THESIS

GRAPH FEATURE ENGINEERING AND COORDINATE-BASED LEARNING FOR TRANSFERABLE AND ENERGY-EFFICIENT ARTIFICIAL INTELLIGENCE

Submitted by

Hansi Yasodara

Department of Electrical and Computer Engineering

In partial fulfillment of the requirements

For the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Spring 2026

Master's Committee:

Advisor: Anura Jayasumana

Sudeep Pasricha
Indrakshi Ray

ABSTRACT

GRAPH FEATURE ENGINEERING AND COORDINATE-BASED LEARNING FOR TRANSFERABLE AND ENERGY-EFFICIENT ARTIFICIAL INTELLIGENCE

A comprehensive framework for efficient and scalable graph representation learning is presented, emphasizing coordinate-based and explicit structural methods. The research addresses the limitations of Graph Neural Networks (GNNs) in resource-constrained environments, including edge devices and large-scale deployments, by developing lightweight, non-neural alternatives.

The first contribution is the Network Feature Embedding (NFE) pipeline, which integrates diffusion-based, positional, and structural descriptors into a unified representation for node classification. The second contribution is the Topology Coordinate-Driven Random Forests (TC-DRF) framework, which combines anchor-based topology coordinates with Random Forest classifiers for graph-level learning and cross-dataset transfer.

Extensive evaluations of NFE and TC-DRF on vision, molecular, and social graph benchmarks demonstrate competitive predictive performance while substantially reducing computational overhead, memory footprint, and energy consumption. The proposed frameworks enable zero-shot cross-dataset transfer, maintain robustness under class imbalance, and support practical deployment in Green AI settings. Edge-device experiments, including deployment on Raspberry Pi hardware, confirm sub-millisecond inference latency and ultra-low energy usage.

This research challenges the prevailing reliance on deep message-passing architectures for graph learning, demonstrating that explicit structural representations coupled with lightweight models provide viable, interpretable, and resource-efficient alternatives. The findings contribute to the advancement of scalable and sustainable graph learning methodologies and establish a foundation for future work in structural embeddings, dynamic graph analysis, and hybrid structural-attribute learning models.

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Professor Anura P. Jayasumana, for his guidance, support, and feedback throughout this research.

I am grateful to my thesis committee members for agreeing to serve on the committee and for the time and effort they spent reviewing this work and providing helpful comments.

I would also like to thank the instructors of the courses I followed during my graduate studies, whose lectures and discussions contributed greatly to my understanding of the subject areas related to this thesis.

This work used the Anvil high-performance computing system at Purdue University through the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program under allocation `CIS230184`. I also acknowledge the Colorado State University Graduate School for providing teaching assistantship opportunities, which supported me during my studies.

I would like to thank the Department of Electrical and Computer Engineering at Colorado State University and the graduate coordination team, with special thanks to Katya, the Graduate Coordinator, for her continued help and responsiveness whenever I had questions.

I gratefully acknowledge the University of Kelaniya, Sri Lanka, for institutional support during my period of study leave, which enabled me to pursue this graduate program.

I am deeply thankful to my parents, my grandparents especially my grandfather and Manika Loku Amma for their constant encouragement and belief in me. I also thank my boyfriend, my friends, and my Sri Lankan friends living in Fort Collins for their support and companionship throughout this journey.

Finally, I thank everyone who supported me, directly or indirectly, during the completion of this work.

DEDICATION

This thesis is lovingly dedicated to my parents, to my boyfriend Kasun Gamage, and to Twig, who is gone yet never forgotten.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Many real-world systems are naturally organised as networks of interacting entities. Examples include social interactions among individuals, biological relationships between proteins, citation links between scientific articles, transportation infrastructures, and sensor connectivity in IoT environments [37]. These systems are commonly modelled as graphs, where nodes represent entities such as users, proteins, documents, or sensors, and edges encode interactions, similarities, or physical connections. Such representations make it possible to analyse how structure, influence, and connectivity shape system behaviour.

Building on this relational structure, graphs serve as a fundamental abstraction for representing and organising interactions among entities. They allow relational information to be stored, queried, and analysed in a structured manner. Beyond their role as knowledge representations, graphs have become increasingly important in modern machine learning, where predictive tasks are often defined over graph-structured data. Many learning problems require reasoning over both entity attributes and the underlying network structure. Examples include classifying protein functions in biological interaction networks, predicting the roles of individuals in collaboration graphs, recommending connections in social networks, and identifying new therapeutic uses of drug molecules whose chemical structure can be modelled as a graph. Addressing such tasks requires effective ways to represent graph structure in a form that can be processed by standard machine learning models [31].

The scale and complexity of graph-structured data continue to grow rapidly [56]. This growth is particularly evident in large-scale IoT and cyber-physical systems, where data is generated by vast numbers of distributed devices and naturally resides on communication and interaction networks. In such settings, sensors, devices, and services form dynamic graphs whose structure evolves over time and spans wide geographic and organisational boundaries. As a result, there is increasing demand for graph-based learning methods that can extract meaningful patterns from

large networks while operating under practical constraints such as limited computation, memory, and energy budgets [50]. These requirements have made graph-based machine learning a critical and active research area, especially for real-world deployments in edge and IoT environments.

GNNs have become a dominant approach for learning on graph-structured data [106]. They rely on message passing to aggregate information from neighbouring nodes [33]. While GNNs achieve strong performance on many benchmarks, they introduce several practical limitations.

First, capturing rich topological information often requires multiple rounds of message passing [88]. For large or complex graphs, information must be propagated across long distances, which increases computational cost, memory usage, and inference time. As the number of layers grows, GNNs become increasingly dependent on specialised hardware such as GPUs, making them less suitable for resource-constrained settings.

Second, repeated neighbourhood aggregation can lead to oversmoothing, where node representations become increasingly similar across layers. This effect reduces the model's ability to distinguish nodes with different structural roles or functional importance, thereby limiting expressive power and potentially degrading predictive performance[88].

Third, many GNN architectures impose substantial resource requirements in terms of model size, memory footprint, and energy consumption. While most graph learning benchmarks emphasise accuracy, such metrics alone do not capture the practical cost of deployment. In mobile, edge, or IoT environments, limited storage, constrained RAM, real-time latency requirements, and power budgets play a critical role [78, 81]. Even when GNN models can be deployed on such platforms, they may be inefficient or impractical for continuous or large-scale operation.

These limitations indicate that, while GNNs are effective for many graph learning tasks, they are not universally suitable across all deployment scenarios. They motivate a different direction for machine learning on graphs, one that emphasises efficient use of graph topology through explicit, topology-aware representations, rather than deep message passing. By transforming graph structure into compact and informative embeddings, such approaches can directly benefit from

advances in standard machine-learning and neural-network techniques, while enabling efficient inference and practical deployment in edge and IoT environments.

Building on these deployment challenges, the carbon cost of machine learning has also become an increasingly important concern. Resource-intensive models, particularly large neural networks, consume substantial energy during both training and deployment [68]. This raises sustainability issues, especially when such models are used on embedded platforms or in large-scale IoT systems that operate continuously and at scale.

Recent studies show that the computational and energy demands of modern machine learning methods continue to grow rapidly [36]. Despite this trend, many works still do not report energy consumption or carbon emissions, limiting transparency and hindering mitigation efforts. Moreover, carbon emissions associated with model execution can vary widely depending on the underlying energy grid, with differences of up to a factor of thirty across regions [36]. These factors further motivate the use of lightweight and energy-efficient models, which are more suitable for practical and sustainable deployment.

This thesis demonstrates that graph topology alone can support strong predictive performance without the computational cost of message passing. It develops lightweight and deterministic embedding methods that directly leverage graph structure, and shows how simple machine-learning models can effectively operate on these representations. Together, these contributions enable efficient and low-power graph learning suitable for deployment in edge and IoT environments.

1.1 Motivation

Despite their success, GNNs exhibit limitations that restrict their use in many real-world deployments. Message passing aggregates information locally at each layer, which makes capturing long-range dependencies computationally expensive on large graphs. Increasing the number of layers raises memory usage and inference time, and often leads to oversmoothing, where node representations lose discriminative power [88]. In addition, most GNN architectures depend on GPU acceleration for efficient training and inference, which is not available on many edge, mobile,

or embedded platforms [78, 81]. These constraints limit the practicality of GNNs in resource-constrained environments.

Energy consumption further exacerbates these challenges. Large neural models incur substantial energy and carbon costs during both training and deployment, raising sustainability concerns for long-running or large-scale systems [45]. This has motivated growing interest in *Green AI*, which prioritises computational efficiency and energy awareness alongside predictive performance, in contrast to accuracy-centric *Red AI* approaches [4, 90]. Energy-efficient models are particularly critical for IoT and embedded systems that operate under limited power budgets or rely on intermittent energy sources.

Existing non-neural graph learning methods also face important limitations. Random-walk-based embeddings such as *node2vec* [39] rely on multiple sources of stochasticity, including walk generation and optimisation procedures, making them sensitive to hyperparameter choices and prone to instability across runs [28]. Spectral approaches can capture global structure but require expensive eigenvalue computations that scale poorly with graph size [84]. Classical graph features are typically local in nature and may fail to reflect higher-order or global topological patterns. Together, these issues reduce robustness, scalability, and reliability in large or dynamic graphs.

At the same time, modern graph-structured systems continue to increase in structural complexity, update frequency, and reliance on machine-learning-driven analytics. IoT networks, social platforms, and biological interaction systems generate continuously evolving graphs that must support prediction, classification, and decision-making under strict resource constraints. Recomputing or updating representations in such settings further amplifies concerns related to scalability, memory footprint, and computational efficiency [70].

These considerations motivate a different direction for graph learning: methods that leverage graph topology directly through deterministic, topology-aware embeddings, without relying on deep message passing. By transforming structural information into compact and informative representations, such approaches enable the use of simple, efficient machine-learning models while

supporting fast inference, interpretability, and low-power deployment. This shift aligns with the goals of Green AI and addresses the practical requirements of edge and IoT environments.

1.2 Contributions

This thesis consists of two complementary research that together address the challenges of scalable, transferable, and energy-efficient graph learning. The first part focuses on explicit graph feature engineering and lightweight node-level learning for AIoT and edge scenarios, while the second part investigates topology-driven graph representations combined with classical machine learning models for cross-dataset transfer and Green AI. The main contributions of this thesis are summarized as follows.

Graph Feature Engineering and Lightweight Learning for AIoT

The first part of this thesis investigates whether effective graph-based learning for resource-constrained and edge environments can be achieved without deep message-passing architectures. The key contributions of this part are as follows:

- We propose a non-neural node feature engineering (NFE) pipeline that combines diffusion-based proximity signals, Laplacian positional encodings, and structural role features to construct compact and informative node representations.
- We demonstrate that carefully engineered graph features can achieve competitive node classification performance on large-scale benchmarks such as OGBN-Arxiv and OGBN-Proteins when paired with lightweight classifiers, without relying on graph neural networks.
- We introduce multiple efficiency-oriented optimizations, including autoencoder-based feature compression, knowledge distillation, pruning, and quantization, enabling ultra-low-latency and low-energy inference suitable for edge and embedded deployments.

- We provide an extensive empirical evaluation of accuracy, latency, energy consumption, memory footprint, and carbon emissions, highlighting the trade-offs between predictive performance and deployment efficiency under Green AI constraints.

- We validate the practicality of the proposed pipeline through real-world edge deployment experiments on Raspberry Pi platforms using TensorFlow Lite, demonstrating sub-millisecond inference and sub-millijoule energy consumption.

Topology-Driven Random Forests for Transferable Graph Learning

The second part of this thesis explores whether explicit structural representations, combined with classical machine learning models, can enable effective and transferable graph learning without deep neural architectures. The key contributions of this part are:

- We adapt Topology Coordinate (TC) embeddings to diverse graph domains, including superpixel-based vision graphs, molecular graphs, and social networks, capturing both local and global structural information in a compact, fixed-dimensional form.

- We propose the Topology Coordinate-Driven Random Forest (TC-DRF) framework, which integrates anchor-based structural embeddings with Random Forest classifiers as a lightweight and interpretable alternative to deep GNNs.

- We demonstrate that TC-DRF achieves competitive predictive performance across multiple benchmarks while significantly reducing computational cost, memory footprint, and energy consumption compared to message-passing neural models.

- We show that the proposed framework supports zero-shot cross-dataset transfer learning, enabling models trained on one dataset (e.g., Pascal VOC-SP) to be directly reused on another dataset (e.g., COCO-SP) without target-domain retraining.

- We provide detailed energy, latency, and model footprint analyses, illustrating that topology-driven, non-neural representations are well suited for Green AI or edge deployment scenarios.

Together, these contributions demonstrate that explicit graph representations and classical learning models can serve as efficient, reliable, and sustainable alternatives to deep graph neural networks, particularly in large-scale, transfer-learning, and resource-constrained settings.

1.3 Impact and Applications

The research presented in this thesis has both methodological and practical impact, particularly in the context of scalable graph learning, Green AI, and edge intelligence. By demonstrating that explicit graph representations and lightweight learning models can achieve competitive performance without deep message passing, this work challenges prevailing assumptions about the necessity of GNNs for effective graph-based inference.

Impact on Graph Learning Methodology

From a methodological perspective, this thesis contributes to a growing body of work advocating for explicit, interpretable, and efficiency-aware graph representations. The proposed pipelines show that carefully engineered structural features, such as diffusion-based proximity signals, positional encodings, and topology coordinates, can capture rich graph information while avoiding the instability, opacity, and computational overhead associated with stochastic embedding methods and deep GNN architectures.

Furthermore, the demonstrated stability and transferability of topology-driven representations address a critical limitation of many embedding-based approaches, which are often sensitive to random initialization, hyperparameter choices, and dataset-specific tuning. By enabling zero-shot cross-dataset transfer and consistent performance across graph domains, this work supports more reliable and reproducible graph learning workflows, particularly in exploratory and applied settings.

Impact on Green AI and Resource-Constrained Learning

A central impact of this research lies in its contribution to Green AI. The proposed methods significantly reduce training and inference costs in terms of latency, memory footprint, energy con-

sumption, and carbon emissions. Through the use of non-neural feature engineering, classical machine learning models, and deployment-oriented optimizations such as compression, distillation, and pruning, this thesis demonstrates that sustainability and performance need not be mutually exclusive.

These results show that CPU-only inference pipelines can achieve performance comparable to, and in some cases exceeding, that of more complex neural models, while requiring substantially fewer computational resources. This demonstrates that the proposed approaches offer a practical and environmentally responsible alternative for deployment in large-scale and continuously operating systems.

Applications in Edge Intelligence and AIoT

The practical applicability of this work is most evident in edge and AIoT scenarios, where computational resources, power budgets, and latency constraints are critical. The lightweight nature of the proposed pipelines enables deployment on low-power devices such as Raspberry Pi-class hardware, making them suitable for real-time inference in embedded and distributed environments.

Potential application domains include:

- IoT networks, where graph-based representations can model device interactions, communication patterns, or anomaly propagation under strict energy constraints.
- Cyber-physical systems and smart infrastructure, where low-latency, reliable graph inference is required for monitoring, fault detection, and decision support.
- Large-scale scientific and biological networks, where interpretability, stability, and transferability are essential for downstream analysis and hypothesis generation.
- Vision and perception systems based on superpixel or region-adjacency graphs, where cross-dataset transfer without retraining can reduce deployment and maintenance costs.

Moreover, this thesis demonstrates that revisiting classical machine learning models and explicit representations can yield practical advantages in modern graph learning pipelines. By decou-

pling structural representation from deep neural inference, the proposed approaches open avenues for hybrid systems that balance accuracy, interpretability, efficiency, and sustainability. These findings encourage a shift toward more principled and resource-aware graph learning strategies, particularly as graph datasets continue to grow in size and complexity.

1.4 Thesis Outline

This thesis is organised as follows. Chapter 1 introduces the problem setting and motivates the need for efficient and deployable graph learning, and summarises the main contributions and applications. Chapter 2 reviews the state of the art in graph representation learning, covering challenges in analysing large and complex networks, strengths and limitations of GNNs, non-neural and classical approaches, node embedding methods, and deployment considerations such as energy, carbon cost, and edge/IoT constraints. Chapter 3 presents the proposed *Network Feature Embedding (NFE)* pipeline for Artificial Intelligence of Things, including the embedding components, autoencoder-based compression, DeepMLP classification, and knowledge distillation, followed by results and analysis. Chapter 4 introduces the second framework, *Topology Coordinate-Driven Random Forests*, detailing topology-coordinate construction, anchor-based embeddings, the Random Forest classifier design, extensions to additional datasets, and an evaluation that includes accuracy, energy/latency, and model footprint. Finally, Chapter 5 concludes the thesis by summarising key insights, discussing limitations, and outlining open problems and future research directions. An appendix provides additional tools and dataset details.

Chapter 2

State-of-the Art

This chapter reviews state-of-the-art research relevant to the graph machine learning problems addressed in this thesis. The focus is on existing methods for analyzing, representing, and learning from large-scale graph-structured data, with particular emphasis on scalability, representation efficiency, and computational feasibility. We survey classical and modern approaches spanning network analysis challenges, node embedding techniques, topology-aware representations, and machine learning methods for graph data. Special attention is given to the trade-offs between expressiveness, computational cost, and deployability, as these factors are central to practical graph learning in large-scale and resource-constrained environments.

2.1 Graph-Structured Data and Representation Needs

Graphs arise naturally in many systems where the behaviour or attributes of an entity is linked to the behaviour of others related to it in some form. They provide a structured way to represent dependencies, interactions, and flows of information within a network. In this setting, each entity participates in a broader relational pattern, and these patterns often determine how the system evolves or functions. Examples include communication patterns in social platforms, functional and physical connectivity in biological and chemical systems, and region-level relationships in image or sensor data. Such relational structure cannot be captured by treating observations as independent, and this motivates the use of graph-based representations in modern machine learning [100].

Modern graph datasets have grown rapidly in both size and structural complexity. Large-scale networks today often comprise millions or even billions of nodes and edges, as observed in social, communication, biological, and information networks [55]. Beyond scale, the value and expressive power of such networks increase super-linearly as new nodes and connections are added. According to Metcalfe's Law [44], the utility of a network grows proportionally to the square of

the number of nodes, reflecting the rapid increase in possible pairwise interactions. Reed's Law [44] further suggests an exponential growth in value for social and collaborative networks, as the number of potential subgroups scales as $2^N - N - 1$, where N denotes the number of participants.

This compounding growth is evident across multiple real-world graph domains. Social and communication graphs are inherently dynamic, with continuously evolving connectivity patterns [71]. Image-derived graphs, such as superpixel or region adjacency graphs, often exhibit dense local structure and fine-grained relational patterns [16]. Similarly, Internet-of-Things (IoT) and cyber-physical systems generate continuous streams of relational data in which both node attributes and graph topology may change over time [86]. As networks expand, each additional node not only increases the size of the graph but also amplifies the space of possible interactions, dependencies, and higher-order structures.

This rapid and super-linear growth places significant demands on graph representation, storage, and learning pipelines. Consequently, there is a growing need for efficient, scalable, and resource-aware graph learning approaches that can operate effectively on large, evolving graph-structured data without incurring prohibitive computational or environmental costs.

