly, including alignment-based approaches such as Procrustes-style matching, but these experiments did not yield results that were sufficiently strong or consistent to support a broad claim of reliable transfer across heterogeneous graph domains. This is an important boundary on the conclusions of the work. While the experiments suggest that structural representations may contain some transferable information, they also show that transfer remains highly sensitive to distribution mismatch, graph-specific structure, and the limitations of the alignment procedure. Therefore, the dissertation should be understood as identifying cross-dataset transfer as a promising but unresolved problem for graph datasets rather than as demonstrating a fully-solved transfer framework.

The sixth limitation concerns the completeness of the scalability and computational-cost analysis. The experiments in this dissertation show that anchor-based structural representations can support downstream models with substantially fewer trainable parameters than selected message-passing GNN baselines, and the resulting predictors avoid iterative neighborhood aggregation during prediction. However, the dissertation does not provide a complete end-to-end comparison of

computational cost against GNN pipelines across all stages of processing. In particular, anchor selection, node-to-anchor distance construction, coordinate transformation, dimensionality reduction, storage, and downstream training each contribute to the total cost of the framework. While these costs were manageable in the experiments reported here, their trade-offs relative to scalable GNN training and inference were not fully characterized across graph sizes, hardware settings, and deployment scenarios. Therefore, the limitation is not that the framework lacks scalability potential, but that its full computational profile remains to be analyzed more systematically.

Finally, the dissertation is limited in the strength of the claims it can make about generality and theory. The experimental results provide evidence that the proposed framework is useful and worth further investigation, but they do not yet yield a complete theoretical account of why transfer works in some cases and fails in others, nor do they provide formal guarantees about robustness across graph families. For this reason, the main contribution of the dissertation is best understood as an empirical and methodological foundation: it establishes a workable framework, identifies settings in which the framework is promising, and clarifies several challenges that must be addressed before broader claims can be justified.

8.5 Broader Impacts and Ethical Considerations

The methods studied in this dissertation are motivated by the broader goal of improving graph learning in settings where structural information is important and direct feature-based transfer is difficult. In principle, approaches of this kind could have positive impact in domains where graphs arise naturally, including scientific networks, biological interaction networks, communication systems, and recommendation or relational data analysis. In such settings, methods that can extract useful structural representations may reduce dependence on extensive manual feature engineering and may help extend learning methods to problems where node attributes are incomplete, noisy, or unavailable.

A potential broader benefit of this work is that improved structural representations could support more data-efficient learning. If transferable structural patterns can be identified reliably, then

future systems may require fewer labeled examples in each new graph domain. This could be especially valuable in applications where labeling is expensive or requires expert knowledge. More generally, the dissertation contributes to the methodological understanding of how graph structure can be reused across settings, which may support the development of more flexible and reusable machine learning systems.

At the same time, graph-based learning methods in general raise ethical concerns because graph data often reflect human, institutional, biological, or social processes. Such data may encode bias, imbalance, exclusion, or historical artifacts. These concerns apply to many graph-learning methods, not only to the approach developed here. However, they are especially relevant for structural representations because topology itself may carry sensitive or socially meaningful information even when explicit sensitive attributes are not included. A model that learns from structural regularities may therefore reproduce or amplify patterns embedded in the graph.

A more specific concern for this dissertation is the structural transfer capability. Because the framework considers reuse across anchor-induced views, overlapping subgraphs, and graph domains, there is a risk that a representation may appear transferable even when the source and target graphs differ in important ways. The experiments in this dissertation show that broad cross-dataset transfer remains difficult and should not be assumed to work reliably without careful validation. This finding has an ethical implication: structural similarity alone should not be treated as evidence that a model is appropriate for deployment in a new domain, especially in high-stakes settings involving scientific interpretation, resource allocation, recommendation, or decisions affecting individuals.

There are also considerations related to transparency and reproducibility. The framework studied in this dissertation involves multiple stages, including anchor selection, representation construction, dimensionality reduction, and downstream modeling. While this modular structure supports systematic experimentation, it also creates several points at which design choices can influence outcomes. Responsible use of such methods therefore requires clear reporting of pre-processing choices, experimental protocols, evaluation settings, and failure cases. Reproducibility

is especially important in transfer-oriented work, where small differences in setup may lead to materially different conclusions.

A further practical consideration is computational cost. Structural representation learning and transfer experiments can require substantial computation, especially when multiple datasets, alignment procedures, or expressive predictive models are involved. Although the scale of computation in this dissertation is modest relative to very large modern machine learning systems, it is still appropriate to recognize that more extensive experimentation and larger-scale deployment would increase resource usage. Future work should therefore consider not only predictive accuracy, but also efficiency and computational sustainability.