Several properties of graphs make learning more difficult than in Euclidean domains such as images or tables. Graphs lack a fixed grid structure, and their connectivity patterns vary across different datasets. Nodes may have very different numbers of neighbours, and local neighbourhoods can differ greatly in size and organisation. Many real-world graphs are sparse, with only a small fraction of possible node pairs connected, which leads to irregular and uneven structural patterns that complicate learning [99]. Edges can also carry additional information such as weights, directions, timestamps [57], or categorical labels, resulting in heterogeneous graph data [104]. Graphs have no natural ordering of nodes, so useful representations must remain invariant to permutations of node identifiers [29]. Taken together, these factors make graph learning substantially more challenging than learning from fixed-size or grid-based inputs.

There is therefore a strong need for compact, informative, and permutation-invariant graph representations. Working directly with adjacency matrices is inefficient for large networks because

memory usage increases with both the number of nodes and the number of edges [53]. To address this, graph structures are often transformed into low-dimensional vectors that preserve key relationships and structural patterns. Such representations enable machine learning models to make use of graph topology more efficiently and support scalable analysis. This process is known as graph representation learning [32].

A useful representation must preserve both local structure, such as node neighbourhoods, and global patterns, such as communities or long-range dependencies [81]. Early embedding methods based on random walks or matrix factorization capture local structure well, but often fail to encode global topology and may become unstable on large or evolving graphs [105]. Spectral approaches preserve global structure but depend on costly eigendecompositions, which limit their scalability [6]. More recent neural approaches, including GNNs and graph transformers, offer strong expressive power but require large memory, long training times, and specialized GPU hardware [79]. These requirements make many existing methods unsuitable for edge and IoT systems, which operate under strict constraints. Such systems may also need to ensure reliability in unpredictable environments, preserve user privacy by avoiding excessive data sharing, and maintain security against adversarial access [14]. They also demand low computational overhead, reduced latency, and high energy efficiency due to limited battery capacity and restricted processing power [81]. As a result, lightweight and topology-aware representations are needed to enable practical graph learning in these settings.

For these reasons, there is increasing interest in lightweight and topology-aware embeddings that encode graph structure without heavy computation. Such representations aim to compress both local and global structural signals into a fixed-length vector in a deterministic and scalable way [81]. This thesis builds on these ideas by developing efficient graph representations that support accurate learning on large networks while remaining practical for deployment on constrained devices.

2.2 Challenges in Analysing Large and Complex Networks

Analysis, inference and learning from models and data in the form of complex networks is difficult because modern networks are massive, highly interconnected, and structurally heterogeneous. These systems appear across diverse fields such as biology, social sciences, infrastructure, and the Internet, each with their own attributes and relationships. Resulting diversity and complexity introduces a range of computational and structural challenges. The wiring patterns, i.e., the network topology, encode important functional information, yet this information is deeply hidden within a large-scale and irregular graph structures, making analysis both demanding and resource-intensive.

2.2.1 Computational intractability

Many fundamental questions in network analysis are computationally hard. Ref. [85] emphasizes that even with unlimited computational power, large classes of problems on real-world networks cannot be solved exactly, necessitating the use of approximate and heuristic methods [85]. This difficulty arises because higher-order structures, such as graphlets or multi-node motifs, grow combinatorially with network size, making exact enumeration infeasible in practice. As a result, scalable approximation techniques are required, although even these can become slow and resource-intensive on large networks.

2.2.2 Scalability and Distributed Processing Bottlenecks

Modern networks routinely contain billions of nodes and trillions of edges, including web graphs, social networks, biological interaction networks, and large-scale communication systems. Distributed processing of such graphs is inherently challenging due to both their scale and the irregular, data-dependent nature of graph computations [98]. Unlike matrix-based or grid-structured workloads, graph algorithms exhibit poor spatial and temporal locality, resulting in frequent cache misses and non-sequential memory access patterns. Furthermore, power-law degree distributions introduce high-degree hub nodes that create communication hotspots, workload imbalance, and

synchronization overheads. These structural characteristics significantly limit parallel efficiency and degrade performance on both distributed clusters and single-machine systems.

To address the storage and querying challenges of large-scale graph data, graph databases have emerged as specialized data management systems [3]. In contrast to relational databases, which rely on expensive join operations to reconstruct relationships, graph databases natively represent entities and their relationships using models such as property graphs or RDF triples. Their index-free adjacency mechanism enables direct traversal between connected nodes without global index lookups, substantially improving performance for relationship-centric queries. This design makes graph databases particularly suitable for applications such as social network analysis, fraud detection, recommendation systems, and knowledge graph management.

Graph databases further provide expressive, declarative query languages that support pattern matching and traversal-based operations while abstracting low-level implementation details. However, despite these advantages, scalability remains a fundamental challenge. Distributed graph databases must partition highly connected structures across machines, minimize cross-partition communication, and maintain consistency under concurrent workloads. Inefficient partitioning can amplify communication costs and negate the benefits of native graph storage. As graph size increases, ensuring fault tolerance, transactional guarantees, and high query performance simultaneously becomes increasingly complex.

Consequently, although graph databases provide an effective abstraction layer for storing and querying graph-structured data, they do not eliminate the inherent computational bottlenecks of large-scale graph analytics. Efficient graph processing still requires careful design of partitioning strategies, communication-aware algorithms, and memory-efficient execution models.

2.2.3 Structural complexity at different scales

Understanding the structural organisation of networks is also difficult. Authors of [74] observe that while local metrics such as degree or clustering are straightforward, characterising medium and large-scale structures is substantially more complicated [74]. Networks often contain over-

lapping communities, hierarchical modularity, and multi-scale patterns that cannot be captured by simple global descriptors. Extracting these structures requires advanced algorithms, many of which become computationally prohibitive at scale.

2.2.4 Heterogeneous and multidimensional data

Large networks frequently incorporate diverse forms of information, that effective analysis must integrate social, temporal, spatial, and contextual dimensions, making modelling more complex and less uniform across datasets [34]. The coexistence of different data modalities introduces additional difficulties, such as inconsistent node or edge semantics, dynamic behaviour, and multimodal relationships that change over time.

2.2.5 Higher-order and functional interactions

Standard graph descriptors, such as node degree, are insufficient to capture functional organization. This paper [85] shows that networks with identical degree distributions can exhibit radically different internal structure, implying that important information lies in higher-order connectivity patterns [85]. Identifying these patterns requires graphlet-based or motif-based methods, which are computationally expensive and sensitive to noise.

2.2.6 Noisy, incomplete, and evolving data

Information about real-world networks is often incomplete, noisy, or dynamically changing, which significantly complicates graph analysis and learning. As highlighted in literature, missing edges, measurement errors, and spurious connections can distort core analytical tasks such as community detection, centrality estimation, and anomaly detection [74]. In many practical settings, the observed graph represents only a partial or imperfect snapshot of the underlying relational structure, leading to uncertainty in both topology and node attributes [7].

Noise and incompleteness are particularly prevalent in large-scale and data-driven graph construction processes. For example, interaction graphs derived from logs, sensors, or user behavior may suffer from sampling bias, delayed observations, or unreliable measurements. Such imper-

fections can propagate through downstream learning pipelines, amplifying errors and reducing the robustness of learned representations [7]. Methods that rely heavily on stochastic optimization or repeated random sampling are especially sensitive to these issues, often producing unstable or inconsistent outputs under small perturbations of the input data [7].

Dynamic networks introduce additional challenges by violating the assumption of stationarity. In evolving graphs, both node attributes and connectivity patterns may change over time, requiring learning approaches that can adapt to temporal drift, incremental updates, and streaming data [7]. Recomputing representations from scratch is often computationally infeasible for large graphs, motivating the need for efficient, robust, and update-friendly graph learning methods that can operate reliably under noisy, incomplete, and non-stationary conditions.

Together, these characteristics highlight the importance of developing graph learning frameworks that explicitly account for uncertainty, partial observability, and temporal evolution, rather than assuming clean, static, and fully observed network structures.

2.2.7 Energy, memory, and processing constraints

Large-scale graph analysis is fundamentally constrained by practical limits on memory capacity, data movement, and energy consumption. In many real-world settings, graphs are too large to fit entirely in main memory, necessitating out-of-core execution strategies or SSD-based processing pipelines, which introduce significant latency and energy overheads [98]. Even when graphs reside in memory, their irregular and sparse access patterns severely stress conventional memory hierarchies.

Graph workloads are dominated by fine-grained, random memory accesses arising from pointer chasing and irregular neighborhood traversal. As a result, traditional cache architectures and hardware prefetchers designed for spatially and temporally regular access patterns exhibit very low effectiveness for graph processing. Prior studies report last-level cache hit rates as low as 10-40%, leading to frequent off-chip DRAM accesses and substantial energy cost per operation [58].

This inefficiency is further exacerbated by the mismatch between graph data granularity and memory access granularity. Vertex and edge attributes are often represented using 4-8 byte values, yet modern DRAM systems operate on cache lines of 64 bytes. Consequently, more than 80% of the transferred data may be unused, wasting memory bandwidth and increasing energy consumption without contributing to computation [58]. Such bandwidth underutilization becomes particularly problematic for large-scale and continuously operating systems, where memory traffic dominates total power draw.

These challenges highlight that graph processing performance is frequently bounded not by arithmetic throughput, but by memory access efficiency and data movement costs. Addressing energy and scalability concerns therefore requires algorithmic and architectural approaches that reduce random memory access, improve locality, and minimize unnecessary data transfers rather than relying solely on increased compute capacity.

Overall, analysing large and complex networks requires techniques that can handle combinatorial explosion, irregular connectivity, multidimensional data, and massive scale. These challenges motivate the development of new graph-representation and learning methods that remain accurate, interpretable, and computationally efficient even on very large networks.

2.3 Strengths and Limitations of GNNs

GNNs have become a widely used learning framework for graph-structured data because they can directly model relationships between entities. Unlike traditional machine learning models that assume independent samples, GNNs operate on graphs where nodes, edges, and their interactions play a central role. This makes GNNs well-suited for tasks such as node classification, link prediction, and graph classification in domains including social networks, biological networks, and recommendation systems.

At the core of most GNN architectures is a message passing mechanism. In this process, each node updates its representation by exchanging information with its neighboring nodes. During a single message passing layer, a node collects feature information from its neighbors, aggregates

it using a predefined function (such as summation, mean, or attention), and then combines it with its own features through a learnable transformation. By stacking multiple layers, information can propagate across larger portions of the graph, allowing nodes to capture multi-hop structural and contextual patterns [19].

One of the key strengths of GNNs is their expressive power. By repeatedly aggregating neighborhood information, GNNs can learn representations that reflect both local structure and higher-order dependencies in the graph. This capability has enabled strong performance across many benchmarks, especially in settings where relational information is essential for prediction. In addition, GNNs are flexible and can incorporate node features, edge features, and graph topology within a unified learning framework.

However, these strengths also introduce important limitations. As message passing layers increase, node representations tend to become overly similar, a phenomenon commonly referred to as over-smoothing. When this occurs, nodes from different classes may become indistinguishable, reducing predictive performance. Furthermore, GNNs rely heavily on repeated neighborhood aggregation, which can be computationally expensive and memory intensive for large graphs.

Another challenge is interpretability. While GNNs learn powerful latent representations, their decision-making process is often opaque. Message passing involves complex interactions between input features, graph structure, and hidden neurons, making it difficult to identify which parts of the graph or which internal components are most responsible for a given prediction. Recent studies highlight that explanations based solely on input features or subgraphs may be insufficient, as they fail to capture how information flows through hidden neurons during message passing [19].

Finally, GNNs are sensitive to noise, missing edges, and distribution shifts. Since node representations are directly influenced by their neighbors, errors or noise in the graph structure can propagate through the network and affect downstream predictions. This sensitivity, combined with high computational cost and limited transparency, motivates the exploration of alternative graph learning approaches that balance predictive performance with efficiency, stability, and interpretability.

2.4 Non-Neural and Classical Approaches for Graph Learning

Non-neural and classical approaches to graph learning form an important foundation for analysing graph-structured data. Unlike graph neural networks, these methods rely on explicit graph algorithms, handcrafted structural features, and optimisation-based formulations rather than learned message passing. As a result, they often provide stronger interpretability, predictable computational cost, and better suitability for large-scale or resource-constrained environments.

Early work in graph learning focused on extracting explicit structural descriptors such as node degree, centrality measures, clustering coefficients, and motif or graphlet statistics. These features capture key properties of network topology and can be directly used with conventional machine learning models, including linear classifiers, support vector machines, and decision trees. Because these representations are deterministic and human-interpretable, classical approaches allow clearer reasoning about how graph structure influences model predictions.

Many classical graph learning techniques also formulate inference as graph-theoretic optimisation problems. Tasks such as shortest-path computation, community detection, graph partitioning, and flow optimisation are expressed using well-defined mathematical objectives and solved using combinatorial or heuristic algorithms. While exact solutions are often computationally intractable, approximate methods exploit sparsity and locality to scale to large graphs.

Matrix-based and spectral approaches represent another major class of non-neural graph learning methods. These techniques operate on adjacency matrices, Laplacians, or similarity matrices and embed graph structure using eigendecomposition or matrix factorisation. Such methods preserve important global properties of graphs but incur high memory and computational costs as graph size grows, limiting their practicality for massive or dynamic networks.

Recent work has revisited classical graph algorithms in the context of efficient and non-neural computation , e.g., [51] demonstrates that many graph problems, including traversal and optimisation tasks, can be mapped to non-von Neumann and algorithmic computing paradigms without relying on neural training. Graph processing and learning can be achieved, e.g., using non-neural

architectures that emphasise event-driven computation, sparse communication, and energy efficiency [2].

These studies highlight that effective graph learning does not inherently require deep neural networks. Instead, explicit topology-aware representations combined with lightweight inference mechanisms can achieve competitive performance while significantly reducing computational and energy costs. Such properties are particularly important for edge, embedded, and IoT deployments, where reliability, interpretability, and power efficiency are critical.

Overall, non-neural and classical approaches demonstrate that graph learning can be achieved through principled algorithmic design and structural feature engineering. This perspective motivates modern GNN-free pipelines that retain the strengths of classical methods while improving scalability and deployability, forming the basis for the approaches explored in this thesis.

2.5 Graph Representation Learning: From GNNs to GENNs

Graph-structured data appear in diverse domains including citation networks, biological systems, recommendation engines, power grids, and communication infrastructures. Machine learning on graphs spans both node-level tasks (e.g., node classification, link prediction) and graph-level tasks (e.g., graph classification, regression).

A central challenge in graph learning is how to represent graph topology in a form suitable for machine learning. Broadly, two paradigms have emerged:

1. **GNNs** - which operate directly on graph topology through message passing.

2. **Graph Embedding Neural Networks (GENNs)** - which first compute explicit graph embeddings and then apply conventional neural architectures.

This thesis focuses on the second paradigm and investigates whether carefully designed topology-aware embeddings can provide competitive performance while significantly reducing computational and energy costs.

2.5.1 GNNs

GNNs update node representations through iterative neighborhood aggregation. At each layer, node features are transformed by combining information from neighboring nodes:

$$h_v^{(k+1)} = \sigma\left(W \cdot \text{AGG}\left(\{h_u^{(k)} : u \in \mathcal{N}(v)\}\right)\right) \tag{2.1}$$

Through multiple layers, GNNs capture progressively larger receptive fields and encode local and global topology.

Despite their empirical success, GNNs face several structural limitations:

- High computational cost: Repeated sparse matrix operations increase runtime and memory usage.
- Over-smoothing: Deep aggregation can lead to indistinguishable node representations.
- Irregular memory access patterns: Graph sparsity limits hardware efficiency.
- Architectural isolation: Advances in conventional neural network architectures cannot be directly leveraged without redesigning message passing mechanisms.

These challenges become critical in large-scale graphs and resource-constrained deployment settings, motivating exploration of alternative paradigms.

2.5.2 GENNs

GENNs, as articulated in [88], decouple graph representation from neural processing.

A GENN consists of two explicit stages:

1. An embedding stage that transforms graph topology into fixed-dimensional vector representations.
2. A conventional neural network that performs learning using these embeddings.

Unlike GNNs, GENNs do not rely on iterative message passing. Instead, topology is captured once during embedding, after which standard neural architectures (MLPs, CNN-style blocks, etc.) can be applied.

This separation enables:

- Reduced computational overhead,
- Better hardware utilization,
- Reuse of optimized neural architectures,
- Greater suitability for edge and Green AI scenarios.

2.5.3 Node Embedding and Graph Embedding

Graph embeddings can operate at two levels:

Node Embedding Each node is mapped to a low-dimensional vector that preserves structural or semantic relationships. These embeddings are used for node classification or link prediction tasks.

Graph Embedding The entire graph (or subgraph) is mapped to a single vector representation, enabling graph-level classification or regression.

Both node and graph embeddings can be incorporated within the GENN framework, provided they are computed explicitly before neural processing.

2.5.4 Families of Graph Embedding Methods

Within the embedding-first paradigm, several classes of methods exist:

(1) Matrix Factorization-Based Methods These methods approximate adjacency or Laplacian matrices using low-rank decompositions. Examples include Laplacian Eigenmaps and related spectral approaches. While effective at capturing global structure, they often require expensive eigen-decomposition.

(2) Random Walk-Based Methods Methods such as DeepWalk and node2vec generate node-context co-occurrence statistics through stochastic walks. These methods scale well but depend heavily on sampling and hyperparameters.

(3) Diffusion-Based Methods Diffusion embeddings encode multi-hop influence using processes such as Personalized PageRank or heat kernels. These methods capture global connectivity patterns but can produce dense representations and incur computational cost without approximation.

(4) Structural Role-Based Methods Structural methods embed nodes based on structural equivalence rather than proximity. They capture roles (e.g., hubs, bridges) but may ignore local neighborhood semantics.

(5) Graph Coordinate-Based Methods Graph coordinate systems derive embeddings from distance-based representations using partial distance matrices. Since graph distance matrices are typically low-rank, compact coordinate representations can capture both local and global topology efficiently.

Importantly, diffusion-based and structural embedding methods belong to the GENN paradigm when used as an explicit preprocessing step followed by a conventional neural network. If these embeddings are instead incorporated implicitly within message passing architectures, they fall under the GNN framework.

Thus, the defining distinction is not the embedding mechanism itself, but whether topology extraction is explicit and decoupled from neural processing.

2.6 Topology-Aware and Coordinate-Based Methods

Topology-aware and coordinate-based methods encode graph structure explicitly, avoiding iterative message passing and reducing dependence on deep neural architectures. These approaches

represent nodes using relative positions within the graph, enabling compact and scalable structural representations.

Virtual Coordinate Systems (VCS) embed nodes based on their distances to a small set of anchors, capturing connectivity patterns without requiring explicit geometric information. Extensions such as Topology Coordinates (TC) and Directional Virtual Coordinates (DVC) exploit the low-rank structure of graph distance matrices to produce compact embeddings that preserve both local and global relationships while maintaining low computational complexity [83, 87]. By relying on anchor-based distance sampling, these methods scale efficiently to large graphs and resource-constrained environments.

From a graph learning perspective, coordinate-based representations introduce a strong topological inductive bias, grouping nodes according to structural roles and relative positions rather than feature similarity. This property is particularly relevant for cross-domain generalization, where semantic attributes may vary across datasets but underlying structural patterns remain stable.

Topology-aware pooling methods further incorporate structural reasoning into neural architectures by guiding hierarchical aggregation using topological importance scores. For example, topology-aware pooling (TAP) improves performance by explicitly incorporating global and local structural cues during node selection [21]. However, such approaches remain tightly coupled to message-passing pipelines and end-to-end neural training.