Overall, the broader impact of this dissertation is primarily methodological. It contributes evidence that anchor-based structural representations are a promising direction for graph learning, while also highlighting the care required when extending such methods across datasets and domains. Ethically, the main implication is that structural transfer should be approached with caution: potential benefits in flexibility and data efficiency must be balanced against risks of bias propagation, spurious transfer, limited interpretability, and unwarranted confidence in settings where domain differences are substantial.

8.6 Future Work

The results of this dissertation suggest several directions for future work. These directions follow naturally from both the promising aspects of the proposed framework and the limitations identified in the previous section. Broadly, future research should aim to strengthen cross-dataset transfer, improve the flexibility and efficiency of anchor-based structural representations, and develop a clearer theoretical understanding of when and why these methods succeed.

The first direction is to develop stronger methods for cross-dataset alignment and adaptation. The experiments in this dissertation showed that transferring structural representations across heterogeneous graph domains remains difficult, and that alignment-based methods such as Procrustes-style matching are not sufficient on their own to produce consistently strong transfer. Future work

could investigate more expressive alignment strategies, including learned mapping functions, domain adaptation techniques, contrastive objectives, or methods that explicitly separate transferable structural information from graph-specific variation. Such approaches may help determine whether transfer failures arise primarily from inadequate alignment, from deeper distribution mismatch, or from limitations in the representation itself.

The second direction is to improve the construction of the anchor-based representation. Although the dissertation demonstrated that anchor-based structural information can support useful downstream learning, many representation choices remain open. Future work could explore alternative anchor selection strategies, including adaptive, task-aware, or learned anchors, rather than relying only on fixed or manually designed procedures. It would also be valuable to study whether different graph distance functions, kernel constructions, or structural summaries yield representations that are more stable, more transferable, or more informative for downstream prediction. A more systematic study of these choices could lead to representations that are both stronger empirically and easier to apply across graph families.

The third direction is to better understand multi-source learning in the train-on-many setting. The results here suggest that training on multiple source graphs can sometimes improve robustness and performance, but not uniformly. This raises an important research question: under what conditions does adding more source graphs help, and under what conditions does it instead introduce negative transfer? Future work could examine this question more systematically by characterizing source compatibility in terms of graph structure, task similarity, or representation geometry. Such analysis could lead to principled criteria for selecting source graphs, weighting their contributions, or identifying when joint training is likely to be beneficial.

The fourth direction is to expand the study of downstream predictive architectures. In this dissertation, the cross-attention model served as the primary reported model for the cross-dataset real-graph experiments because it provided the strongest main results among the evaluated alternatives. However, this does not imply that it is the final or optimal architecture for all settings. Future work could investigate whether other interaction mechanisms, hierarchical models, graph-

transformer-style predictors, or lighter-weight attention variants offer better trade-offs among performance, interpretability, and computational cost. It would also be valuable to disentangle more carefully how much of the observed benefit comes from the representation and how much comes from the expressive power of the downstream model.

The fifth direction is to improve scalability. As graph size grows, the computation required for structural representation construction, dimensionality reduction, and downstream training can become a practical bottleneck. Future work could therefore examine approximate distance computation, more efficient anchor sampling, incremental or distributed representation building, and training strategies designed for large-scale graphs. Improving scalability is important not only for efficiency, but also for evaluating the framework on broader and more realistic benchmarks than those considered in this dissertation.

The sixth direction is to strengthen the theoretical foundations of the framework. Much of the present dissertation is empirical and methodological, which is appropriate for establishing feasibility, but a deeper theoretical account is still needed. Future work could study why certain anchor-based representations preserve useful graph information, when representations learned from different anchor sets remain comparable, and what structural conditions make transfer possible or impossible. A stronger theoretical understanding could help explain the empirical patterns observed here and provide more principled guidance for model and representation design.

Finally, future work should broaden the empirical evaluation itself. Additional datasets, task types, and graph domains are needed to assess the robustness and generality of the conclusions drawn here. In particular, it would be useful to evaluate the framework on settings with different graph scales, sparsity patterns, label regimes, and domain mismatches. More extensive evaluation could clarify whether the findings of this dissertation reflect broad properties of anchor-based structural learning or are more specific to the datasets and protocols studied here.