In contrast, coordinate-based methods provide a model-agnostic and lightweight alternative, enabling topology-driven representations to be paired with classical machine learning models. This decoupling allows structural information to be preserved explicitly while avoiding the computational and memory overheads of deep graph neural networks.

2.7 Energy, Carbon Cost, and Green AI

Machine learning on edge platforms and in large-scale IoT environments must operate under constraints that differ markedly from the *performance-at-any-cost* philosophy that drives today's massive AI models. Green AI [93] advocates for research that foregrounds computational effi-

ciency, energy-aware computing, and equitable access to resources. In contrast, Red AI [25] prioritizes accuracy above all else, often disregarding environmental or resource implications. Green AI argues that efficiency should be treated as a primary evaluation metric alongside accuracy, with the broader goal of reducing the carbon footprint of AI systems, democratizing participation in AI research, and enabling innovation without dependence on high-end computing infrastructure.

Recent analyses underscore the environmental impact associated with training large-scale neural models, including GNNs. State-of-the-art architectures have been shown to emit tens of tonnes of CO_2 and consume substantial amounts of electrical energy throughout training. For example, the BLOOM language model was estimated to produce up to 50.5 tonnes of CO_2 when considering its full lifecycle, while GPT-3 is reported to have generated more than 500 metric tonnes of CO_2 during training [82, 63, 62]. Such findings highlight the need for sustainable practices, as well as transparent reporting of energy use, carbon emissions, and computational cost in machine learning research.

Accurately estimating the carbon and energy cost of AI training and inference typically involves tracking GPU/CPU power draw, compute time, cooling overheads, and regional carbon intensity. Tools such as energy-tracking libraries and CO_2 accounting frameworks facilitate reproducible reporting and enable researchers to quantify environmental impact with greater precision [10]. Standardizing such measurements is increasingly encouraged in responsible AI guidelines [20].

The deployment of machine learning and graph inference models on edge devices, including microcontrollers, mobile SoCs, and platforms such as the Raspberry Pi, introduces additional constraints. These systems operate with limited RAM, restricted compute capability, and strict energy budgets, which complicate the deployment of modern deep learning architectures. TinyML frameworks such as TensorFlow Lite and PyTorch Mobile have emerged to address these limitations, offering optimizations that include quantization, pruning, weight clustering, and memory-efficient execution paths [12, 49, 67]. TensorFlow Lite supports a wide range of microcontrollers and

provides lightweight inference kernels, while PyTorch Mobile enables streamlined deployment of PyTorch models on resource-constrained embedded platforms.

Despite these advances, running GNNs on edge hardware remains a particularly challenging task. Unlike CNNs or MLPs, GNNs rely heavily on message passing and neighborhood aggregation, which introduce irregular memory access patterns and large adjacency-based lookup operations. These properties lead to elevated memory bandwidth requirements and energy consumption, making GNNs disproportionately expensive to compute on devices with tight resource envelopes [114]. Further, edge processors often lack energy proportionality: brief bursts of computation can cause disproportionate thermal stress and battery drain, driving the need for architectures optimized not only for FLOPs but also for data movement, duty cycle, and memory locality.

To address these constraints, recent work has introduced hardware-aware neural architecture search techniques, such as HGNAS, which automatically construct GNNs tailored for edge devices. These methods have achieved up to a 10.6× speedup and an 82.5% reduction in peak memory usage without substantial loss in accuracy [115]. Additional strategies—such as model compression, graph partitioning, adaptive sampling, and subgraph-level processing—further help align computation with the limited memory and bandwidth budgets of embedded platforms [46, 112]. Collectively, these techniques contribute to a broader shift toward sustainable and resource-sensitive GNN deployment in IoT and edge ecosystems.

2.8 Machine Learning for Edge and IoT Deployment

The rapid growth of IoT ecosystems has led to billions of connected devices generating continuous streams of sensor data. While cloud computing was initially used for processing this data, the high communication cost, network congestion, and latency associated with continuous offloading make cloud-centric solutions impractical for many real-time IoT applications [69]. Edge computing addresses these constraints by enabling machine learning inference closer to the data source, reducing latency and bandwidth usage while improving privacy and reliability [47].

Deploying machine learning on edge and IoT devices introduces several challenges. IoT devices typically operate with limited memory, low-power processors, and strict energy budgets [18]. These constraints make it difficult to run large deep learning models without modification. As a result, research has focused on designing lightweight architectures such as MobileNet and SqueezeNet [69], and using model compression techniques including quantization, pruning, and knowledge distillation to reduce model size and inference latency [69, 18]. Quantization reduces numerical precision, pruning removes redundant parameters, and distillation trains smaller models to imitate larger ones. These methods significantly lower memory usage and power consumption, making them suitable for microcontroller-class hardware.

A second line of work explores efficient edge hardware platforms. Popular devices such as Raspberry Pi, NVIDIA Jetson Nano, STM32 microcontrollers, Arduino Nano 33 BLE Sense, and Google Coral Board support embedded machine learning through optimized frameworks like TensorFlow Lite, EdgeImpulse, Core ML, and MXNet [47]. These frameworks provide reduced footprint runtimes for on-device inference and support hardware acceleration when available.

Edge intelligence also benefits applications requiring real-time responsiveness, such as health monitoring, human activity recognition, autonomous vehicles, industrial automation, and smart city infrastructure [69, 110]. Real-time IoT environments often require deterministic inference times and high reliability. Studies have shown that on-device inference can reduce reaction time from hundreds of milliseconds (cloud-based) to a few milliseconds, enabling safety-critical decision-making [8].

Beyond computation and latency, edge ML must also address broader system-level constraints. Privacy preservation is essential because transmitting raw data to the cloud can expose sensitive user information such as location, images, or biometric signals [47]. Local inference reduces data exposure and strengthens security. Energy efficiency is also a key requirement. Many IoT devices are battery-powered and must operate for long periods. Power measurement studies show that compressed neural networks can reduce energy consumption by up to 5-10× compared to uncompressed models when deployed on microcontroller-based devices [18].

Reliability poses another challenge. Edge devices often operate in environments with intermittent connectivity, environmental noise, and fluctuating power availability. ML models must therefore be robust to missing data, noisy measurements, and resource variability. Distributed learning architectures, including federated learning and hierarchical edge, cloud processing, have been proposed to improve resilience and reduce dependence on centralized servers [47]. These methods allow devices to collaborate without sharing raw data, improving both reliability and privacy.

Overall, the state of the art shows a clear trend toward lightweight, energy-efficient, and privacy-preserving machine learning models designed specifically for edge and IoT deployment. Despite significant progress, challenges remain in balancing accuracy, power usage, memory footprint, and real-time behaviour. These limitations motivate the need for new approaches such as topology-aware, GNN-free representations and resource-efficient learning pipelines, as explored in this thesis.

2.9 Class Imbalance in Graph Learning

Class imbalance is a common and challenging issue in graph learning, where the number of instances belonging to different classes is highly uneven. In many real-world graphs, only a small fraction of nodes belong to minority classes, while the majority of nodes are concentrated in a few dominant classes. This imbalance is observed in applications such as fraud detection, anomaly detection, citation networks, and biological interaction graphs [59].

One important reason for class imbalance in graphs is the presence of power-law degree distributions. Real-world networks often contain a small number of highly connected hub nodes and a large number of low-degree nodes. When class labels are correlated with structural properties such as node degree or community membership, this uneven connectivity can naturally lead to skewed class distributions. For example, well-connected nodes may dominate certain classes, while sparsely connected or peripheral nodes form small and underrepresented classes [59]. As a result, class imbalance in graphs is not only a data issue but is closely tied to the underlying network structure.

Class imbalance poses unique challenges in graph learning compared to traditional independent and identically distributed (i.i.d.) data. In graph-based models, node representations are influenced by neighboring nodes through message passing or neighborhood aggregation. When majority-class nodes dominate local neighborhoods, minority-class information can be overwhelmed, causing biased representations that favor majority classes. This effect is further amplified in deep graph neural networks, where repeated aggregation may dilute minority-class signals across layers [59].

Several strategies have been proposed to address class imbalance in graph learning. At the data level, resampling techniques such as node oversampling, undersampling, or synthetic node generation aim to rebalance class distributions. However, naive resampling can disrupt graph structure or introduce unrealistic connections. At the algorithmic level, cost-sensitive learning assigns higher penalties to misclassifications of minority-class nodes, encouraging models to pay more attention to underrepresented classes [59]. Other approaches modify loss functions, introduce class-aware regularization, or adjust message passing mechanisms to reduce majority-class dominance.

More recent methods leverage graph structure explicitly to mitigate imbalance. These include neighborhood reweighting, adaptive aggregation schemes, and topology-aware sampling strategies that preserve minority-class connectivity while limiting bias from high-degree majority nodes [59]. Despite these advances, class imbalance remains an open challenge, particularly in large-scale and dynamic graphs where imbalance, noise, and structural heterogeneity coexist.

Overall, class imbalance in graph learning arises from a combination of data scarcity, structural skewness, and power-law network properties. Addressing this issue requires solutions that jointly consider class distributions, graph topology, and learning dynamics, rather than relying solely on techniques developed for non-relational data.

2.10 Gaps in Existing Literature

Despite extensive research on graph representation learning, several critical gaps remain in the existing literature, particularly with respect to efficiency, scalability, and practical deployment constraints. While recent advances in GNNs have demonstrated strong predictive performance

across benchmark datasets, they often rely on deep message-passing architectures that incur high computational, memory, and energy costs. This reliance poses significant challenges for large-scale graphs and resource-constrained environments, such as edge devices and embedded systems, where latency, power consumption, and hardware limitations are critical concerns.

A substantial portion of the literature emphasizes increasingly complex neural architectures to address graph learning tasks, frequently prioritizing accuracy improvements over computational efficiency. However, empirical studies have shown that many GNN-based models suffer from issues such as oversmoothing, limited depth scalability, and sensitivity to graph size and structure. Moreover, recent findings indicate that simpler models, when combined with well-designed structural or proximity-based features, can achieve performance comparable to or exceeding that of deep GNNs, calling into question the necessity of end-to-end message passing for many real-world applications.

Classical graph embedding methods, including factorization-based, random walk-based, proximity diffusion-based, and structural role-based approaches, offer valuable insights into preserving global structure, multi-hop relationships, and role equivalence. Nevertheless, these methods are often treated in isolation within the literature and are rarely explored as unified, topology-aware feature pipelines suitable for modern machine learning models. In particular, there is limited work on systematically combining explicit structural representations with lightweight downstream classifiers in a manner that balances predictive performance with efficiency.

Another notable gap lies in the deployment-oriented evaluation of graph learning methods. While benchmark datasets and leaderboard-driven comparisons are widely used, relatively little attention is paid to metrics such as energy consumption, memory footprint, inference latency, and hardware compatibility. As a result, many proposed methods remain impractical for edge intelligence and sustainable AI scenarios. This limitation is especially pronounced in IoT and cyber-physical systems, where graph-structured data is abundant but computational resources are scarce.

Furthermore, the majority of existing approaches implicitly assume that representation learning must be tightly coupled with neural architectures. This assumption overlooks alternative paradigms

in which rich, topology-aware embeddings are computed explicitly and subsequently consumed by conventional machine learning models. Such decoupled pipelines offer advantages in interpretability, modularity, and hardware efficiency, yet remain underexplored in comparison to neural message-passing frameworks.

Motivated by these gaps, this thesis investigates a GNN-free graph learning framework that emphasizes explicit topology-aware embeddings, efficient feature engineering, and lightweight classifiers. By revisiting foundational concepts from graph theory and embedding literature and integrating them with modern machine learning practices, this work aims to demonstrate that competitive graph learning performance can be achieved without reliance on deep graph neural networks. The proposed approach aligns with the principles of Green AI, prioritizing computational efficiency, scalability, and deployability while maintaining strong predictive capability.

In doing so, this thesis contributes toward a more sustainable and practical direction for graph learning, particularly in edge and resource-constrained environments, and provides a systematic alternative to prevailing GNN-centric methodologies.

Chapter 3

Graph Feature Engineering and Embedding for Artificial Intelligence of Things

3.1 Introduction

Graphs provide a flexible framework for representing complex systems in diverse domains such as neuroimaging, drug discovery, genomics, and online networks [108]. Node classification is a fundamental graph learning task, where the goal is to predict class labels based on node attributes and topology. GNNs have achieved state-of-the-art accuracy by aggregating neighborhood information [61].

Despite their success, GNNs exhibit three major limitations that reduce deployability in resource-constrained environments [88]. First, obtaining a comprehensive understanding of topological information in large or complex graphs often requires multiple rounds of message passing and aggregation, increasing computational cost and latency. Second, repeated aggregation across layers can lead to information dilution, where node features become indistinguishable, a phenomenon known as over-smoothing, which reduces the model's discriminative power. Third, GNNs require substantial computational, memory, and power resources, which limit their practicality for real-time or IoT applications. These challenges highlight the need for scalable and energy-efficient alternatives to support sustainable graph learning.

We propose a *GNN-free* pipeline for node classification that replaces neighborhood aggregation with informative graph descriptors. These features are combined into a novel embedding scheme, termed *Network Feature Embedding* (NFE), capturing local diffusion, positional, and structural information. The high-dimensional NFE vectors are compressed using an unsupervised AutoEncoder, and a Deep MLP classifier is trained on the compact representations. Knowledge distillation further transfers accuracy from a large teacher to a lightweight student model.

On the OGBN-Arxiv benchmark [75], our distilled student achieves ≈70% accuracy while maintaining a model size under 1.5 MB and latency below 1 ms on Raspberry Pi-class hardware. This GNN-free design preserves competitive performance with drastically reduced computational cost, bridging the gap between accurate graph models and deployable, energy-efficient edge AI.

3.2 Need for Node Embedding

We begin by examining the theoretical and practical limitations of the Weisfeiler-Lehman (1-WL) test, which underpin many message-passing GNNs. While 1-WL encodes local structural patterns by iteratively aggregating node labels [41], its expressivity is limited to local neighborhoods, restricting its ability to distinguish nodes with similar local but differing global structures. This limitation is critical for graph learning, where reliance on 1-WL-like mechanisms may overlook semantically meaningful differences. To illustrate this, we conducted a 1-WL label analysis on a subgraph of our dataset.

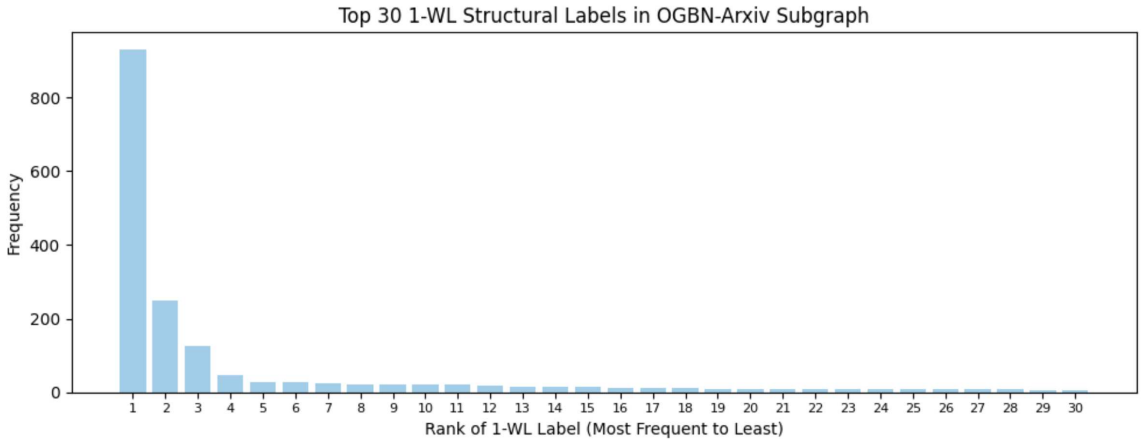

Figure 3.1: Distribution of the top 30 Weisfeiler–Lehman (1-WL) structural labels in the OGBN-Arxiv subgraph. The x-axis shows labels ranked by frequency, and the y-axis indicates the number of nodes assigned to each label.

Figure 3.1 shows a highly skewed label distribution, where a few frequent structures dominate. This imbalance is likely to cause GNNs to under represent rare yet important patterns.

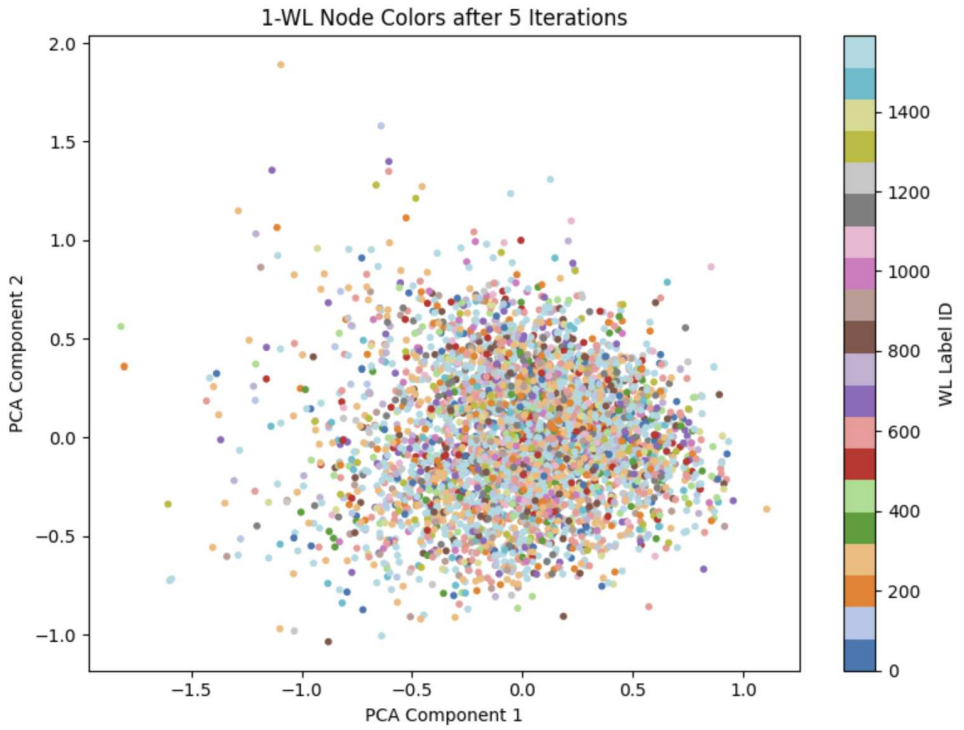

Figure 3.2: PCA visualization of node features colored by their 1-WL label IDs. The dense and overlapping clusters highlight the limited separability achievable through 1-WL–based representations.

We further applied PCA to the 1-WL-based features. As shown in Figure 3.2, overlapping clusters reveal poor separability, confirming that local structural labels alone are insufficient for effective classification in large or complex graphs.

In summary, while 1-WL is effective for simple local patterns, it lacks the capacity to capture nuanced structural roles. These findings motivate the need for richer, multi-perspective node embeddings. In the next section, we introduce NFE, which overcomes these limitations by integrating diffusion, positional, and structural features without message passing.

3.3 Node Feature Embedding (NFE)

GNNs excel at local structure learning but remain constrained by 1-WL expressivity and are often impractical for edge deployment. We propose NFE, a GNN-free hybrid embedding that fuses diffusion, spectral, and structural features for efficient, expressive node representation.

3.3.1 Components of NFE:

Our proposed NFE integrates three complementary signals, Personalized PageRank (PPR) for local diffusion, Laplacian positional encodings for global spectral context, and structural role features for topological importance yielding expressive node representations without message passing for efficient, scalable graph learning.

Personalized PageRank (PPR) Embeddings

To model the contextual influence of neighboring nodes, we compute diffusion-based features using the PPR mechanism [60]. PPR encodes smooth neighborhood-aware information while maintaining locality. Unlike random-walk or k-hop methods, it introduces soft connectivity that reflects the semantic impact of nearby nodes and ensures reproducible, deterministic behavior. It also aligns well with GNN-like behavior when paired with non-graph neural classifiers, making it suitable for GNN-free architectures.

Given an initial node feature matrix $X_0 \in \mathbb{R}^{N\times d}$, PPR embeddings are iteratively updated using:

$$X^{(t+1)} = \alpha X^{(0)} + (1-\alpha)\hat{A}X^{(t)}, \tag{3.1}$$

where $\hat{A}$ is the symmetrically normalized adjacency matrix (with self-loops), and $\alpha = 0.15$ is the teleportation probability. After ten iterations, the process yields $X_{\text{PPR}} \in \mathbb{R}^{N\times 128}$, capturing the local semantic context around each node. The 128-D embedding dimension was selected via grid search [95] for a balanced trade-off between representation quality and computational cost.

Laplacian Positional Encodings

To capture global structure, we use Laplacian Positional Encodings (LPE), derived from the spectral decomposition of the normalized graph Laplacian. Let $A \in \mathbb{R}^{N\times N}$ be the adjacency matrix (including self-loops) and D the corresponding degree matrix. We compute the symmetrically normalized adjacency matrix as $\hat{A} = D^{-1/2}AD^{-1/2}$ and define the normalized graph Laplacian as [77]:

$$L = I - \hat{A}, \tag{3.2}$$

where I is the identity matrix. The eigenvectors of L associated with the smallest non-zero eigenvalues encode smooth positional gradients across the graph and are known to distinguish certain non-isomorphic structures where 1-WL fails. We exclude the trivial eigenvector and retain $d_{\text{pos}} = 32$, concatenating them with other NFE components to provide topology-aware global context.

Although full spectral decomposition can be computationally intensive for very large graphs, we compute only the top-k eigenvectors ($d_{pos} = 32$), which is tractable for mid-scale datasets such as OGBN-Arxiv. For graphs with tens of millions of nodes, scalable spectral approximations such as randomized SVD or Nyström methods can be employed within the same framework to reduce complexity.

Structural Role Features

To enrich the hybrid embedding with structural role awareness, we incorporate three widely used and computationally efficient topological features: normalized degree centrality, local clustering coefficient, and eigenvector centrality [91]. Each captures a distinct aspect of a node's prominence and connectivity pattern within the graph.

Normalized degree centrality measures a node's relative connectedness by dividing its degree by the graph's maximum degree. The local clustering coefficient quantifies how densely connected a node's neighbors are. Eigenvector centrality assigns importance recursively, rewarding connections to other influential nodes and modeling a more global influence pattern.

These features were selected because of their complementary nature, covering local, mesolevel, and global structural perspectives. Moreover, they are efficient to compute, scale well to large graphs, and are interpretable, making them well suited for edge-constrained environments where runtime and memory efficiency are critical.

Alternative structural metrics, such as betweenness centrality, closeness centrality, or motif counts, while informative, are significantly more computationally intensive and do not scale easily to large graphs. Some, like PageRank, are already reflected in the diffusion-based PPR component, while others, such as HITS [109] or k-core numbers [9], offer limited additional value in undirected graphs. Given these trade-offs, our chosen triad offers a strong balance between expressiveness and efficiency.

The resulting three per-node values are concatenated to form the structural role feature matrix $X_{\text{struct}} \in \mathbb{R}^{N\times 3}$, where N is the number of nodes.

Final NFE Representation

The final NFE is constructed by concatenating the three components: diffusion, positional, and structural features.

$$\mathbf{X}_{\text{NFE}} = [\mathbf{X}_{\text{PPR}} \,\|\, \mathbf{U}_{\text{PE}} \,\|\, \mathbf{X}_{\text{struct}}] \in \mathbb{R}^{N\times 163}, \tag{3.3}$$

where $\mathbf{X}_{\text{PPR}}$ captures local diffusion via Personalized PageRank, $\mathbf{U}_{\text{PE}}$ encodes global position using Laplacian eigenvectors, and $\mathbf{X}_{\text{struct}}$ reflects structural roles.

This fused embedding offers rich semantic signals without message passing, enabling efficient shallow classifiers. Even in dense graphs, NFE maintains its discriminative capacity, as the teleportation factor in PPR preserves locality, Laplacian eigenvectors capture spectral variation, and structural features encode relative node prominence without requiring message passing.

As shown in Fig.3, an optional autoencoder compresses the features before MLP-based node prediction, supporting edge-friendly deployment without sacrificing expressivity.

In summary, the NFE framework integrates diffusion-based, positional, and structural cues into a unified and interpretable embedding, without relying on message passing. This design offers a lightweight, scalable alternative to traditional GNNs, particularly well-suited for edge deployment.

The NFE representations, once constructed, are passed through a lightweight autoencoder for dimensionality reduction (Section V), followed by a DeepMLP classifier (Section VI). To improve

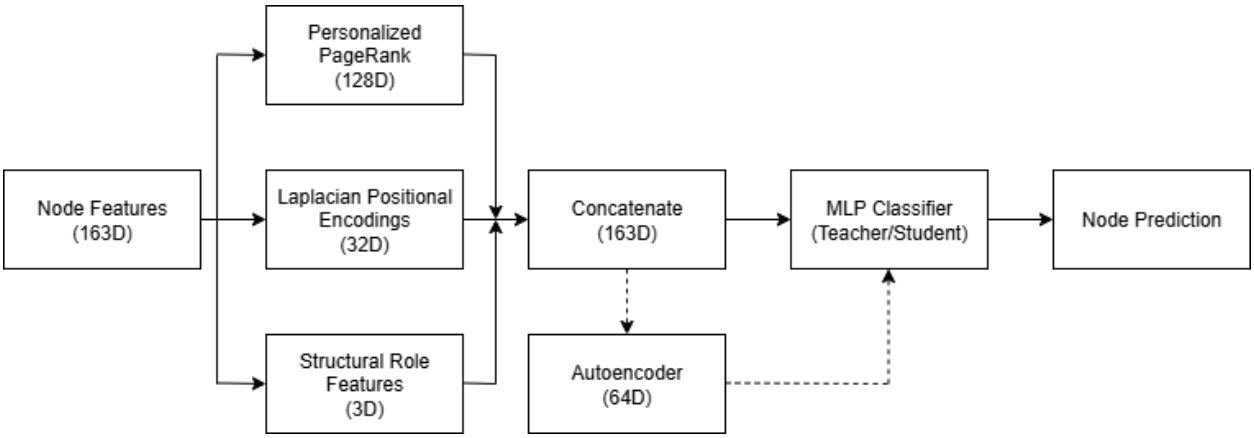

Figure 3.3: Overview of the NFE pipeline, including optional feature compression and MLP-based node prediction.

efficiency without sacrificing accuracy, we then apply knowledge distillation to train a compact student model (Section VII).

3.4 Autoencoder-Based Feature Compression

To reduce the dimensionality of NFE embeddings while preserving semantic structure, we employ a lightweight symmetrical autoencoder [66] with two fully connected layers in each half. The encoder compresses each 163-dimensional vector via FC(163→128) + ReLU + FC(128→64); the decoder mirrors this structure to reconstruct the input.

The model is trained on all 169,343 nodes using MSE loss and the Adam optimizer (learning rate = 0.001) for 20 epochs, with standardized inputs and batch size of 2048. A grid search identified 64 dimensions as the best trade-off between compression and performance.

Training loss declined from 0.571 to 0.064, confirming the effectiveness of the learned representations. The resulting $[169{,}343 \times 64]$ matrix is used in the next stage for classification with DeepMLP.

3.5 Classification with DeepMLP

After generating NFE embeddings, a deep multi-layer perceptron (DeepMLP) was trained for node classification using two variants: one based on the full 163-dimensional NFE representation

and another on the 64-dimensional compressed features obtained from the autoencoder. This setup enables analysis of the trade-off between model complexity and performance while maintaining a GNN-free design. The DeepMLP captures non-linear patterns in these embeddings without message passing. Although the NFE framework is applicable to general graph classification tasks, the OGBN-Arxiv benchmark is used here as a running example; thus, the feature dimensions 163 and 64 correspond to OGBN-Arxiv and may vary for other datasets.

3.5.1 Network Architecture

The DeepMLP comprises five fully connected layers, beginning with an input layer that processes 64-dimensional embeddings, followed by hidden layers of 512, 256, 128, and 64 units. Each hidden layer employs batch normalization, ReLU activation, and dropout for regularization, with dropout rates gradually reduced from 0.4 to 0.2. The output layer contains 40 units, corresponding to the number of node classes in the dataset.

3.5.2 Training and Evaluation

The model was trained with the Adam optimizer (1×10^{-3}) and categorical cross-entropy loss for 100 epochs on standard splits. Training and validation loss decreased from 1.31 to 1.05 and 1.22 to 1.01, respectively, confirming the effectiveness of compressed NFE embeddings. The results show that a simple DeepMLP can deliver competitive accuracy in a GNN-free, scalable setup. In the following section, knowledge distillation is introduced to further reduce model size and inference cost.

3.6 Knowledge Distillation

Knowledge distillation (KD) is applied to enhance deployability by transferring the representational power of a DeepMLP teacher trained on 163-dimensional NFE embeddings to a smaller student model. The student, trained using both hard labels and soft teacher outputs, achieves lower

model complexity and inference cost with minimal accuracy loss. Two variants are evaluated: KD-RAW (163D) and KD-AE (64D compressed embeddings).

3.6.1 Distillation Objective and Training

Following standard KD practices, we used a softmax temperature $T = 4.0$ and combined the losses as:

$$\mathcal{L}_{KD} = \alpha \cdot \mathcal{L}_{CE} + (1 - \alpha) \cdot T^2 \cdot \mathcal{L}_{KL}, \tag{3.4}$$

where $\mathcal{L}_{CE}$ is the cross-entropy loss with true labels, $\mathcal{L}_{KL}$ is the Kullback–Leibler divergence between softened student and teacher predictions, and $\alpha = 0.5$. The student models were trained under standard optimization settings to ensure stable convergence.

3.6.2 Results and Comparison

Table 3.1: Knowledge distillation results: student model performance on the OGBN-Arxiv and OGBN-Proteins datasets.

Model	Validation Accuracy	Test Accuracy
KD-AE (64D Input, Arxiv)	71.37%	69.91%
KD-RAW (163D Input, Arxiv)	**72.72%**	**70.80%**
KD-AE (16D Input, Proteins)	67.57%	64.07%
KD-RAW (43D Input, Proteins)	**68.88%**	**64.16%**

3.6.3 Performance Analysis

The KD-RAW model, trained on uncompressed NFE features, yields higher accuracy than KD-AE. However, the KD-AE model benefits from a 60% reduction in input dimensionality, which substantially lowers the memory footprint and computational load. This trade-off is highly relevant for real-world deployment on resource-limited devices.

In summary, knowledge distillation bridges the gap between accuracy and deployability. While students trained on full-dimensional NFE embeddings yield the best results, compressed variants

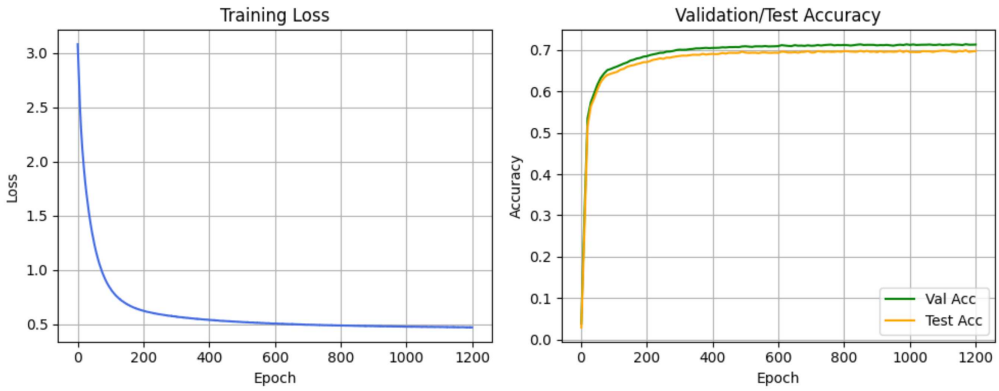

Figure 3.4: Training loss and accuracy curves comparing KD-AE and KD-RAW student models.

offer a strong balance between performance and efficiency. These findings validate KD as a practical strategy for building scalable, edge-friendly models and set the stage for the discussion in the following section.

3.7 Discussion

This section compares our GNN-free, embedding-based pipeline with traditional GNNs, emphasizing its efficiency, interpretability, and deployability. We analyze how reduced parameter counts support resource-constrained environments, evaluate NFE embeddings against node2vec, and discuss accuracy-efficiency trade-offs. Finally, we summarize deployment implications, limitations, and future work.

3.7.1 Comparison with GNN-based Methods

The current OGBN-Arxiv leaderboard is dominated by high-performing GNN models such as BiGTex (5.33M parameters) [5], SimTeG+TAPE+RevGAT (1.39G parameters) [15], and TAPE+RevGAT (280M parameters) [35], which achieve test accuracies of up to 88.51% the highest reported for this benchmark. However, these models rely heavily on message passing mechanisms and require powerful GPUs such as the NVIDIA A100 or RTX 4090.

In contrast, our distilled student models are lightweight, message-passing-free, and suitable for deployment without specialized hardware. Table 3.2 summarizes their parameter efficiency:

despite having 7×–75× fewer parameters than leading GNNs, the KD-based students maintain competitive accuracy. Notably, the AE-based student achieves a 72% reduction in parameter count with only a 0.9% drop in accuracy compared to the raw NFE-based student.

Table 3.2: Performance and parameter comparison of the proposed NFE models on OGBN-Arxiv and OGBN-Proteins relative to representative GNN baselines.

Model	Accuracy	Parameters	MP
Raw NFE (Arxiv)	**0.7099**	**260,968**	No
NFE+AE (Arxiv)	**0.7004**	**210,280**	No
Raw NFE (Proteins)	**0.6416**	**204,720**	No
NFE+AE (Proteins)	**0.6407**	**190,384**	No
BiGTex	0.8851	5,332,968	Yes
SimTeG+TAPE+RevGAT	0.7803	1,386,219,488	Yes
TAPE+RevGAT	0.7750	280,283,296	Yes

3.7.2 Is Fewer Better? A Look at Parameter-Efficient Models

While accuracy remains important, parameter efficiency is critical for deploying models on devices with limited resources. Edge-friendly approaches with fewer parameters achieve low-latency inference, reduced memory use, and align with Green AI principles. As shown in Tables III and IV, our NFE-based models maintain strong accuracy while offering compact, interpretable architectures well suited for IoT and edge environments.

These comparisons collectively reinforce that our NFE-based models offer a compelling balance between accuracy and parameter efficiency, making them especially attractive for edge scenarios where model compactness, low memory usage, and minimal inference overhead are critical deployment considerations.

Table 3.3: Comparison of lightweight models on OGBN-Arxiv: accuracy, parameter count, and suitability for edge deployment (Part A).

Method	Test Accuracy	Parameters	Edge-Friendly
NFE+AE (Ours)	**0.7004 ± 0.0013**	**210,280**	Yes
Raw NFE (Ours)	**0.7099 ± 0.0019**	**260,968**	Yes
Paper 43 [42]	0.7313 ± 0.0017	155,824	Yes
Paper 55 [43]	0.7222 ± 0.0002	15,400	Yes
Paper 56 [101]	0.7219 ± 0.0016	100,648	Yes
Paper 66 [30]	0.7149 ± 0.0027	218,664	Yes
Paper 71 [65]	0.5638 ± 0.0016	120,912	Yes

Table 3.4: Comparison of lightweight models on OGBN-Arxiv: embedding types, model architectures, and key observations (Part B).

Method	Embedding + Model	Notes
NFE+AE (Ours)	**PPR + Laplacian + Role + MLP**	**Most parameter-efficient variant; substantially reduces memory footprint.**
Raw NFE (Ours)	**PPR + Laplacian + Role + MLP**	**Achieves higher accuracy while remaining suitable for edge deployment.**
Paper 43 [42]	GCN_res + C&S_v2	Additional memory and compute overhead from residual layers and smoothing stages.
Paper 55 [43]	Linear + C&S	Very few parameters; relies on transductive label propagation (C&S) and uses all training labels at inference, limiting true edge scalability.
Paper 56 [101]	EGC-S (100k)	Shares filters per node to avoid quadratic complexity, improving scalability.
Paper 66 [30]	GraphSAGE	Scalable message-passing design; LSTM-based variants may introduce additional latency on edge hardware.
Paper 71 [65]	CoLinkDistMLP	Lightweight approach; may underrepresent unseen local topologies during inference.

3.7.3 Green Node Embeddings: Comparing NFE and node2vec

As a representative baseline, we compare our approach with node2vec, a stochastic random walk based embedding method widely used for graph representation learning. According to the OGB leaderboard, node2vec achieves 0.7007 and 0.6881 test accuracy on OGBN-Arxiv and OGBN-Proteins, respectively, using models with 17-22 M parameters executed on an RTX 2080 GPU [26]. In contrast, our distilled NFE models achieve comparable or higher accuracy (0.7099 and 0.6416) with fewer than 0.3 million parameters and without requiring a GPU, highlighting the substantial reductions in runtime, memory, and hardware cost achieved by our lightweight pipeline.

Unlike node2vec, which relies on stochastic random walks and neural training, NFE generates fixed, interpretable node embeddings deterministically in a single pass by combining diffusion-based local features, spectral global encodings, and structural descriptors. Node2vec suffers from issues such as sensitivity to hyperparameters [27], transductiveness [54], inability to differentiate between node and relationship types [11], computational and memory constraints [116], and poor generalization to evolving networks [24]. In contrast, NFE avoids these drawbacks by eliminating sampling and training, resulting in lightweight, reproducible embeddings that require minimal memory and computational resources. This makes NFE suitable for real-time, privacy-preserving, and energy-efficient applications on resource-constrained edge and IoT devices, supporting rapid updates and human-in-the-loop [23] interpretability, thus advancing green AI principles.

3.7.4 Deployment and Efficiency

The proposed pipeline emphasizes IoT and edge deployability through model compression and low-power inference. Following knowledge distillation, structured pruning was applied to the 163-dimensional student MLP, yielding 30% sparsity with minimal accuracy loss (Val. $72.72\% \rightarrow 72.65\%$, Test $70.80\% \rightarrow 70.79\%$). Post-training INT8 quantization was explored but caused notable accuracy degradation (Test $\approx 23\%$) and was therefore omitted from deployment. The final 1.4 MB FP32 model was converted to TensorFlow Lite (v2.15) and deployed on a Raspberry Pi 4 Model B (Cortex-A72, 1.5 GHz, 4 threads). It required only $\approx$ 31MB RAM and

achieved real-time inference with an average latency of 0.78 ms/sample. Power profiling using a TC66C USB meter recorded 2.48W idle and 2.50W active, yielding a dynamic draw of ≈ 0.02W (≈ 0.016mJ/inference, $\approx 1.7\,\mu$gCO_2e/inference). For comparison, a lightweight Node2Vec+MLP baseline was deployed under identical conditions. Table 3.5 summarizes the latency, throughput, memory, power, and energy characteristics, demonstrating the superior energy efficiency and deployability of the proposed compressed NFE model.