In summary, the most important next step is to move from demonstrating feasibility toward understanding reliability. This dissertation shows that anchor-based structural representations are an extremely promising alternative for graph learning and that they merit further study under both

train-on-one and train-on-many settings. The central challenge for future work is to determine how these methods can be made more transferable, more scalable, and more principled, so that their use across diverse graph domains can be supported by stronger empirical evidence and clearer theoretical insight.

8.7 Conclusions

This dissertation argues that graph learning need not be understood only through iterative message passing. The work developed and evaluated an anchor-based structural learning framework in which graph topology is sampled through anchor relationships, compressed into coordinate representations, and reused in downstream prediction settings. By separating structural representation construction from downstream prediction, the dissertation provides an alternative way to exploit graph topology when full message-passing pipelines are costly, node features are limited, or reusable structural embeddings are desirable.

The central contribution is not simply the use of anchors, but the systematic development of anchor-based coordinates into a resource-adaptable graph-learning methodology. The results show that VC, TC, and DVC representations can preserve useful local and non-local structural information, support lightweight downstream predictors, and in selected node-classification benchmarks provide favorable accuracy and efficiency trade-offs compared with selected message-passing baselines. The link-prediction and transfer experiments further clarify the operating boundaries of the approach: anchor-based coordinates can be useful beyond a single fixed task, but more complex prediction settings and cross-dataset transfer require stronger modeling, richer alignment, and more systematic validation.

A broader lesson of the dissertation is that representation design is as important as predictor design. Message-passing GNNs remain important, but they are not the only way to use graph structure for learning. Coordinate systems, anchor-based distance features, compressed structural embeddings, and alignment procedures provide a complementary graph-learning toolbox, espe-

cially for settings where computation, memory, latency, energy budget, or node-feature quality impose practical constraints.

The work therefore provides both concrete methods and a research direction. It demonstrates that anchor-based structural representations can be effective and computationally manageable in important graph-learning regimes, while also identifying open problems in cross-graph transfer, adaptive anchor selection, scalable representation construction, richer structural signals, and theoretical analysis of alignment and representation stability. These open questions are not separate from the contribution of the dissertation; they define the next stage of research toward graph-learning systems that are efficient, reusable, and structurally informed.

Bibliography

- [1] Anura P Jayasumana, Randy Paffenroth, Gunjan Mahindre, Sridhar Ramasamy, and Kelum Gajamannage. Network topology mapping from partial virtual coordinates and graph geodesics. *IEEE/ACM Transactions on Networking*, 27(6):2405–2417, 2019.
- [2] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.
- [3] Will Hamilton, Zhitaoy Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems*, 30, 2017.
- [4] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.
- [5] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1263–1272. PMLR, 2017.
- [6] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2020.
- [7] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI open*, 1:57–81, 2020.
- [8] Chen Cai and Yusu Wang. A note on over-smoothing for graph neural networks. *arXiv preprint arXiv:2006.13318*, 2020.

- [9] Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M. Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. In *International Conference on Learning Representations*, 2022.
- [10] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-*gn*: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 257–266, 2019.
- [11] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Graphsaint: Graph sampling based inductive learning method. In *International Conference on Learning Representations*, 2020.
- [12] Roy Schwartz, Jesse Dodge, Noah A Smith, and Oren Etzioni. Green AI. *Communications of the ACM*, 63(12):54–63, 2020.
- [13] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687*, 2020.
- [14] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. In *Advances in Neural Information Processing Systems*, volume 31, pages 5165–5175, 2018.
- [15] Dulanjalie C. Dhanapala and Anura P. Jayasumana. Topology preserving maps from virtual coordinates for wireless sensor networks. In *Proceedings of the 35th IEEE Conference on Local Computer Networks*, pages 136–143, 2010.
- [16] D. C. Dhanapala and A. P. Jayasumana. Directional virtual coordinate systems for wireless sensor networks. In *2011 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2011.
- [17] Dulanjalie C. Dhanapala and Anura P. Jayasumana. Geo-logical routing in wireless sensor networks. In *Proceedings of IEEE SECON*, pages 305–313, 2011.

- [18] Dulanjalie C. Dhanapala, Anura P. Jayasumana, and Sahil Mehta. On boundary detection of 2-d and 3-d wireless sensor networks. In *Proceedings of IEEE GLOBECOM*, pages 1–5, 2011.
- [19] Dulanjalie C. Dhanapala and Anura P. Jayasumana. Clueless nodes to network-cognizant smart nodes: Achieving network awareness in wireless sensor networks. In *Proceedings of IEEE Consumer Communications and Networking Conference*, pages 174–179, 2012.
- [20] Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*, 2020.
- [21] Zhenqin Wu, Bharath Ramsundar, Evan N. Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S. Pappu, Karl Leswing, and Vijay Pande. Moleculenet: A benchmark for molecular machine learning. *Chemical Science*, 9(2):513–530, 2018.
- [22] Thomas Moscibroda, Regina O’Dell, Mirjam Wattenhofer, and Roger Wattenhofer. Virtual coordinates for ad hoc and sensor networks. In *Proceedings of the 2004 Joint Workshop on Foundations of Mobile Computing*, pages 8–16, 2004.
- [23] Qing Cao and Tarek F. Abdelzaher. A scalable logical coordinates framework for routing in wireless sensor networks. In *Proceedings of the 25th IEEE International Real-Time Systems Symposium*, pages 349–358, 2004.
- [24] Gunjan Mahindre, Rasika Karkare, Randy Paffenroth, and Anura Jayasumana. Inference in social networks from ultra-sparse distance measurements via pretrained hadamard autoencoders. In *2020 IEEE 45th Conference on Local Computer Networks (LCN)*, pages 256–266. IEEE, 2020.
- [25] Dulanjalie C. Dhanapala and Anura P. Jayasumana. Anchor selection and topology preserving maps in wsns - a directional virtual coordinate based approach. In *IEEE 36th Conference on Local Computer Networks (LCN 2011)*, pages 571–579. IEEE Computer Society, 2011.

- [26] Petar Veličković. Message passing all the way up. *arXiv preprint arXiv:2202.11097*, 2022.
- [27] Haitz Sáez de Ocáriz Borde, Artem Lukoianov, Anastasis Kratsios, Michael M Bronstein, and Xiaowen Dong. Scalable message passing neural networks: no need for attention in large graph representation learning. 2024.
- [28] Zhihao Shi, Jie Wang, Zhiwei Zhuang, Xize Liang, Bin Li, and Feng Wu. Accurate and scalable graph neural networks via message invariance. *arXiv preprint arXiv:2502.19693*, 2025.
- [29] Yassine Abbahaddou, Fragkiskos D Malliaros, Johannes F Lutzeyer, and Michalis Vazirgianis. Admp-gnn: Adaptive depth message passing gnn. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, pages 4–13, 2025.
- [30] Antonis Vasileiou, Stefanie Jegelka, Ron Levie, and Christopher Morris. Survey on generalization theory for graph neural networks. *arXiv preprint arXiv:2503.15650*, 2025.
- [31] Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. Spectral clustering with graph neural networks for graph pooling. In *International Conference on Machine Learning*, pages 874–883. PMLR, 2020.
- [32] Minyi Dai, Mehmet F Demirel, Yingyu Liang, and Jia-Mian Hu. Graph neural networks for an accurate and interpretable prediction of the properties of polycrystalline materials. *NPJ Computational Materials*, 7(1):103, 2021.
- [33] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.
- [34] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

- [35] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-GCN: Geometric graph convolutional networks. In *International Conference on Learning Representations*, 2020.
- [36] Jie Chen, Tengfei Ma, and Cao Xiao. FastGCN: Fast learning with graph convolutional networks via importance sampling. In *International Conference on Learning Representations*, 2018.
- [37] Fabrizio Frasca, Emanuele Rossi, Davide Eynard, Benjamin Chamberlain, Michael M. Bronstein, and Federico Monti. SIGN: Scalable inception graph neural networks. In *ICML Workshop on Graph Representation Learning and Beyond*, 2020.
- [38] Aleksandar Bojchevski, Johannes Gasteiger, Bryan Perozzi, Amol Kapoor, Martin Blais, Benedek Rózemberczki, Michal Lukasik, and Stephan Günnemann. Scaling graph neural networks with approximate PageRank. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2464–2473. ACM, 2020.
- [39] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? *Advances in neural information processing systems*, 34:28877–28888, 2021.
- [40] Vijay Prakash Dwivedi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Graph neural networks with learnable structural and positional representations. *arXiv preprint arXiv:2110.07875*, 2021.
- [41] Ladislav Rampášek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. *Advances in Neural Information Processing Systems*, 35:14501–14515, 2022.