Table 3.5: Edge deployment benchmarks on Raspberry Pi 4 Model B. Node2Vec results use full OGBN-Arxiv embeddings. All models are evaluated using TFLite FP32 with 4 threads.

Model	**Latency (ms)**	**Throughput (samples/s)**	**RAM (MB)**	**ΔPower (W)**	**Energy/inf (mJ)**	**CO_2e/inf (μg)**
NFE (KD + Pruned Student, 163D)	0.78	1.28k	31.0	**0.02**	**0.016**	**1.7**
Node2Vec + MLP (128D, full)	**0.432**	**2.32k**	**30.7**	0.26	0.018	1.9

Despite Node2Vec showed lower latency on the Raspberry Pi (0.43 ms) than the NFE student model (0.78 ms), this benefit stems from its *offline* embedding generation, shifting computation away from real-time execution. To compare raw forward costs fairly, both models were evaluated on the same host (CPU-only, 4 threads, batch=4096) with RAPL energy metering. Under these settings, Node2Vec achieved 0.0058 ms latency with 172 k samples/s throughput, whereas NFE reached 0.0014 ms and 70 k samples/s. Thus, while Node2Vec gains from precomputed embeddings, NFE delivers consistently low-latency, real-time inference with a more sustainable accuracy-efficiency balance on resource-limited platforms.

3.8 Conclusion

In this paper, we presented a lightweight, GNN-free framework for node classification that unifies diffusion, spectral, and structural features through the proposed NFE. Combining this representation with autoencoder compression, pruning, and knowledge distillation, we achieved accurate, interpretable, and energy-efficient graph learning without message passing. Experi-

ments on OGBN-Arxiv and OGBN-Proteins showed substantial reductions in parameters, memory, and power consumption while maintaining competitive accuracy. These results demonstrate the promise of NFE for scalable, deployable, and sustainable graph learning aligned with the goals of Green AI.

Chapter 4

Coordinate-Driven Random Forests - A Transferable Approach for Graph Data

4.1 Introduction

A graph $G = (V, E)$ is a mathematical structure consisting of two sets: V and E. The elements of V are called *vertices* (or nodes), and the elements of E are called *edges* (or links). Each edge is associated with one or two vertices, which are its *endpoints* [113]. They have proven to be highly useful in diverse fields, including social networks, biological systems, and computer vision, among others [88].

In computer vision, graph-based methods are particularly valuable for modeling complex and irregular image structures [97]. By representing superpixels [111] or regions as nodes and their shared boundaries as edges, these approaches capture boundary and spatial relationships more naturally than traditional pixel-based methods. Graph models facilitate the integration of long-range dependencies, enabling the propagation of information across distant regions, which is crucial for tasks like semantic segmentation. Moreover, the flexibility of graph representations allows for efficient encoding of structural information, making them a powerful framework to improve understanding of visual scenes and improve performance of vision tasks that benefit from global context and long-range interactions [17].

Recent advances in GNNs have demonstrated the potential of graph representations in various tasks, including the detection of salient objects and the understanding of the scene [64]. However, GNNs and related deep learning models can be computationally expensive, requiring substantial resources for training and inference. This cost becomes a critical limitation when targeting deployment on constrained platforms such as embedded systems or edge devices. Furthermore, large-scale datasets such as Pascal VOC [48] introduce additional difficulties due to severe class

imbalance, where common object categories dominate while others are underrepresented. This imbalance can bias classifiers and degrade recognition performance for minority classes.

While a growing body of work explores graph embeddings and neural networks, a gap remains in approaches that leverage graph coordinate embeddings in conjunction with classical machine learning models. In particular, Random Forests provide a compelling alternative, as they are robust, interpretable, and computationally efficient [52]. However, their potential, when combined with topology-based embeddings for computer vision, remains largely unexplored.

Most GNN approaches are designed for specific graph topologies and learning tasks, such as node classification or link prediction, and rely on learned message-passing mechanisms that are closely tied to the source dataset. Consequently, their effectiveness under transfer learning settings has not been consistently demonstrated. This dependence on topology-specific parameters makes robust cross-dataset transfer a challenging and often underexplored problem in graph learning.

This paper is motivated by the question of whether graph learning truly requires deep neural architectures to achieve robust cross-domain generalization. In particular, we investigate whether structural information alone, when coupled with classical machine learning models, can support effective and transferable learning across diverse graph domains.

In this work, we show that effective and transferable graph learning can be achieved using non-GNN or GE-based architectures, without relying on deep neural message-passing models. By combining topology coordinates, a purely structural graph representation, with a random forest classifier, we demonstrate that non-neural models can achieve competitive performance across vision, molecular, and social graph datasets. The proposed approach exhibits strong robustness to class imbalance and distribution shift and supports zero-shot transfer across domains [40]. Moreover, it achieves these results with substantially lower computational cost, memory footprint, and energy consumption, making it well-suited for edge and resource-constrained deployments [80].

Our key contributions are summarized as follows.

1. We adapt the *Topology Coordinate* (TC) framework to superpixel-based region adjacency graphs, capturing long-range structural information in a compact and vectorised form.

2. We integrate TC embeddings with *Random Forest* (RF) classifiers, presenting a simple and interpretable alternative to neural network-based graph learning methods.

3. We analyze the effects of class imbalance in Pascal VOC-SP node classification and incorporate cost-sensitive learning within the RF framework, emphasising the importance of weighted evaluation metrics.

4. We demonstrate that the proposed TC-DRF pipeline achieves competitive performance with significantly lower computational and memory costs than neural approaches, supporting efficient and *Green AI* learning.

5. We evaluate cross-dataset generalisation by transferring the TC-DRF model from Pascal VOC-SP to COCO-SP without target-domain retraining, maintaining stable accuracy and weighted F1 scores.

6. We further validate the generality of the proposed framework by applying it to molecular (Alchemy) and social (COLLAB) graph datasets, achieving strong performance at a fraction of the computational cost of neural models.

4.2 Topology Coordinate-Driven Random Forests

The Topology Coordinate-Driven Random Forests (TC-DRF) framework consists of three main phases: (i) anchor selection, (ii) embedding construction, and (iii) classification. The first phase involves the selection of representative anchors to encode structural information. In the second phase, we aggregate distances and features to form topology-aware embeddings. Finally, these embeddings are fed into a Random Forest classifier for prediction. An overview of the proposed TC-DRF process is shown in Fig. 2.

4.2.1 Graph Construction from Structured Inputs

The first stage of the proposed TC-DRF pipeline converts structured inputs into graph representations suitable for topology-aware learning. In vision-based settings, we construct region

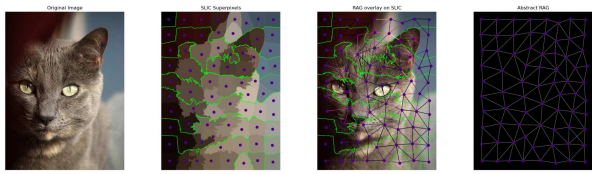

Figure 4.1: Graph construction from image data. An input image (left) is segmented into superpixels using SLIC, followed by region adjacency graph construction based on shared boundaries (middle). The resulting abstract graph (right) represents regions as nodes and spatial relationships as edges, forming the input to the proposed topology-driven learning pipeline.

adjacency graphs (RAGs), where nodes represent spatially coherent regions and edges encode neighborhood relationships between adjacent regions.

A common approach is to first apply superpixel segmentation to partition an image into perceptually homogeneous regions. Each superpixel is treated as a node, and an undirected edge is added between two nodes if their corresponding regions share a boundary. This boundary-based construction preserves local spatial connectivity while avoiding dense pixel-level graphs, yielding sparse and irregular structures that are computationally tractable. In this work, superpixels are produced using SLIC [1], which is widely adopted due to its efficiency and controllable granularity.

As an illustrative instance, we consider the Pascal VOC-SP benchmark [103], where each image is converted into a superpixel RAG with up to a few hundred nodes. Each node is assigned a semantic label derived from pixel-level ground truth, resulting in a multi-class node classification task. This construction produces graphs that encode both local contiguity and longer-range image structure through paths across adjacent regions.

A key challenge in image-derived graphs is class imbalance, where background or frequent categories dominate the node label distribution. This can mask poor performance on rare classes when using accuracy alone. Therefore, we report macro-averaged F1 as the primary metric, as it weights all classes equally and better reflects performance under imbalance.

While image-derived RAGs provide a concrete example, the proposed framework is not restricted to visual data. The same abstraction nodes as entities and edges as relations applies to other domains, including molecular graphs and social networks, which are evaluated in Section IV.

4.2.2 Graph Representation

We represent each structured input as an undirected graph

$$G = (V, E), \quad |V| = N, \ |E| = M, \tag{4.1}$$

where nodes correspond to entities (e.g., regions, atoms, or actors) and edges encode pairwise relationships between them. This abstraction unifies graph construction across domains while enabling topology-driven feature extraction.

Edges are specified by index pairs

$$e_{\text{idx}} \in \mathbb{Z}^{2\times M}, \tag{4.2}$$

and may be associated with scalar or vector-valued attributes

$$e_{\text{attr}} \in \mathbb{R}^{M} \text{ or } \mathbb{R}^{M\times K}, \tag{4.3}$$

representing domain-specific interaction properties such as boundary strength, bond type, or link weight. Each node carries a discrete label

$$y \in \mathbb{Z}^{N}. \tag{4.4}$$

The effective number of nodes N is inferred from the maximum node index present in either e_{idx} or y, ensuring consistency between graph topology and labeling. This formulation accommodates irregular graphs with varying sizes and connectivity patterns.

For graph-level supervision, we employ a majority-voting pooling strategy [73], in which a single label is assigned to each graph based on the most frequent node label it contains. This provides a deterministic aggregation mechanism that avoids introducing additional learned pooling parameters and yields a fixed-dimensional target for graph-level classification.

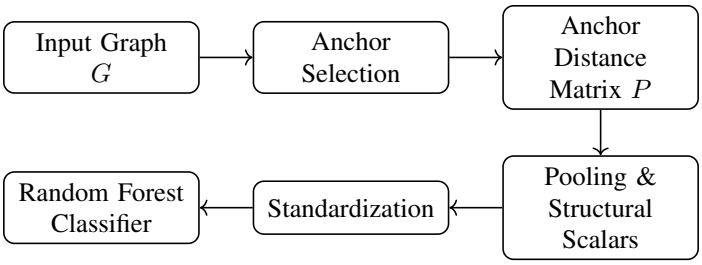

Figure 4.2: Overview of the proposed TC+RF pipeline. The input graph G undergoes anchor selection to obtain the anchor distance matrix P. Distances are pooled with structural scalars to form $\phi(G)$, standardized, and classified with a Random Forest to produce $\hat{y}(G)$.

As shown in Fig. 4.2, the input graph G is first mapped to a topology-aware representation via anchor-based distance encoding (detailed in the next subsection). The resulting distance representation is pooled with simple structural statistics to obtain a fixed-dimensional graph descriptor $\phi(G)$, standardized using training statistics, and finally classified with a Random Forest to produce the predicted label $\hat{y}(G)$.

4.2.3 TC Embedding with Anchors

The Topology Coordinate (TC) embedding is designed to provide a compact, fixed-dimensional, and topology-preserving representation of graphs that scales to large and heterogeneous datasets. Encoding full pairwise shortest-path distances is computationally prohibitive and yields representations whose dimensionality grows with graph size, making them unsuitable for classical machine learning models such as Random Forests. Anchor-based TC embeddings address this limitation by approximating global graph structure through distances to a small, representative subset of nodes.

By encoding distances to well-distributed anchor nodes, TC embeddings capture both local and long-range structural information while maintaining a graph-size–independent dimensionality. This enables permutation-invariant pooling and allows graphs of varying sizes and domains to be embedded into a common vector space. Importantly, TC embeddings avoid iterative message passing and gradient-based optimization, resulting in lower computational cost and energy consumption while preserving structural expressiveness. These properties make anchor-based rep-

resentations particularly suitable for cross-domain transfer and resource-constrained learning settings.

Anchor selection Let $G = (V, E)$ denote a graph with $|V| = N$ nodes. A set of M anchors

$$\mathcal{A} = \{a_1, a_2, \ldots, a_M\} \subseteq V$$

is selected using a greedy farthest-point heuristic [92]. The first anchor is chosen as the node with maximum degree, and each subsequent anchor is selected to maximize its minimum shortest-path distance to previously chosen anchors:

$$a_k = \arg\max_{v \in V} \min_{j<k} d(v, a_j),$$

where $d(\cdot, \cdot)$ denotes the shortest-path distance on G. This strategy promotes a well-dispersed set of anchors that uniformly covers the graph topology, reducing redundancy compared to random anchor selection and yielding more stable topology-aware embeddings.

Distance computation For each node $v \in V$ and anchor $a_j \in \mathcal{A}$, the shortest-path distance is computed using Dijkstra's algorithm [96] on the weighted adjacency matrix A:

$$D(v, a_j) = d(v, a_j).$$

The resulting distances form a matrix

$$P \in \mathbb{R}^{N \times M}, \qquad P_{ij} = D(v_i, a_j),$$

which encodes the distances from all nodes to all anchors. For graphs with disconnected components, infinite distances are replaced with twice the maximum finite distance to ensure numerical stability. Distances are further transformed as $\log(1 + P)$ to reduce dynamic range.

Pooling statistics To obtain permutation-invariant graph-level features [72, 13], distances are aggregated across nodes for each anchor dimension. Specifically, for each column j of P, we compute the mean μ_j, standard deviation σ_j, minimum $m_j^{\min}$, maximum $m_j^{\max}$, and empirical quantiles

$$q_j^{(p)}, \quad p \in \{0.25, 0.5, 0.75\}.$$

The pooled anchor representation is given by

$$\begin{aligned} f_{\text{anchors}} = [\, &\mu_1, \ldots, \mu_M,\, \sigma_1, \ldots, \sigma_M,\, m_1^{\min}, \ldots, m_M^{\min}, \\ &m_1^{\max}, \ldots, m_M^{\max},\, q_1^{(0.25)}, \ldots, q_M^{(0.75)}\,], \end{aligned} \tag{4.5}$$

with $f_{\text{anchors}} \in \mathbb{R}^{7M}$.

Structural scalars Global structural properties are appended as

$$f_{\text{struct}} = [N, |E|, \tfrac{2|E|}{N}],$$

encoding graph size and average connectivity.

Final embedding The final graph representation is obtained by concatenation:

$$\phi(G) = [f_{\text{anchors}} \,\|\, f_{\text{struct}}] \in \mathbb{R}^{7M+3}.$$

This embedding jointly captures fine-grained topological information through anchor-based distance pooling and coarse global graph statistics, yielding a compact yet expressive graph-level descriptor suitable for efficient classification.

Using raw node-to-anchor distance matrices without pooling would produce variable-sized representations that depend on graph order and size, making them incompatible with permutation-invariant graph classification and classical classifiers. Pooling aggregates these distances into sta-

ble global descriptors while retaining structural information, enabling direct comparison across graphs of different sizes.

4.2.4 Feature Scaling

Let $\phi(G) \in \mathbb{R}^{7M+3}$ denote the TC embedding of graph G. Prior to classification, each feature dimension is standardized using statistics computed exclusively from the training split:

$$\tilde{\phi}_j(G) = \frac{\phi_j(G) - \mu_j^{\text{train}}}{\sigma_j^{\text{train}} + \varepsilon}, \qquad j = 1, \ldots, 7M + 3, \tag{4.6}$$

where μ_j^{train} and σ_j^{train} denote the per-dimension mean and standard deviation estimated on the training set, and $\varepsilon > 0$ is a small constant added for numerical stability.

Although tree-based ensemble methods such as Random Forests are generally robust to feature scaling, standardization improves numerical stability when aggregating heterogeneous statistics (e.g., distances, counts, and degrees) and ensures compatibility with alternative classifiers used in ablation or transfer settings. All scaling parameters are fixed after training and applied consistently to validation and test graphs to prevent information leakage.

4.2.5 Random Forest Classifier

The standardized embedding $\tilde{\phi}(G)$ is fed to a Random Forest (RF) classifier [94], chosen for its robustness to nonlinear decision boundaries, resilience to class imbalance, and low computational overhead. The RF constructs an ensemble $\{T_k\}_{k=1}^{K}$ of decision trees trained on bootstrap resamples with randomized feature subsets. Predictions are obtained via majority voting:

$$\hat{y}(G) = \arg\max_{c \in \mathcal{C}} \sum_{k=1}^{K} \mathbf{1}\Big(T_k\Big(\tilde{\phi}(G)\Big) = c\Big), \tag{4.7}$$

where $\mathcal{C}$ denotes the label set and $\mathbf{1}(\cdot)$ is the indicator function.

Unless otherwise stated, we use K=800 trees, a maximum depth of 28, and a minimum of 3 samples per leaf. To mitigate class imbalance, we adopt the `balanced_subsample` strategy,

which reweights classes inversely proportional to their frequencies within each bootstrap sample. All models employ multi-threaded execution (`n_jobs = -1`) and a fixed random seed to ensure reproducibility.

4.2.6 Why Random Forest?

The selection of Random Forests as the downstream classifier in the proposed TC-based pipeline is driven by considerations of computational efficiency, robustness, and architectural compatibility with explicit structural embeddings. Unlike neural classifiers, RFs do not rely on gradient-based optimization, thereby avoiding backpropagation and iterative parameter updates. Training complexity scales approximately as $\mathcal{O}(K \cdot n \cdot d \log n)$, where K is the number of trees, n the number of samples, and d the feature dimension, making RFs well suited for large-scale graph datasets with fixed-dimensional representations.

At inference time, RFs require only shallow tree traversals and simple comparison operations, resulting in low latency and CPU-efficient execution. This characteristic aligns with the goals of Green AI and edge deployment, where predictable runtime behavior and energy efficiency are critical. As demonstrated in Section IV-F, RF-based models achieve competitive predictive performance while significantly reducing inference cost compared to neural baselines.

RFs are also inherently robust to class imbalance through bootstrap sampling and class-weighted training strategies, which is particularly important for datasets such as Pascal VOC-SP that exhibit a dominant background class. In addition, RFs offer a degree of interpretability via feature importance measures, enabling insight into the contribution of topology-coordinate statistics without introducing additional model complexity.

Finally, the use of RFs is consistent with the design philosophy of Topology Coordinates. Since TC embeddings explicitly encode structural information through pooled distance statistics, they do not require deep hierarchical feature learning. Tree-based models are therefore sufficient to learn non-linear decision boundaries over these representations, allowing the proposed TC+RF

framework to balance accuracy, interpretability, and computational efficiency without resorting to parameter-heavy neural architectures.

4.2.7 Extension to Other Datasets

To assess the generality of the proposed TC+RF framework beyond image-derived graphs, we evaluate the same pipeline on additional benchmarks drawn from molecular and social network domains. These datasets differ substantially in graph semantics, scale, and learning objectives, enabling an evaluation of cross-domain transfer without modifying the embedding or classifier design.