- [42] Yinan Huang, William Lu, Joshua Robinson, Yu Yang, Muhan Zhang, Stefanie Jegelka, and Pan Li. On the stability of expressive positional encodings for graphs. *arXiv preprint arXiv:2310.02579*, 2023.
- [43] Charilaos I Kanatsoulis, Evelyn Choi, Stefanie Jegelka, Jure Leskovec, and Alejandro Ribeiro. Learning efficient positional encodings with graph neural networks. *arXiv preprint arXiv:2502.01122*, 2025.
- [44] Sohir Maskey, Ali Parviz, Maximilian Thiessen, Hannes Stärk, Ylli Sadikaj, and Haggai Maron. Generalized laplacian positional encoding for graph representation learning. *arXiv preprint arXiv:2210.15956*, 2022.
- [45] Vijay Prakash Dwivedi, Ladislav Rampášek, Michael Galkin, Ali Parviz, Guy Wolf, Anh Tuan Luu, and Dominique Beaini. Long range graph benchmark. In *Advances in Neural Information Processing Systems*, volume 35, 2022.
- [46] Devin Kreuzer, Dominique Beaini, William L. Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking graph transformers with spectral attention. In *Advances in Neural Information Processing Systems*, volume 34, 2021.
- [47] Derek Lim, Joshua David Robinson, Lingxiao Zhao, Tess Smidt, Suvrit Sra, Haggai Maron, and Stefanie Jegelka. Sign and basis invariant networks for spectral graph representation learning. In *International Conference on Learning Representations*, 2023.
- [48] Haorui Wang, Haoteng Yin, Muhan Zhang, and Pan Li. Equivariant and stable positional encoding for more powerful graph neural networks. In *International Conference on Learning Representations*, 2022.
- [49] Sunil Nishad, Shubhangi Agarwal, Arnab Bhattacharya, and Sayan Ranu. Graphreach: Position-aware graph neural network using reachability estimations. *arXiv preprint arXiv:2008.09657*, 2020.

- [50] Jiaxuan You, Rex Ying, and Jure Leskovec. Position-aware graph neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7134–7143. PMLR, 2019.
- [51] Pan Li, Yanbang Wang, Hongwei Wang, and Jure Leskovec. Distance encoding: Design provably more powerful neural networks for graph representation learning. In *Advances in Neural Information Processing Systems*, volume 33, 2020.
- [52] Leonardo F. R. Ribeiro, Pedro H. P. Savarese, and Daniel R. Figueiredo. struc2vec: Learning node representations from structural identity. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 385–394. ACM, 2017.
- [53] Claire Donnat, Marinka Zitnik, David Hallac, and Jure Leskovec. Learning structural node embeddings via diffusion wavelets. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1320–1329. ACM, 2018.
- [54] Dulanjalie C. Dhanapala and Anura P. Jayasumana. Topology preserving maps-extracting layout maps of wireless sensor networks from virtual coordinates. *IEEE/ACM Transactions on Networking*, 22(3):784–797, 2014.
- [55] Dulanjalie C. Dhanapala and Anura P. Jayasumana. Dimension reduction of virtual coordinate systems in wireless sensor networks. In *Proceedings of IEEE GLOBECOM*, pages 1–6, 2010.
- [56] Dulanjalie C. Dhanapala and Anura P. Jayasumana. Convex subspace routing (csr): Routing via anchor-based convex virtual subspaces in sensor networks. *Computer Communications*, 34(18):2147–2161, 2011.
- [57] Gunjan S Mahindre and Anura P Jayasumana. Link dimension and exact construction of graphs from distance vectors. *Discrete Applied Mathematics*, 309:160–171, 2022.

- [58] Zheyi Qin, Randy Paffenroth, and Anura P Jayasumana. Graph coordinates and conventional neural networks-an alternative for graph neural networks. In *2023 IEEE International Conference on Big Data (BigData)*, pages 4456–4465. IEEE, 2023.
- [59] Md Joshem Uddin, Astrit Tola, Varin Sikand, Cuneyt Gurcan Akcora, and Baris Coskunuzer. Scnode: Spatial and contextual coordinates for graph representation learning. *arXiv preprint arXiv:2410.02158*, 2024.
- [60] My Le, Luana Ruiz, and Souvik Dhara. Local distance-preserving node embeddings and their performance on random graphs. *arXiv e-prints*, pages arXiv–2504, 2025.
- [61] Zheyi Qin, Anura P Jayasumana, and Randy Paffenroth. Virtual-coordinate based sampling and embedding for machine learning with graph data. In *2024 International Conference on Machine Learning and Applications (ICMLA)*, pages 1192–1197. IEEE, 2024.
- [62] Haihong Zhao, Zhixun Li, Chenyi Zi, Aochuan Chen, Fugee Tsung, Jia Li, and Jeffrey Xu Yu. A survey of cross-domain graph learning: Progress and future directions. *arXiv preprint arXiv:2503.11086*, 2025.
- [63] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay S. Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. In *International Conference on Learning Representations*, 2020.
- [64] Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. GCC: Graph contrastive coding for graph neural network pre-training. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1150–1160. ACM, 2020.
- [65] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

- [66] Yufei He, Yuan Sui, Xiaoxin He, and Bryan Hooi. Unigraph: Learning a unified cross-domain foundation model for text-attributed graphs. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*, pages 448–459, 2025.
- [67] Divyansha Lachi, Mehdi Azabou, Vinam Arora, and Eva L Dyer. Graphfm: A generalist graph transformer that learns transferable representations across diverse domains. *Transactions on Machine Learning Research*, 2025.
- [68] Zhe-Rui Yang, Jindong Han, Chang-Dong Wang, and Hao Liu. Graphlora: Structure-aware contrastive low-rank adaptation for cross-graph transfer learning. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*, pages 1785–1796, 2025.
- [69] Li Ju, Xingyi Yang, Qi Li, and Xinchao Wang. Graphbridge: Towards arbitrary transfer learning in gnns. *arXiv preprint arXiv:2502.19252*, 2025.
- [70] Chen Wang, Yueqing Liang, Zhiwei Liu, Tao Zhang, and Philip S Yu. Pre-training graph neural network for cross domain recommendation. In *2021 IEEE Third International Conference on Cognitive Machine Intelligence (CogMI)*, pages 140–145. IEEE, 2021.
- [71] Jingzhe Liu, Haitao Mao, Zhikai Chen, Bingheng Li, Wenqi Fan, Mingxuan Ju, Tong Zhao, Neil Shah, and Jiliang Tang. One model for one graph: A new perspective for pretraining with cross-domain graphs. *arXiv preprint arXiv:2412.00315*, 2024.
- [72] Taraneh Younesian, Daniel Daza, Emile van Krieken, Thiviyan Thanapalasingam, and Peter Bloem. Grapes: Learning to sample graphs for scalable graph neural networks. *arXiv preprint arXiv:2310.03399*, 2023.
- [73] Weichen Zhao, Tiande Guo, Xiaoxi Yu, and Congying Han. A learnable sampling method for scalable graph neural networks. *Neural Networks*, 162:412–424, 2023.
- [74] Xinyi Gao, Wentao Zhang, Junliang Yu, Yingxia Shao, Quoc Viet Hung Nguyen, Bin Cui, and Hongzhi Yin. Accelerating scalable graph neural network inference with node-adaptive

- propagation. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pages 3042–3055. IEEE, 2024.
- [75] Mucong Ding, Tahseen Rabbani, Bang An, Evan Wang, and Furong Huang. Sketch-gnn: Scalable graph neural networks with sublinear training complexity. *Advances in neural information processing systems*, 35:2930–2943, 2022.
- [76] Rui Xue, Tong Zhao, Neil Shah, and Xiaorui Liu. Haste makes waste: A simple approach for scaling graph neural networks. *arXiv preprint arXiv:2410.05416*, 2024.
- [77] Haolin Li, Haoyu Wang, and Luana Ruiz. Graph sampling for scalable and expressive graph neural networks on homophilic graphs. In *2025 33rd European Signal Processing Conference (EUSIPCO)*, pages 2397–2401. IEEE, 2025.
- [78] Anton Tsitsulin, Benedek Rozemberczki, John Palowitch, and Bryan Perozzi. Synthetic graph generation to benchmark graph learning. *arXiv preprint arXiv:2204.01376*, 2022.
- [79] Vijay Prakash Dwivedi, Chaitanya K. Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *Journal of Machine Learning Research*, 24(43):1–48, 2023.
- [80] Jiaxuan You, Zhitao Ying, and Jure Leskovec. Design space for graph neural networks. In *Advances in Neural Information Processing Systems*, volume 33, 2020.
- [81] John Palowitch, Anton Tsitsulin, Brandon Mayer, and Bryan Perozzi. Graphworld: Fake graphs bring real insights for gnns. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*, pages 3691–3701, 2022.
- [82] Ming-Yi Hong, Yi-Hsiang Huang, Shao-En Lin, You-Chen Teng, Chih-Yu Wang, and Che Lin. Synhing: Synthetic heterogeneous information network generation for graph learning and explanation. *arXiv preprint arXiv:2401.04133*, 2024.