Alchemy The Alchemy dataset [102] consists of molecular graphs in which nodes represent atoms and edges denote chemical bonds, with learning tasks defined over continuous-valued molecular properties. Without introducing domain-specific features or architectural changes, we apply the same anchor-based Topology Coordinate embedding, distance pooling, and structural aggregation used for vision graphs. The resulting fixed-dimensional representations are standardized and fed to a Random Forest regressor for property prediction. This setting evaluates whether topology-driven representations can support regression tasks in chemically structured graphs.

COLLAB COLLAB [102] is a social network graph classification benchmark where nodes correspond to researchers and edges represent co-authorship relations. Here, TC embeddings capture global collaboration structure through anchor-based distance statistics, producing permutation-invariant graph-level descriptors. The same Random Forest classifier is used for multi-class prediction, allowing a direct comparison with vision and molecular domains under an identical learning pipeline.

Across all datasets, the TC+RF framework is evaluated using a unified topology-driven design, with no dataset-specific tuning or architectural adaptation. Although models are trained separately per dataset, the same anchor-based representation and classification strategy is retained throughout. The results show that topology-coordinate embeddings consistently encode meaningful structural

information across vision, molecular, and social graphs, while preserving low computational and memory cost.

4.2.8 Evaluation Protocol

We report Accuracy, Macro-F1, and Weighted-F1 for classification tasks, and R^2, MAE, and RMSE for regression tasks, using the designated validation and test splits. Let TP_c, FP_c, FN_c, and n_c denote true positives, false positives, false negatives, and the number of instances for class $c \in \mathcal{C}$, respectively. Per-class precision and recall are defined as

$$\mathrm{Prec}_c = \frac{TP_c}{TP_c + FP_c + \varepsilon}, \qquad \mathrm{Rec}_c = \frac{TP_c}{TP_c + FN_c + \varepsilon}, \tag{4.8}$$

and the per-class F1 score is

$$\mathrm{F1}_c = \frac{2\,\mathrm{Prec}_c\,\mathrm{Rec}_c}{\mathrm{Prec}_c + \mathrm{Rec}_c + \varepsilon}. \tag{4.9}$$

Macro-F1 averages uniformly across classes:

$$\mathrm{F1}_{\text{macro}} = \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \mathrm{F1}_c, \tag{4.10}$$

while Weighted-F1 uses class-support weighting:

$$\mathrm{F1}_{\text{weighted}} = \frac{1}{\sum_c n_c} \sum_{c \in \mathcal{C}} n_c\,\mathrm{F1}_c. \tag{4.11}$$

Accuracy is computed as

$$\mathrm{Accuracy} = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \mathbf{1}(\hat{y}_i = y_i), \tag{4.12}$$

where $\mathbf{1}(\cdot)$ is the indicator function.

For regression benchmarks, we employ coefficient of determination (R^2), mean absolute error (MAE), and root mean square error (RMSE):

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}, \qquad \text{MAE} = \frac{1}{N} \sum_i |y_i - \hat{y}_i|, \tag{4.13}$$

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_i (y_i - \hat{y}_i)^2}. \tag{4.14}$$

Depending on the dataset, task-appropriate metrics are used: classification datasets (Pascal VOC-SP and Collab) are evaluated with Accuracy, Macro-F1, and Weighted-F1 to address label imbalance, whereas the regression-oriented Alchemy dataset is assessed using R^2, MAE, and RMSE [38] [89]. This protocol enables consistent and interpretable comparison of the TC+RF framework across both categorical and continuous graph-learning settings.

This section presented the complete Topology-Aware Random Forest (TC-DRF) framework, detailing graph construction, topology coordinate embedding via anchor-based distance encoding, permutation-invariant pooling, feature standardization, and classification using Random Forests. Each design choice was motivated by the need for scalability, robustness to class imbalance, and computational efficiency, while preserving structural expressiveness across heterogeneous graph domains. The unified evaluation protocol further enables fair comparison across classification and regression tasks. In the next section, we evaluate the proposed TC+RF pipeline, analyzing its predictive performance, cross-domain generalization, and efficiency relative to neural graph learning baselines.

4.3 Results

This section presents a comprehensive evaluation of the proposed TC-DRF framework across multiple graph learning benchmarks. We first examine predictive effectiveness and cross-dataset transfer learning behavior on vision, molecular, and social graph datasets. We then analyze deployment-oriented efficiency metrics, including energy consumption, inference latency, and model footprint, to assess the suitability of the approach for resource-constrained and Green AI deployment scenarios.

Table 4.1: Performance of TC-DRF Pipeline on Pascal VOC-SP

Split	Accuracy	Weighted-F1	Macro-F1
Train	0.8386	0.7657	0.0452
Validation	0.8417	0.7694	0.0435
Test	0.8397	0.7666	0.0435

4.3.1 Effectiveness and Transfer Learning Performance

This section evaluates the predictive effectiveness and transfer learning behavior of the proposed TC-DRF framework across four benchmark datasets: Pascal VOC-SP, COCO-SP, Alchemy, and COLLAB, covering both classification and regression tasks. For node classification benchmarks (Pascal VOC-SP, COCO-SP, and COLLAB), we report Accuracy, Macro-F1, and Weighted-F1 to account for severe class imbalance, while regression performance on Alchemy is assessed using R^2, MAE, and RMSE.

TC-based models use a fixed preprocessing, embedding, and classifier configuration across all datasets, with fixed random seeds to ensure reproducibility. Cross-dataset transfer learning is evaluated only from Pascal VOC-SP to COCO-SP, while Alchemy and COLLAB are trained and tested independently using the same pipeline. GNN baselines are evaluated under a transfer-learning setting, with models trained on Pascal VOC-SP and evaluated on COCO-SP using learned representations.

Performance on Pascal VOC-SP

Table 4.1 summarizes the performance of the TC-DRF pipeline on the Pascal VOC-SP dataset across training, validation, and test splits. The model demonstrates stable generalization, achieving accuracies of 0.8386, 0.8417, and 0.8397, with corresponding Weighted-F1 scores of 0.7657, 0.7694, and 0.7666, respectively. Macro-F1 remains low (approximately 0.04) across all splits, reflecting the severe class imbalance inherent to Pascal VOC-SP rather than model bias. The close alignment between validation and test results indicates that TC-based structural embeddings provide robust and consistent performance under highly skewed label distributions.

Table 4.2: Performance of TC-DRF Pipeline on Alchemy (Regression)

Split	R^2 (mean)	MAE (mean)	RMSE (mean)
Train	0.9218	6.6477	10.7753
Validation	0.8804	8.2048	13.4855
Test	0.8822	8.3378	13.8497

Performance on Alchemy

Table 4.2 summarizes regression results for the Alchemy dataset. The TC-DRF pipeline attains a mean coefficient of determination (R^2) of 0.9218 on the training set, 0.8804 on validation, and 0.8822 on the test set, indicating that more than 88% of the variance in molecular property values is captured by the model. Mean absolute error (MAE) values between 6.65 and 8.34 and corresponding root mean square errors (RMSE) ranging from 10.77 to 13.85 demonstrate low prediction dispersion and numerical stability across splits.

The slight increase in RMSE from training to test ($\approx$28%) suggests mild overfitting but remains within an acceptable range for molecular regression. Overall, these scores indicate strong predictive performance on Alchemy, where an R^2 above 0.85 and MAE below 10 are typically considered reliable. These findings suggest that TC embeddings encode molecular structure effectively and support robust graph-level regression without gradient-based learning.

Performance on COLLAB

Results on the COLLAB dataset are summarized in Table 4.3. The model achieved 97.98% training accuracy and a Weighted-F1 of 0.9798, showing excellent fit. On the validation set, the model reached 78.60% accuracy and Weighted-F1 of 0.7837, and maintained 81.00% accuracy and Weighted-F1 of 0.8101 on the test split. This performance gap between training and validation/test suggests some sensitivity to distribution differences across splits, but the test performance remains strong for social network-style node classification.

Across Pascal VOC-SP, Alchemy, and COLLAB, the TC-DRF pipeline demonstrates consistent effectiveness on both classification and regression tasks without changing the learning model or feature construction procedure.

Table 4.3: Performance of TC-DRF Pipeline on COLLAB

Split	Accuracy	Weighted-F1	Macro-F1
Train	0.9798	0.9798	0.9831
Validation	0.7860	0.7837	0.7523
Test	0.8100	0.8101	0.7870

Table 4.4: Transfer Learning Performance of TC-DRF under Intra- and Cross-Dataset Settings

Setting	Accuracy	Weighted-F1	Macro-F1
Pascal-A $\rightarrow$ Pascal-B	0.8444	0.7732	0.0436
COCO-Train	0.8498	0.7808	0.0119
COCO-Val	0.8534	0.7859	0.0167
COCO-Test	0.8378	0.7639	0.0160

Transfer Learning Evaluation (TC-DRF)

In this work, transferability is a central objective, as our goal is to develop representations that generalize across datasets without retraining. Therefore, we explicitly evaluate transfer learning performance under both intra-dataset distribution shift and cross-dataset zero-shot settings, and compare against a GNN baseline under identical conditions.

To evaluate transferability, we consider both intra-dataset distribution shift and cross-dataset zero-shot transfer. The original Pascal VOC-SP splits are combined and randomly partitioned into Pascal-A (70%) and Pascal-B (30%) using a fixed random seed. A TC-DRF model trained on Pascal-A is directly evaluated on Pascal-B, demonstrating stable performance under distributional variation.

The same model trained on Pascal VOC-SP is further applied to COCO-SP without retraining. Despite the increased dataset scale and diversity, TC-DRF maintains consistent accuracy and Weighted-F1 across splits. The observed low Macro-F1 is attributable to class imbalance rather than degraded transfer performance.

These results suggest that the proposed topology-aware embedding enables strong generalization across datasets. We hypothesize that embedding-based representations, which explicitly encode structural information, are inherently more transferable and can be applied to new datasets with minimal or no retraining.

Overall, TC-DRF demonstrates consistent generalization under both intra-dataset distribution shifts and cross-dataset transfer without target-domain retraining.

Transfer Learning Evaluation (GNN Baseline)

Table 4.5 summarizes the transfer performance of a GNN baseline trained on Pascal-A and evaluated on both Pascal-B and COCO-SP under identical settings. The model achieves an accuracy of 0.7019 and a Weighted-F1 score of 0.5789 on Pascal-B, indicating that the GNN retains a degree of generalization under intra-dataset distribution shift.

When applied to COCO-SP without retraining, the model achieves accuracies around 0.70 across splits, suggesting partial cross-dataset generalization. However, performance remains consistently lower than that of TC-DRF in both intra-dataset and cross-dataset settings.

To further analyze this behavior, we conducted a graphon-inspired [107] structural similarity study using averaged degree-sorted adjacency proxies. The resulting pairwise distances between Pascal-A, Pascal-B, and COCO-SP are nearly identical, indicating that all three datasets share similar coarse structural patterns.

Despite this structural similarity, the GNN achieves only $\sim$0.71 accuracy, whereas TC-DRF reaches $\sim$0.84 under the same transfer setting. This suggests that the performance gap is not due to structural mismatch between datasets, but rather due to limitations of message-passing architectures, which rely primarily on local neighborhood aggregation and do not explicitly encode global topology.

It is important to note that transfer learning is not commonly evaluated in the GNN literature, and we are not claiming that GNNs are inherently transferable. To the best of our knowledge, there is limited prior work demonstrating strong zero-shot transfer of GNNs across datasets without retraining. Some recent studies (e.g., graphon-based approaches [107]) suggest that transferability may be possible under specific conditions, particularly in dense graph regimes. However, real-world graphs, including those used in this work, are typically sparse and exhibit heterogeneous structures, making such assumptions less applicable.

Table 4.5: Transfer Learning Performance of GNN under Intra- and Cross-Dataset Settings

Setting	Accuracy	Weighted-F1
Pascal-A $\rightarrow$ Pascal-B	0.7019	0.5789
COCO-Train	0.6978	0.5956
COCO-Val	0.6964	0.5956
COCO-Test	0.7012	0.5956

In this study, both Pascal VOC-SP and COCO-SP are derived from image superpixels, resulting in graphs with similar construction pipelines and comparable topological characteristics. This controlled setting allows us to meaningfully evaluate transferability. However, in more general scenarios where graph topologies differ significantly, transferability of message-passing GNNs remains an open challenge.

Comparison to Prior Work on Pascal VOC-SP

Table 4.6 compares the proposed TC-DRF pipeline against representative graph neural and transformer-based models reported in prior work on Pascal VOC-SP [16]. Classical message-passing architectures such as GCN, GCNII, GINE, and GatedGCN variants achieve relatively modest test Weighted-F1 scores in the range of 0.12-0.29, while transformer-style models with positional encodings (e.g., SAN+LapPE and SAN+RWSE) improve performance to approximately 0.32 under comparable parameter budgets. In contrast, the proposed TC-DRF model attains a test Weighted-F1 of **0.7666**, exceeding the strongest reported neural baseline by more than **2**$\times$. This margin indicates that explicit structural features combined with tree-based learning can capture relational information that is often learned through deep message passing.

Positioning with Respect to CNN and GNN Approaches

PASCAL VOC is widely used for object detection and segmentation, where convolutional neural networks (CNNs) such as R-CNN achieve approximately 50–60% mean Average Precision (mAP), and segmentation models report mean performance in the range of 40–50% [22]. These approaches operate at the pixel or region level and focus on fine-grained localization.

Table 4.6: Comparison of TC-DRF Pipeline with Prior Graph Models on Pascal VOC-SP

Model	Params	Test F1 (Weighted-F1)
GCN	496k	0.1268
GCNII	492k	0.1698
GINE	505k	0.1265
GatedGCN	502k	0.2873
GatedGCN+LapPE	502k	0.2860
Transformer+LapPE	501k	0.2694
SAN+LapPE	531k	0.3230
SAN+RWSE	468k	0.3216
Ours (TC-DRF)	–	**0.7666**

It is important to note that these reported CNN results are not directly comparable to the results obtained in this work. First, prior studies typically evaluate performance using mAP, whereas our work reports classification accuracy and F1-based metrics, reflecting differences in task formulation and evaluation criteria. Second, the original PASCAL VOC benchmarks exhibit significant class imbalance, which can negatively affect performance metrics. In contrast, our experimental setup applies preprocessing and balancing strategies to mitigate this imbalance, as the primary goal of this study is to evaluate transferability rather than raw detection performance.

Nevertheless, these CNN-based results are included to provide a coarse, approximate reference point for comparison. To the best of our knowledge, there is no prior work reporting classification accuracy metrics on the PASCAL VOC 2011 dataset under comparable graph-based formulations, making direct apples-to-apples comparison challenging. Therefore, the comparison should be interpreted cautiously, primarily as contextual grounding rather than a strict performance benchmark.

In contrast, the proposed TC-DRF framework operates on graph representations derived from superpixel-based region adjacency graphs and performs graph-level classification. Due to differences in task formulation and evaluation metrics, direct numerical comparison with CNN-based detection or segmentation methods is not appropriate.

Graph neural networks (GNNs) applied to similar superpixel-based graph representations attempt to capture relational structure through message passing, but are often limited by local ag-

Table 4.7: Generalization of TC-DRF Framework Across Datasets

Dataset	Metric	Validation	Test
Regression (Alchemy)			
	R^2	0.880	0.882
	MAE	8.20	8.34
	RMSE	13.49	13.85
Classification (COLLAB)			
	Accuracy	0.786	0.810
	Weighted-F1	0.784	0.810
	Macro-F1	0.752	0.787

gregation and higher computational cost. TC-DRF instead encodes topology explicitly through coordinate-based representations, avoiding iterative message passing.

Under this formulation, TC-DRF achieves strong performance on Pascal VOC-SP (e.g., $\sim$84% accuracy and $\sim$0.77 Weighted-F1), while maintaining a lightweight and transferable design. This highlights the effectiveness of explicit topology-aware representations for graph-based visual reasoning.

Generalization Across Non-Vision Graph Domains

To evaluate adaptability across domains, the TC-DRF pipeline is additionally applied to molecular property regression (Alchemy) and social network node classification (COLLAB). For each dataset, the same feature construction strategy and learning pipeline are employed, following the official data splits and a consistent experimental protocol. Table 4.7 reports the corresponding results. On Alchemy, the model achieves an R^2 score of 0.882 with low MAE and RMSE, while on COLLAB it maintains test accuracy and Weighted-F1 above 0.81. These results demonstrate that TC-DRF remains effective across structural, chemical, and relational graph domains without requiring architectural modification.

Summary. Across vision (Pascal VOC-SP, COCO-SP), molecular (Alchemy), and social (COLLAB) benchmarks, the TC-DRF pipeline achieves competitive within-dataset effectiveness and supports zero-shot cross-dataset transfer learning from Pascal VOC-SP to COCO-SP. These find-

ings motivate the deployment-focused analysis of energy, latency, and model footprint in Section 4.3.2.

4.3.2 Energy, Latency, and Model Footprint Analysis

This section evaluates the deployment efficiency of the proposed TC-DRF framework beyond predictive effectiveness, focusing on energy consumption, inference latency, and model footprint. These results complement the effectiveness and transfer learning analysis in Section IV-A by quantifying the suitability of TC-based models for resource-constrained, edge, and Green AI deployment scenarios.

Experimental Setup

Energy and latency measurements were conducted under controlled conditions to ensure fair and reproducible comparisons. CPU experiments were performed on an *AMD EPYC 7763* dual-socket server with 128 logical cores and 527.9 GB RAM, using the Linux `powercap` interface to measure package-level energy consumption. The CPU frequency governor was fixed to *performance* mode to avoid dynamic frequency scaling effects.

GPU measurements were carried out on an *NVIDIA A100-SXM4-40GB* accelerator using NVML-based power sampling. For both CPU and GPU experiments, each model was executed with 30 warm-up runs followed by 300 measured inference runs. Dynamic energy consumption was computed as

$$E_{\text{dyn}} = \max(0, E_{\text{work}} - E_{\text{idle}}),$$

where idle energy was measured over a matched time window. Inference latency was recorded using `time.perf_counter()` and averaged across five repetitions. All measurements were conducted with identical batch configurations.

Energy and Latency Analysis

Energy and latency measurements on the Pascal VOC-SP benchmark are summarized across multiple TC-DRF variants and compared with a transfer-trained GNN baseline. On the CPU,

the Baseline RF consumed 72.06 ± 16.36 mJ per inference with an average latency of 25.52 ± 0.39 ms. Structural pruning (**P**) slightly increased energy consumption to 84.68 mJ per inference with comparable latency, while 8-bit quantization (**Q**) and two-stage (**2S**) configurations increased energy usage to 128.08 mJ and 123.25 mJ per inference, respectively, with latencies around 37-38 ms.

In contrast, the distilled student model (**KD**) achieved 0.182 mJ per inference with an average latency of 0.052 ms, corresponding to a reduction of more than $400\times$ in energy and $700\times$ in latency relative to the Baseline RF, with only a 1.2 percentage-point absolute accuracy loss. This translates to an estimated 0.009 mg CO_2 per inference, highlighting the effectiveness of distillation for ultra-low-power inference.

A transfer-trained GNN baseline was evaluated under the same protocol. When executed on the GPU, the GNN achieved an average latency of 1.05 ms per inference (0.26 ms per graph) with a throughput of approximately 950 inferences per second. Dynamic GPU energy consumption was measured at 36.26 mJ per inference, corresponding to $18.9\,\mu$J per node. When executed on the CPU, the same GNN incurred substantially higher latency (9.67 ms per inference) and lower throughput (103 inferences per second), demonstrating a strong dependence on accelerator hardware for efficient execution.