- [83] Dongzhuoran Zhou, Evgeny Kharlamov, and Egor V. Kostylev. GLora: A benchmark to evaluate the ability to learn long-range dependencies in graphs. In *The Thirteenth International Conference on Learning Representations*, 2025.
- [84] Alireza Dizaji, Benedict Aaron Tjandra, Mehrab Hamidi, Shenyang Huang, and Guillaume Rabusseau. T-grab: A synthetic diagnostic benchmark for learning on temporal graphs, 2025.
- [85] Luca Miglior, Matteo Tollosi, Alessio Gravina, and Davide Bacciu. Can you hear me now? a benchmark for long-range graph propagation. In *The Fourteenth International Conference on Learning Representations*, 2026.
- [86] Jianan Zhao, Meng Qu, Chaozhuo Li, Hao Yan, Qian Liu, Rui Li, Xing Xie, and Jian Tang. Learning on large-scale text-attributed graphs via variational inference. *arXiv preprint arXiv:2210.14709*, 2022.
- [87] Zhihao Shi, Jie Wang, Fanghua Lu, Hanzhu Chen, Defu Lian, Zheng Wang, Jieping Ye, and Feng Wu. Label deconvolution for node representation learning on large-scale attributed graphs against learning bias. *arXiv preprint arXiv:2309.14907*, 2023.
- [88] Aditya Grover and Jure Leskovec. Node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge discovery and Data Mining*, pages 855–864, 2016.
- [89] Houyi Li, Zhihong Chen, Zhao Li, Qinkai Zheng, Peng Zhang, and Shuigeng Zhou. GIPA: A general information propagation algorithm for graph learning. In *International Conference on Database Systems for Advanced Applications*, pages 465–476. Springer, 2023.
- [90] Guohao Li, Chenxin Xiong, Ali Thabet, and Bernard Ghanem. DeeperGCN: All you need to train deeper GCNs. *arXiv preprint arXiv:2006.07739*, 2020.
- [91] Ziqi Liu, Zhengwei Wu, Zhiqiang Zhang, Jun Zhou, Shuang Yang, Le Song, and Yuan Qi. Bandit samplers for training graph neural networks. *Advances in Neural Information Processing Systems*, 33:6878–6888, 2020.

- [92] David Liben-Nowell and Jon Kleinberg. The link prediction problem for social networks. In *Proceedings of the twelfth international conference on Information and knowledge management*, pages 556–559, 2003.
- [93] Lada A Adamic and Eytan Adar. Friends and neighbors on the web. *Social networks*, 25(3):211–230, 2003.
- [94] Open Graph Benchmark (OGB). Node property prediction leaderboard, n.d. Retrieved June-6th-2023.
- [95] Linyuan Lü and Tao Zhou. Link prediction in complex networks: A survey. *Physica A: statistical mechanics and its applications*, 390(6):1150–1170, 2011.
- [96] Wolfgang Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A*, 32(5):922–923, 1976.

Appendix A

Justification for Orthogonal Procrustes Alignment

Given matrices $X, Y \in \mathbb{R}^{n \times d}$, the orthogonal Procrustes problem seeks

$$R^* = \arg \min_{R \in \mathbb{R}^{d \times d}, R^\top R = I} \|XR - Y\|_F^2 \quad (\text{A.1})$$

i.e., an orthogonal matrix R that best maps X to Y in Frobenius norm. In our setting, each local graph view v produces an embedding $X^{(v)} \in \mathbb{R}^{n \times d}$, where n is the number of nodes and d is the number of anchors and orthogonal Procrustes (OP) is applied on the shared nodes between two views v_i and v_0 .

Let $Z \in \mathbb{R}^{n \times d}$ denote an existing latent Euclidean embedding of the graph. We explicitly truncate to a low embedding dimension d by keeping the top- d principal components of the corresponding distance-based kernel.

We recall a perturbation bound for orthogonal Procrustes due to Arias-Castro et al. (2020). Consider two tall matrices $X, Y \in \mathbb{R}^{m \times d}$ of the same size, with X full rank, and define

$$\varepsilon^2 = \|YY^\top - XX^\top\|_p$$

for some Schatten p -norm. Then Theorem 1 in Arias-Castro shows that

$$\min_{Q \in O(d)} \|Y - XQ\|_p \leq \left(\|X^\dagger\| \varepsilon^2 + (1 - \|X^\dagger\|^2 \varepsilon^2)^{-1/2} \|X^\dagger\| \varepsilon^2 \right) \wedge (d^{1/2p} \varepsilon), \quad (\text{A.2})$$

and, in particular, if $\|X^\dagger\| \varepsilon \leq 2^{-1/2}$, then

$$\min_{Q \in O(d)} \|Y - XQ\|_p \leq (1 + \sqrt{2}) \|X^\dagger\| \varepsilon^2. \quad (\text{A.3})$$

Here $X^\dagger$ denotes the Moore–Penrose pseudo-inverse of X and $O(d)$ is the orthogonal group in $\mathbb{R}^d$. We will use this result with $p = 2$, so that $\|\cdot\|_p$ is the Frobenius norm.

In words, the Procrustes error $\min_Q \|Y - XQ\|_F$ is small whenever (i) the Gram matrices $XX^\top$ and $YY^\top$ are close and (ii) the configuration X is not close to being rank-deficient (i.e., the smallest singular value of X is not too small, so $\|X^\dagger\|$ is not too large).

Fix a reference view v_0 and another view v . Let S denote the set of nodes shared between these two views, and write

$$X := X_S^{(v_0)} \in \mathbb{R}^{m \times d}, \quad Y := X_S^{(v)} \in \mathbb{R}^{m \times d},$$

where $m = |S|$. The orthogonal Procrustes alignment between v and v_0 is then

$$R^{(v)} \in \arg \min_{R \in O(d)} \|XR - Y\|_F.$$

Theorem (A.3) yields

$$\|Y - XR^{(v)}\|_F \leq C_P \|X^\dagger\| \|YY^\top - XX^\top\|_F, \quad (\text{A.4})$$

for a universal constant $C_P \leq 1 + \sqrt{2}$, provided $\|X^\dagger\| \varepsilon \leq 2^{-1/2}$ holds (which is satisfied as long as the Gram discrepancy ε and the conditioning of X are moderate).

This directly instantiates steps (3)–(4) of our informal argument:

- ***(Step 3)*** The embedding error after alignment, $\|Y - XR^{(v)}\|_F$, is controlled by the Procrustes bound. - ***(Step 4)*** The bound in (A.4) is the product of two terms: the pseudo-inverse norm $\|X^\dagger\|$ and the Gram difference $\|YY^\top - XX^\top\|_F$.

We now relate each factor to structural properties of the graph embeddings.

Let the singular values of X be $\sigma_1(X) \geq \dots \geq \sigma_d(X) > 0$. Then $\|X^\dagger\| = 1/\sigma_d(X)$. Since X consists of the top- d principal components of a distance-derived embedding restricted to the shared nodes S , choosing d well inside the spectral bulk (rather than deep in the tail) ensures that $\sigma_d(X)$

is not extremely small. Empirically, we observe that the smallest retained singular values remain well above numerical precision across all datasets and choices of anchors. Thus

$$\|X^\dagger\| = \frac{1}{\sigma_d(X)} \leq C_{\text{cond}}, \quad (\text{A.5})$$

for a moderate constant C_{cond} that depends on the spectrum of the reference embedding but not on the choice of view v .

This realizes ***(Step 5)***: a controlled smallest singular value implies a small pseudo-inverse norm.

Appendix B

Codebase

We implemented different algorithms, tools, and experimental pipelines throughout this dissertation to support the graph-coordinate-based learning methods described in the main chapters. This appendix provides information about the source code repository and the major components included in the codebase.

The source code associated with this dissertation is available in the following GitHub repository:

https://github.com/i721/Dissertation_Codes

The repository includes implementation files and scripts for generating graph-coordinate representations, training neural-network models, evaluating model performance, and conducting the experimental analyses reported in this dissertation. The main text provides the methodological descriptions, experimental settings, and reported results, while the repository provides supplemental implementation details.

B.1 Graph Coordinate Generation

The codebase includes scripts for constructing anchor-based graph coordinate representations, including virtual coordinates, topology coordinates, and directional virtual coordinates. These scripts support the coordinate-generation procedures described in the corresponding methodology chapters.

B.2 Model Training and Evaluation

The repository also includes neural-network training and evaluation scripts used for the node classification and link prediction experiments. These scripts implement the model configurations,

data loading procedures, and evaluation protocols used to obtain the results reported in this dissertation.

B.3 Transfer and Alignment Experiments

For the transferability studies, the codebase includes scripts for generating multiple graph views, applying coordinate alignment procedures, and evaluating trained models across different coordinate systems or graph subsets. These materials correspond to the transfer and Procrustes-alignment experiments discussed in the main text.

B.4 Supplemental Implementation Notes

The repository is provided as supplemental material to help readers inspect and reuse the implementation. Future maintenance updates to the repository should be interpreted as supplemental code maintenance and do not modify the experimental results reported in this dissertation.