Model Footprint and Compression

Model disk footprints follow the same efficiency hierarchy observed in energy and latency measurements. The distilled KD student occupies only 32 KB, while the pruned RF remains under 1 MB and the Baseline RF occupies 5.7 MB. Two-stage and quantized RF variants grow substantially larger, reaching tens to hundreds of megabytes. The transfer-trained GNN checkpoint occupies 1.24 MB and contains 105,621 trainable parameters, exceeding the KD student footprint by nearly two orders of magnitude.

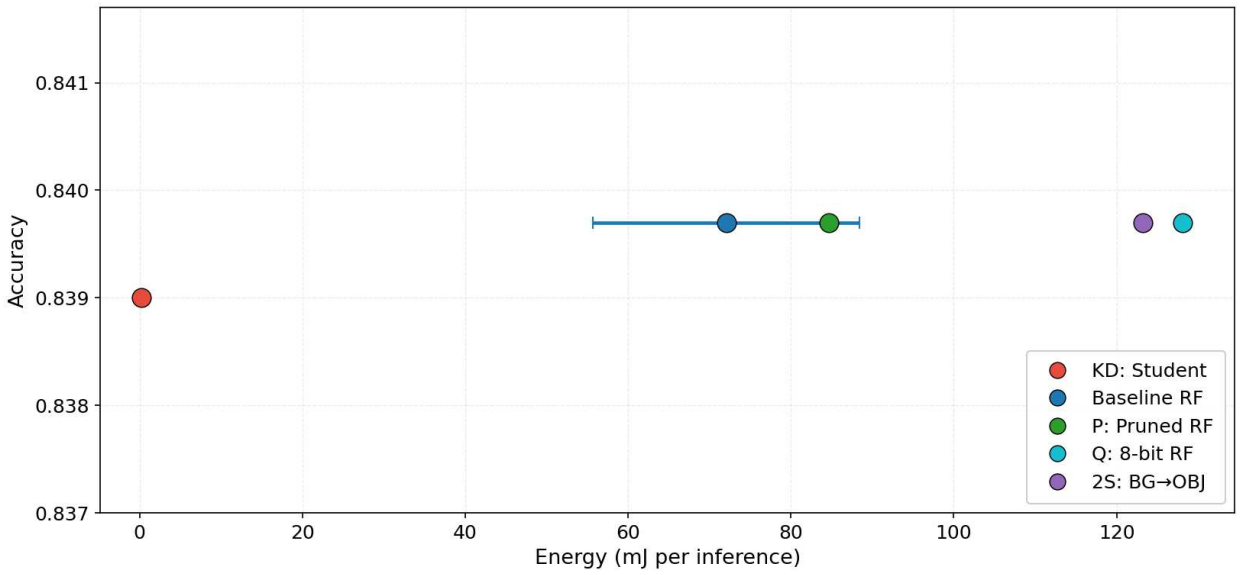

Figure 4.3: Energy-accuracy Pareto frontier across TC-DRF variants on the CPU. The Baseline RF includes one-sigma horizontal error bars. The KD student achieves sub-mJ energy consumption and sub-0.1 ms latency, while transfer-trained GNN models require accelerator support to reach comparable latency.

Table 4.8: Model Size Comparison on Disk

Model	Size (MB)
KD: Student	0.032
P: Pruned RF	0.941
Baseline RF	5.679
2S: BG→OBJ (bg)	33.813
Small RF	41.824
2S: BG→OBJ (obj)	101.067
Q: 8-bit RF	135.334
GNN (transfer)	1.240

Compression Performance

Table 4.8 compares the on-disk model sizes of different TC-DRF variants and baselines. The results highlight substantial reductions in storage footprint achieved through pruning, quantization, and knowledge distillation. In particular, the KD student model is extremely compact, requiring only 0.032 MB, which is over two orders of magnitude smaller than the baseline Random Forest and significantly smaller than compressed RF variants. Pruned and small Random Forest models also achieve notable size reductions compared to the uncompressed baseline, while quantized RF models remain relatively large due to tree structure storage overhead. In contrast, neural baselines

such as transfer-trained GNNs exhibit larger model sizes despite fewer parameters. These results demonstrate that compression techniques, especially knowledge distillation, are highly effective for producing storage-efficient models suitable for deployment on edge and IoT devices.

Efficiency Comparison to Graph Baselines

While transfer-trained GNNs achieve competitive predictive accuracy, their reliance on GPU acceleration to maintain acceptable latency and energy consumption limits their applicability in low-power or CPU-only settings. In contrast, the TC-DRF pipeline achieves strong performance without graph convolution or message passing, while maintaining stable efficiency across both CPU and GPU execution environments. These results highlight the advantages of topology-driven, non-neural representations for Green AI objectives and practical edge deployment.

Overall, the TC-DRF framework demonstrates a favorable balance between predictive performance and deployment efficiency. Its compressed variants, particularly the distilled student model, achieve substantial reductions in energy consumption, latency, and model footprint, while transfer-trained GNNs remain dependent on specialized hardware for efficient execution. These findings reinforce the suitability of topology coordinate representations for sustainable, scalable, and resource-efficient graph learning.

Taken together, the results demonstrate that the TC-DRF framework achieves competitive predictive performance while enabling efficient zero-shot cross-dataset transfer learning. In addition, its favorable energy, latency, and footprint characteristics particularly under compressed variants highlight its practicality for sustainable and edge-oriented graph learning applications. These findings support the use of explicit structural representations combined with lightweight, non-neural models as a viable alternative to deep GNN architectures.

4.4 Discussion and Future Work

The proposed TC-DRF framework demonstrates that explicit structural encoding can rival, and in several cases exceed, the performance of deep message-passing models on large, heterogeneous graph benchmarks. Across Pascal VOC-SP and COCO-SP, the approach achieves strong Weighted-

F1 under extreme class imbalance, while maintaining stable cross-dataset transfer without target-domain retraining. Results on Alchemy and Collab further confirm that topology-coordinate embeddings preserve global relational structure and generalize across vision, molecular, and social graph domains.

Beyond predictive performance, TC-DRF highlights an alternative design philosophy aligned with Green AI principles. By eliminating iterative message passing and large parameter counts, the framework achieves substantial gains in energy efficiency and latency. Empirical profiling shows that the distilled KD variant operates at sub-mJ inference energy and sub-0.1 ms latency on CPU, enabling deployment in real-time and resource-constrained environments. In contrast, transfer-trained GNN baselines exhibit notable degradation under cross-dataset evaluation and rely heavily on accelerator hardware to maintain acceptable inference speed, underscoring the practical advantages of parameter-free, topology-aware representations.

Despite these strengths, the framework inherits limitations from both the datasets and the representation itself. Severe class imbalance in superpixel benchmarks can bias learning toward dominant background categories, reducing Macro-F1 despite stable Weighted-F1. Additionally, distance-based topology coordinates may lose discriminative power in extremely sparse, noisy, or dynamically evolving graphs. Addressing these challenges will require adaptive anchor selection, cost-sensitive training objectives, and dynamic weighting schemes that better reflect local structural variability.

Future work will extend the TC-DRF framework to heterogeneous and attributed graphs with multiple node and edge types, broadening its applicability to more complex real-world settings. We also plan to investigate hybrid pipelines that combine topology-coordinate representations with lightweight neural components, aiming to retain interpretability and efficiency while improving robustness in highly irregular graph topologies. These directions position TC-DRF as a scalable and sustainable alternative for graph learning beyond accuracy-centric paradigms.

4.5 Conclusion

This work introduced the TC-DRF framework as a lightweight, topology-driven alternative to deep GNNs for large-scale graph learning. By explicitly encoding graph structure through anchor-based topology coordinates and decoupling representation learning from gradient-based optimization, the proposed approach achieves strong predictive performance while dramatically reducing computational, memory, and energy costs.

Extensive evaluation across vision, molecular, and social graph benchmarks demonstrates the effectiveness and generality of the framework. On Pascal VOC-SP and COCO-SP, TC-DRF achieves stable Weighted-F1 under extreme class imbalance and exhibits robust cross-dataset transfer without any target-domain retraining. Results on Alchemy and Collab further confirm that topology-coordinate embeddings preserve meaningful structural information across heterogeneous graph domains, supporting both classification and regression tasks with a unified pipeline.

Beyond accuracy-centric evaluation, TC-DRF advances Green AI objectives by prioritizing efficiency and deployability. Energy and latency profiling show that TC-based Random Forest models operate efficiently on CPU-only systems, while the distilled KD variant achieves sub-mJ inference energy, sub-0.1 ms latency, and a negligible memory footprint. In contrast, transfer-trained GNN baselines exhibit notable performance degradation under cross-dataset evaluation and rely heavily on accelerator hardware to maintain acceptable inference speed.

In general, these findings demonstrate that effective and transferable graph learning does not require deep message passing. Explicit, parameter-free topology-aware representations combined with classical learning models offer a practical, interpretable, and sustainable alternative to deep GNNs, particularly for edge, embedded, and resource-constrained deployment scenarios. This work underscores the value of shifting graph learning research beyond the model scale to energy-conscious structural design principles.

Chapter 5

Conclusion

This chapter summarises the main contributions and insights of the thesis, reflects on its limitations, and outlines promising directions for future research. The goal is to position the proposed topology-aware graph learning approaches within the broader landscape of graph representation learning, edge intelligence, and Green AI, while highlighting their practical and conceptual implications.

Contributions

This thesis develops topology-aware and coordinate-based methods for learning representations from graph-structured data, with a focus on efficiency, scalability, and suitability for resource-constrained environments. In response to the limitations of deep GNNs in large-scale, edge, and energy-sensitive settings, the proposed approaches leverage explicit structural information to model graph topology without relying on message passing or deep neural architectures.

The primary contributions of this thesis are as follows. First, a comprehensive analysis of existing graph representation learning paradigms was presented, highlighting the strengths and limitations of factorization-based, random walk-based, neural, diffusion-based, and structural role-based methods. This analysis established the motivation for topology-aware representations that encode structural relationships directly.

Second, this thesis demonstrated coordinate-based and topology-driven embeddings that transform graph structure into fixed-dimensional representations amenable to conventional machine learning models. By decoupling representation learning from iterative message passing, these methods enable efficient training and inference while preserving essential structural information.

Third, the proposed framework demonstrated competitive performance on node-level prediction tasks while exhibiting favorable computational characteristics, including reduced memory footprint, lower energy consumption, and improved inference efficiency. These properties align

with the principles of Green AI and make the approach particularly suitable for edge devices, embedded systems, and large-scale deployments where resource constraints are critical.

Fourth, this work extended the evaluation of graph learning beyond standard in-distribution settings by explicitly examining transferability under both intra-dataset distribution shifts and cross-dataset zero-shot scenarios. The results indicate that topology-aware, embedding-driven representations can generalize across datasets without retraining in controlled settings, highlighting the role of explicit structural encoding in enabling robust generalization.

Finally, this work provided insights into the behavior of topology-driven representations under class imbalance and sparse supervision, contributing to a broader understanding of how structural information can mitigate limitations commonly observed in graph learning tasks.

Insights and Lessons Learned

Several key insights emerged from this research.

First, explicit structural representations remain highly competitive. Despite the dominance of GNNs in recent literature, carefully designed non-neural and shallow-learning approaches can achieve comparable performance on many graph learning tasks, especially when structural information is encoded explicitly.

Second, decoupling graph structure from learning architecture improves efficiency. Representing graphs through coordinate-based or topology-aware embeddings allows the use of conventional machine learning models, avoiding the computational overhead of message passing and enabling better control over memory and energy consumption.

Third, global and multi-hop structural information can be captured without deep propagation. By leveraging distances, proximity measures, or coordinate systems, it is possible to encode long-range dependencies that are difficult for shallow GNNs to capture and expensive for deep GNNs to maintain.

Fourth, transferability is influenced by how structural information is represented. The results suggest that explicitly encoded topology can support generalization across datasets when the un-

derlying graph construction processes produce comparable structural patterns. In contrast, models that rely on localized message passing may struggle to generalize under distributional or structural shifts.

Finally, efficiency is not merely an engineering concern but a modeling choice. The results reinforce the principles of Green AI, showing that model design decisions, rather than hardware scaling alone, play a critical role in enabling sustainable graph learning.

Limitations

While the proposed approach demonstrates several advantages, this work has a number of limitations.

First, reliance on precomputed structural information introduces upfront computational costs, particularly for distance-based or proximity-based features on very large graphs. Although these costs can be amortized during inference, they may limit applicability in highly dynamic settings where graph structure changes frequently.

Second, this thesis focuses on learning from static graph snapshots. The proposed methods are evaluated on fixed-topology datasets, and incremental updates or streaming graph scenarios were not investigated. Supporting dynamic graphs without recomputation remains an open direction for future work.

Third, the approach assumes that structural signals are informative for the downstream task. In domains where node attributes dominate predictive performance, topology-driven representations alone may be insufficient and could benefit from integration with richer attribute-based features.

Fourth, the evaluation of transferability is conducted under controlled conditions where datasets share similar graph construction processes (e.g., superpixel-based image graphs). While this enables meaningful comparison, it limits the generality of conclusions. In scenarios where graph topologies differ significantly across domains, transferability may not hold, particularly for methods that do not explicitly account for structural variation.

Finally, while this thesis evaluates performance across widely used benchmark datasets, additional validation on large-scale industrial or production graphs would further strengthen conclusions regarding robustness and generalisability.

Open Problems and Future Research Directions

This thesis opens several promising directions for future research.

Evolving Graphs

One of the most important open problems is extending topology-aware representations to dynamic and temporal graphs. Real-world networks evolve continuously, and efficient mechanisms for updating structural embeddings without full recomputation remain largely unexplored.

Hybrid Topology-Attribute Models

Combining coordinate-based structural embeddings with lightweight neural or attention-based mechanisms could enable hybrid models that balance expressiveness with efficiency. Exploring principled fusion strategies is an important direction.

Scalable Approximation Techniques

Developing approximate or localized structural measures that preserve global properties while reducing preprocessing costs would significantly improve scalability, particularly for billion-scale graphs.

Fairness and Class Imbalance in Graph Learning

While this work touches on class imbalance, deeper investigation into bias, fairness, and minority representation in topology-aware embeddings is needed, especially in social and biological networks.

Energy-Aware Graph Learning Benchmarks

There is a clear need for standardized benchmarks that evaluate accuracy, latency, memory, and energy consumption jointly. Such benchmarks would enable fair comparison between GNN-based and non-GNN approaches in Green AI contexts.

Edge and Embedded Deployment

Further exploration of deployment in microcontrollers, mobile systems-on-chip, and edge accelerators could solidify the practical impact of topology-aware graph learning methods.

This thesis demonstrates that effective graph learning does not necessarily require deep or computationally intensive models. By revisiting and reinterpreting structural representations through a modern, efficiency-oriented lens, this work contributes to a growing body of research advocating for simpler, more interpretable, and more sustainable graph learning paradigms.

As graph-structured data grows in scale and real-world deployment, especially in edge and IoT settings, the importance of graph learning methods that jointly optimize performance and resource efficiency becomes paramount. From a Sustainable AI perspective, this thesis demonstrates that topology-aware, non–message-passing representations offer a viable and energy-efficient alternative to deep graph models, encouraging further research into resource-conscious graph learning frameworks.

Bibliography

[1] Radhakrishna Achanta et al. “SLIC superpixels compared to state-of-the-art superpixel methods”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.11 (Nov. 2012), pp. 2274–2282.

[2] James B. Aimone et al. *Non-Neural Network Applications for Spiking Neuromorphic Hardware*. Tech. rep. SNL-NM Technical Report. Albuquerque, NM, USA: Sandia National Laboratories, Oct. 2018.

[3] Renzo Angles and Claudio Gutierrez. “Survey of Graph Database Models”. In: *ACM Computing Surveys* 40.1 (Feb. 2008), pp. 1–39.

[4] Enrico Barbierato and Andrea Gatti. “Toward green AI: A methodological survey of the scientific literature”. In: *IEEE Access* 12 (Jan. 2024), pp. 23989–24013.

[5] Amir Beiranvand and Seyed Mohammad Vahidipour. “Integrating structural and semantic signals in text-attributed graphs with BiGTex”. In: *arXiv preprint arXiv:2504.12474* (2025).

[6] Mohamed-Ali Belabbas and Patrick J. Wolfe. “Spectral methods in machine learning and new strategies for very large datasets”. In: *Proceedings of the National Academy of Sciences* 106.2 (2009), pp. 369–374.

[7] Philipp Berger, Georg Hannak, and Gerald Matz. “Efficient graph learning from noisy and incomplete data”. In: *IEEE Transactions on Signal and Information Processing over Networks* 6 (Jan. 2020), pp. 105–119.

[8] Jiang Bian et al. “Machine learning in real-time Internet of Things (IoT) systems: A survey”. In: *IEEE Internet of Things Journal* 9.11 (2022), pp. 8364–8386.

[9] Ariel Bickle. “The k-cores of a graph”. MA thesis. Western Michigan University, 2010.

[10] Laura Bouza, Aurélie Bugeau, and Laurent Lannelongue. “How to estimate carbon footprint when training deep learning models? A guide and review”. In: *Environmental Research Communications* 5.11 (2023), p. 115014.

[11] Tomaž Bratanic. *Complete guide to understanding Node2Vec algorithm – TDS Archive – Medium*. https://medium.com/data-science/complete-guide-to-understanding-node2vec-algorithm-4e9a35e5d147. Aug. 2021.

[12] DFRobot. *Top 8 TinyML frameworks and compatible hardware platforms (TensorFlow Lite, Edge Impulse, PyTorch Mobile, etc.)* https://www.dfrobot.com/blog-13921.html. July 2024.

[13] Edgar Dobriban. “Permutation methods for factor analysis and PCA”. In: *The Annals of Statistics* 48.5 (2020), pp. 2824–2849.

[14] Guimin Dong et al. “Graph neural networks in IoT: A survey”. In: *ACM Transactions on Sensor Networks* 19.2 (2023), pp. 1–50.

[15] Kai Duan et al. “SimTeG: A frustratingly simple approach improves textual graph learning”. In: *arXiv preprint arXiv:2308.02565* (2023).

[16] Vijay Prakash Dwivedi et al. “Long range graph benchmark”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 22326–22340.

[17] Vijay Prakash Dwivedi et al. “Long range graph benchmark”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 35. Dec. 2022, pp. 22326–22340.

[18] Anastasios Fanariotis et al. “Power efficient machine learning models deployment on edge IoT devices”. In: *Sensors* 23.3 (2023), p. 1595.

[19] Junfeng Fang et al. “Cooperative explanations of graph neural networks”. In: *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining (WSDM)*. ACM, Feb. 2023, pp. 616–624.

[20] Federation of American Scientists. *Measuring AI's Energy/Environmental Footprint to Assess Impacts*. https://fas.org/publication/measuring-and-standardizing-ais-energy-footprint/. 2025.

[21] Hongyang Gao, Yujie Liu, and Shuiwang Ji. "Topology-aware graph pooling networks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.12 (Mar. 2021), pp. 4512–4518.

[22] Ross Girshick et al. "Region-based Convolutional Networks for Accurate Object Detection and Segmentation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.1 (Jan. 2016), pp. 142–158.

[23] Google. *What is human in the loop*. https://cloud.google.com/discover/human-in-the-loop. 2025.

[24] Minas A. Goulis. "Optimising node2vec in dynamic graphs through local retraining". MA thesis. University of Twente, 2024.

[25] *Green AI – Communications of the ACM*. https://cacm.acm.org/research/green-ai/. Accessed: Jul. 08, 2025. Dec. 2020.

[26] 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 (KDD)*. ACM, Aug. 2016, pp. 855–864.

[27] 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 (KDD)*. ACM, 2016, pp. 855–864.

[28] Christian Hacker and Bastian Rieck. "On the surprising behaviour of node2vec". In: *arXiv preprint arXiv:2206.08252* (June 2022).

[29] William L. Hamilton, Rex Ying, and Jure Leskovec. "Inductive representation learning on large graphs". In: *Advances in Neural Information Processing Systems*. Vol. 30. 2017.

[30] William L. Hamilton, Rex Ying, and Jure Leskovec. “Inductive representation learning on large graphs”. In: *Advances in Neural Information Processing Systems*. Vol. 30. 2017.

[31] William L. Hamilton, Rex Ying, and Jure Leskovec. “Representation learning on graphs: Methods and applications”. In: *arXiv preprint arXiv:1709.05584* (Sept. 2017).

[32] William L. Hamilton, Rex Ying, and Jure Leskovec. “Representation learning on graphs: Methods and applications”. In: *arXiv preprint arXiv:1709.05584* (2017).

[33] Hao He et al. “Message passing meets graph neural networks: A new paradigm for massive MIMO systems”. In: *IEEE Transactions on Wireless Communications* 23.5 (Oct. 2023), pp. 4709–4723.

[34] Qi He et al. “Large Scale Network Analysis Workshop Chairs’ Welcome Message (LSNA 2013)”. In: *WWW 2013 Companion – Proceedings of the 22nd International Conference on World Wide Web*. 2013, p. 485.

[35] Xiang He et al. “Harnessing explanations: LLM-to-LM interpreter for enhanced text-attributed graph representation learning”. In: *arXiv preprint arXiv:2305.19523* (2023).

[36] Peter Henderson et al. “Towards the systematic reporting of the energy and carbon footprints of machine learning”. In: *Journal of Machine Learning Research* 21.248 (2020), pp. 1–43.

[37] Van Tuan Hoang et al. “Graph representation learning and its applications: A survey”. In: *Sensors* 23.8 (Apr. 2023), p. 4168.

[38] Frank Hoffmann et al. “Benchmarking in classification and regression”. In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 9.5 (Sept. 2019), e1318.

[39] K. Holmberg. “Node Embedding Algorithms in Product Recommendation Systems”. Master of Science Programme in Computing Science and Engineering. Supervisor: Andreas Theodorou; External Supervisor: Martin Rosvall; Examiner: Henrik Björklund. Master’s thesis (30 credits). Umeå, Sweden: Umeå University, 2022.

[40] Lifu Huang et al. “Zero-shot transfer learning for event extraction”. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*. ACL, July 2018, pp. 2160–2170.

[41] Nhat Thanh Huang and Soledad Villar. “A short tutorial on the Weisfeiler–Lehman test and its variants”. In: *Proceedings of the 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 8533–8537.

[42] Qian Huang et al. “Combining label propagation and simple models out-performs graph neural networks”. In: *arXiv preprint arXiv:2010.13993* (2020).

[43] Qian Huang et al. “Combining label propagation and simple models out-performs graph neural networks”. In: *arXiv preprint arXiv:2010.13993* (2020).

[44] Anura P. Jayasumana. *ECE 519: Network-Centric Systems – Lecture Slides*. Department of Electrical and Computer Engineering, Colorado State University. Lecture slides for ECE 519: Network-Centric Systems. 2025.

[45] Zhen Ji and Meng Jiang. “A systematic review of electricity demand for large language models: Evaluations, challenges, and solutions”. In: *Renewable and Sustainable Energy Reviews* 225 (Jan. 2026), p. 116159.

[46] Zhanghao Jiang, Tianyun Chen, and Mu Li. “Efficient deep learning inference on edge devices”. In: *arXiv preprint arXiv:1904.12841* (2019).

[47] Oumayma Jouini et al. “A survey of machine learning in edge computing: Techniques, frameworks, applications, issues, and research directions”. In: *Technologies* 12.6 (2024), p. 81.

[48] Kaggle. *PASCAL VOC 2012 dataset*. https://www.kaggle.com/datasets/gopalbhattrai/pascal-voc-2012-dataset. Accessed: Sep. 24, 2025.

[49] KDnuggets. *On-Device Deep Learning: PyTorch Mobile and TensorFlow Lite*. https://www.kdnuggets.com/2021/11/on-device-deep-learning-pytorch-mobile-tensorflow-lite.html. Nov. 2021.

[50] Anis Khan, Xin Ke, and Yinghui Wu. “Graph data management and graph machine learning: Synergies and opportunities”. In: *ACM SIGMOD Record* 54.2 (July 2025), pp. 28–42.

[51] Mihir Kinderkhedia. “Learning Representations of Graph Data: A Survey”. In: *arXiv preprint arXiv:1906.02989* (June 2019).

[52] Balaji Lakshminarayanan, Daniel M. Roy, and Yee Whye Teh. “Mondrian forests: Efficient online random forests”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 27. 2014.

[53] Matthieu Latapy. “Main-memory triangle computations for very large sparse power-law graphs”. In: *Theoretical Computer Science* 407.1–3 (2008), pp. 458–473.

[54] Niklas Lell and Ansgar Scherp. “iN2V: Bringing transductive node embeddings to inductive graphs”. In: *arXiv preprint arXiv:2506.05039* (2025).

[55] Jure Leskovec. “Dynamics of large networks”. PhD thesis. Carnegie Mellon University, 2008.

[56] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. “Graph evolution: Densification and shrinking diameters”. In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 1.1 (Mar. 2007), 2–es.

[57] D. Lin et al. “T-edge: Temporal weighted multidigraph embedding for Ethereum transaction network analysis”. In: *Frontiers in Physics* 8 (2020), p. 204.

[58] Jianhao Lin et al. “Overcoming the memory hierarchy inefficiencies in graph processing applications”. In: *Proceedings of the 2021 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, Nov. 2021, pp. 1–9.

[59] Zhiyuan Liu et al. “A survey of imbalanced learning on graphs: Problems, techniques, and future directions”. In: *IEEE Transactions on Knowledge and Data Engineering* (Mar. 2025).

[60] Peter Lofgren, Siddhartha Banerjee, and Ashish Goel. “Personalized PageRank estimation and search: A bidirectional approach”. In: *Proceedings of the 9th ACM International Conference on Web Search and Data Mining (WSDM)*. ACM, 2016, pp. 163–172.

[61] Dmytro Lopushanskyy and Bei Shi. “Graph neural networks on graph databases”. In: *arXiv preprint arXiv:2411.11375* (2024).

[62] Alexandra Sasha Luccioni and Alvaro Hernandez-Garcia. “Counting carbon: A survey of factors influencing the emissions of machine learning”. In: *arXiv preprint arXiv:2302.08476* (2023).

[63] Alexandra Sasha Luccioni, Sylvain Viguier, and Anne-Laure Ligozat. “Estimating the carbon footprint of BLOOM, a 176B parameter language model”. In: *Journal of Machine Learning Research* 24.253 (2023), pp. 1–15.

[64] Ao Luo et al. “Cascade graph neural networks for RGB-D salient object detection”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Cham, Switzerland: Springer, Aug. 2020, pp. 346–364.

[65] Yuchen Luo et al. “Distilling self-knowledge from contrastive links to classify graph nodes without passing messages”. In: *arXiv preprint arXiv:2106.08541* (2021).

[66] Gauri Mahindre et al. “Inference in social networks from ultra-sparse distance measurements via pretrained Hadamard autoencoders”. In: *Proceedings of the 45th IEEE Conference on Local Computer Networks (LCN)*. Nov. 2020.

[67] N. Malingan. *TinyML – Getting started with TensorFlow Lite for Microcontrollers*. https://www.scaler.com/topics/tensorflow/tinyml/. Accessed: Jul. 10, 2025. Dec. 2023.

[68] Li Mei and Mark Stamp. “Energy considerations for large pretrained neural networks”. In: *arXiv preprint arXiv:2506.01311* (June 2025).

[69] Massimo Merenda, Carlo Porcaro, and Demetrio Iero. “Edge machine learning for AI-enabled IoT devices: A review”. In: *Sensors* 20.9 (2020), p. 2533.

[70] George B. Mertzios et al. “The complexity of growing a graph”. In: *Proceedings of the International Symposium on Algorithms and Experiments for Wireless Sensor Networks*. Cham, Switzerland: Springer, Sept. 2022, pp. 123–137.

[71] A. Mitra and S. Paul. “Analyzing social networks with dynamic graphs: Unravelling the ever-evolving connections”. In: *Applied Graph Data Science*. Morgan Kaufmann, 2025, pp. 195–214.

[72] Ryan L. Murphy et al. “Janossy pooling: Learning deep permutation-invariant functions for variable-size inputs”. In: *arXiv preprint arXiv:1811.01900* (Nov. 2018).

[73] Nicolas Murray and Florent Perronnin. “Generalized max pooling”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014, pp. 2473–2480.

[74] M. E. J. Newman. “Communities, modules and large-scale structure in networks”. In: *Nature Physics* 8.1 (2012), pp. 25–31.

[75] Open Graph Benchmark. *Leaderboards for node property prediction*. https://ogb.stanford.edu/docs/leader_nodeprop/#ogbn-arxiv. Apr. 2025.

[76] Open Graph Benchmark. *Node Property Prediction*. https://ogb.stanford.edu/docs/nodeprop/. Accessed: Jul. 08, 2025. May 2025.

[77] *ORIE 6334 Spectral Graph Theory Lecture 7*. https://people.orie.cornell.edu/dpw/orie6334/Fall2016/lecture7.pdf. Accessed: Jul. 30, 2025. 2016.

[78] Jeongmin Brian Park et al. “Accelerating sampling and aggregation operations in GNN frameworks with GPU-initiated direct storage accesses”. In: *arXiv preprint arXiv:2306.16384* (June 2023).

[79] Jeongmin Brian Park et al. “LSM-GNN: Large-scale storage-based multi-GPU GNN training by optimizing data transfer scheme”. In: *arXiv preprint arXiv:2407.15264* (2024).

[80] Hansi Yasodara Paththini Hettiarachchige and Anura P. Jayasumana. “Coordinate-Driven Random Forests: A Transferable Approach for Graph Data”. In: *Proceedings of the IEEE International Conference on AI and Data Analytics (ICAD)*. Accepted. 2026.

[81] Hansi Yasodara Paththini Hettiarachchige and Anura P. Jayasumana. “Graph Feature Engineering and Embedding for Artificial Intelligence of Things”. In: *Proceedings of the IEEE Annual Congress on Artificial Intelligence of Things (IEEE AIoT)*. Osaka, Japan, Dec. 2025.

[82] David Patterson and et al. *Carbon emissions and large neural network training*. https://deepai.org/publication/carbon-emissions-and-large-neural-network-training. Accessed: Jul. 09, 2025. Apr. 2021.

[83] Gaurav A. Pendharkar. “Topology inference of smart fabric grids: A virtual coordinate based approach”. MA thesis. Fort Collins, CO, USA: Colorado State University, 2023.

[84] Quentin Petit et al. “Efficient embedding initialization via dominant eigenvector projections”. In: *Proceedings of the SC’25 Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis*. Nov. 2025, pp. 1790–1799.

[85] Nataša Pržulj and Nataša Malod-Dognin. “Network analytics in the age of big data”. In: *Science* 353.6295 (2016), pp. 123–124.

[86] Yongrui Qin et al. “When things matter: A survey on data-centric internet of things”. In: *Journal of Network and Computer Applications* 64 (2016), pp. 137–153.

[87] Z. Qin, A. P. Jayasumana, and R. Paffenroth. “Virtual-Coordinate Based Sampling and Embedding for Machine Learning with Graph Data”. In: *Proceedings of the 2024 International Conference on Machine Learning and Applications (ICMLA)*. IEEE, Dec. 2024, pp. 1192–1197.

[88] Zidi Qin, Randy Paffenroth, and Anura P. Jayasumana. “Graph coordinates and conventional neural networks—an alternative for graph neural networks”. In: *Proceedings of the 2023 IEEE International Conference on Big Data (BigData)*. IEEE, 2023, pp. 4456–4465.

[89] Otto Rainio, Joni Teuho, and Roope Klén. “Evaluation metrics and statistical tests for machine learning”. In: *Scientific Reports* 14.1 (Mar. 2024), p. 6086.

[90] Shubham Rajput, Mohamed Saad, and Tanmay Sharma. “Tu(r)ning AI Green: Exploring energy efficiency cascading with orthogonal optimizations”. In: *arXiv preprint arXiv:2506.18289* (June 2025).

[91] Arjun Saxena and S. S. Iyengar. “Centrality measures in complex networks: A survey”. In: *arXiv preprint arXiv:2011.07190* (2020).

[92] Thomas Schlömer, Daniel Heck, and Oliver Deussen. “Farthest-point optimized point sets with maximized minimum distance”. In: *Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics*. ACM, Aug. 2011, pp. 135–142.

[93] Roy Schwartz et al. “Green AI”. In: *Communications of the ACM* 63.12 (2020), pp. 54–63.

[94] Scikit-learn. *sklearn.ensemble.RandomForestClassifier — scikit-learn Documentation*. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html. Accessed: Aug. 01, 2025. 2025.

[95] Scikit-learn. *sklearn.model_selection.GridSearchCV — scikit-learn 0.22 documentation*. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html. 2019.

[96] SciPy. `shortest_path` — *SciPy v1.15.1 Manual*. https://tinyurl.com/mc57afaw. Accessed: Oct. 10, 2025. 2025.

[97] Abhishek Singh et al. “Image Segmentation: Inducing graph-based learning”. In: *arXiv preprint arXiv:2501.03765* (Jan. 2025).

[98] D. K. Singh and R. Patgiri. “Big graph: tools, techniques, issues, challenges and future directions”. In: *Proceedings of the Sixth International Conference on Advances in Computing and Information Technology (ACITY 2016)*. 2016, pp. 119–128.

[99] Christian Sommer, Elad Verbin, and Wei Yu. “Distance oracles for sparse graphs”. In: *2009 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2009, pp. 703–712.

[100] Stanford University, Department of Computer Science. *Chapter 3: Graphs*. https://web.stanford.edu/class/archive/cs/cs161/cs161.1176/KT_ch3.pdf. Accessed: Aug. 16, 2025. 2017.

[101] Shyam A. Tailor et al. “Do we need anisotropic graph neural networks?” In: *arXiv preprint arXiv:2104.01481* (2021).

[102] TUDataset. *Datasets*. https://chrsmrrs.github.io/datasets/docs/datasets/. Accessed: Nov. 2, 2025. May 2023.

[103] vijaydwivedi75. *Long Range Graph Benchmark (LRGB): NeurIPS 2022 Track on D&B*. https://github.com/vijaydwivedi75/lrgb. Accessed: Oct. 2, 2025. 2022.

[104] Xiao Wang et al. “A survey on heterogeneous graph embedding: methods, techniques, applications and sources”. In: *IEEE Transactions on Big Data* 9.2 (2022), pp. 415–436.

[105] S.F. Windels, N. Malod-Dognin, and N. Pržulj. “Graphlets correct for the topological information missed by random walks”. In: *arXiv preprint arXiv:2405.14194* (2024).

[106] Lei Wu et al. “Learning the implicit semantic representation on graph-structured data”. In: *Proceedings of the International Conference on Database Systems for Advanced Applications (DASFAA)*. Cham, Switzerland: Springer, Apr. 2021, pp. 3–19.

[107] T. Wu et al. “Universal Graph Learning for Power System Reconfigurations: Transfer Across Topology Variations”. In: *arXiv preprint arXiv:2509.08672* (Sept. 2025).

[108] Ming Xu. “Understanding graph embedding methods and their applications”. In: *SIAM Review* 63.4 (2021), pp. 825–853.

[109] Liping Yan et al. “Research on PageRank and hyperlink-induced topic search in web structure mining”. In: *Proceedings of the 2011 International Conference on Internet Technology and Applications*. IEEE, 2011, pp. 1–4.

[110] Mahmut Taha Yazici, Shadi Basurra, and Mohamed Medhat Gaber. “Edge machine learning: Enabling smart internet of things applications”. In: *Big Data and Cognitive Computing* 2.3 (2018), p. 26.

[111] Hongyuan Yu et al. “SuperpixelGraph: Semi-automatic generation of building footprint through semantic-sensitive superpixel and neural graph networks”. In: *International Journal of Applied Earth Observation and Geoinformation* 125 (Dec. 2023), p. 103556.

[112] Linqi Zeng et al. “GNN at the edge: Cost-efficient graph neural network processing over distributed edge servers”. In: *IEEE Journal on Selected Areas in Communications* 41.3 (2022), pp. 720–739.

[113] Ping Zhang and Gary Chartrand. *Introduction to Graph Theory*. New York, NY, USA: Tata McGraw-Hill, 2006.

[114] Ao Zhou et al. “Brief industry paper: Optimizing memory efficiency of graph neural networks on edge computing platforms”. In: *2021 IEEE 27th Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, 2021, pp. 445–448.

[115] Ao Zhou et al. “HGNAS: Hardware-aware graph neural architecture search for edge devices”. In: *IEEE Transactions on Computers* (2024).

[116] Daixin Zhou, Shaohua Niu, and Siheng Chen. “Efficient graph computation for Node2Vec”. In: *arXiv preprint arXiv:1805.00280* (2018).

Appendix A

Tools and Datasets

A.1 Source Code

The complete implementation of the proposed frameworks is publicly available at https://github.com/YashDolorase/genn-embedding. The repository provides a structured collection of Jupyter Notebook files that implement an end-to-end experimental pipeline, including data preprocessing, topology-aware embedding generation, model training, evaluation, and efficiency analysis.

The source code, related materials, and supporting research artifacts are also available via the Colorado State University Mountain Scholar repository. These materials include curated code packages, experimental configurations, and additional documentation related to this work.

A.2 Computational Tools

The experimental evaluation and analysis presented in this thesis were conducted primarily using the Python programming language. Python was used for graph preprocessing, feature engineering, graph coordinate extraction, model implementation, training, and evaluation. Standard scientific computing and machine learning libraries were employed to support these tasks.

A.3 Datasets

This research utilized multiple publicly available graph datasets to evaluate the proposed methods across diverse graph structures, scales, and prediction tasks. Several benchmark datasets were obtained from the Open Graph Benchmark (OGB) [76], which provides standardized datasets and evaluation protocols for graph machine learning research. Additional network datasets were sourced from the TUDataset [102] collection, which hosts a variety of real-world graph datasets commonly used in graph representation learning. These datasets include social, interaction, and

biochemical networks exhibiting heterogeneous structural characteristics. All datasets were used strictly for research purposes in accordance with their respective licenses.

Acknowledgements of Computational Resources

This research made use of the Anvil high-performance computing system at Purdue University, supported by the National Science Foundation through the ACCESS program under allocation `CIS230184`. The author gratefully acknowledges this support. The Open Graph Benchmark (OGB) and TUDataset initiatives are acknowledged for providing open-access benchmark datasets that supported the experimental validation in this work. The manuscript was prepared using the Colorado State University LaTeX thesis template, with document compilation and collaborative editing supported through the CSU Overleaf institutional subscription.