

**UNIVERSITÀ
DEGLI STUDI
DI PADOVA**



**DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE**

DEPARTMENT OF INFORMATION ENGINEERING

MASTER'S DEGREE IN COMPUTER ENGINEERING

Improving Multi-Class Metagenomic Contig Classification with Graph-Based Learning

Supervisor

Prof. Matteo Comin

Candidate

Shabnam Zareshahraki

Student ID: 2091106

2025-2026

2 April 2026

Abstract

Metagenomic assembly and classification are essential for uncovering the taxonomic structure of microbial communities, yet existing approaches often struggle with fragmented short-read assemblies and ambiguous contig assignments. This thesis investigates a graph-based learning framework for metagenomic contig classification, introducing a Graph Neural Network (GNN) that incorporates both sequence-derived features and assembly graph topology. By leveraging contig connectivity, the model learns contextual relationships that traditional sequence-based classifiers cannot capture.

Synthetic benchmark datasets were constructed to mimic realistic microbial communities under two ecological conditions, **GENERIC** (balanced composition) and **FILTERED** (enriched in mobile genetic elements), for both short- and long-read sequencing modalities. The GNN was evaluated against the state-of-the-art 4CAC baseline across these scenarios. Results show consistent improvements in macro-F1, with the most pronounced gains (up to 20%) for minority classes such as viruses and plasmids in short-read data. For long-read assemblies, where contigs exhibit higher contiguity, the GNN achieved performance comparable to 4CAC. Validation on a real-world PacBio HiFi gut metagenome further confirmed the model's robustness to empirical biological noise.

While the framework proved robust and generalizable, training and graph construction were extremely memory-intensive, requiring over 200 GB of RAM per large-scale experiment. Future work should focus on memory-efficient graph sampling, hybrid long-short read integration, and broader validation on diverse environmental samples. Overall, this study demonstrates that topology-aware learning provides a scalable and biologically meaningful extension to existing metagenomic classification methods.

The source code for this project can be found on [GitHub](#)

Keywords: Metagenomics; Contig classification; Graph Neural Networks; Assembly graphs; Long- and short-read sequencing

Sommario

L'assemblaggio e la classificazione metagenomica sono fondamentali per comprendere la struttura tassonomica delle comunità microbiche, tuttavia gli approcci esistenti incontrano spesso difficoltà a causa degli assemblaggi frammentati derivanti da letture corte e delle assegnazioni ambigue dei contig. Questa tesi esplora un framework di apprendimento basato su grafi per la classificazione metagenomica dei contig, introducendo una *Graph Neural Network* (GNN) che integra sia le caratteristiche derivate dalle sequenze sia la topologia del grafo di assemblaggio. Sfruttando la connettività tra i contig, il modello apprende relazioni contestuali che i classificatori basati esclusivamente sulle sequenze non sono in grado di catturare.

Sono stati costruiti dataset di riferimento sintetici per riprodurre comunità microbiche realistiche sotto due condizioni ecologiche—**GENERIC** (composizione bilanciata) e **FILTERED** (arricchita in elementi genetici mobili)—per entrambe le modalità di sequenziamento, a lettura corta e lunga. La GNN è stata valutata rispetto al sistema di riferimento all'avanguardia 4CAC in questi scenari. I risultati mostrano miglioramenti costanti nel punteggio macro-F1, con incrementi più marcati (fino al 20%) per le classi minoritarie come virus e plasmidi nei dati a lettura corta. Per gli assemblaggi a lettura lunga, la GNN ha raggiunto prestazioni comparabili a 4CAC. La validazione su un metagenoma intestinale reale (PacBio HiFi) ha ulteriormente confermato la robustezza del modello su dati empirici.

Sebbene il framework si sia dimostrato robusto e generalizzabile, l'addestramento e la costruzione del grafo si sono rivelati estremamente dispendiosi in termini di memoria, richiedendo oltre 200 GB di RAM per esperimenti su larga scala. I lavori futuri dovrebbero concentrarsi su tecniche di campionamento dei grafi più efficienti in termini di memoria, sull'integrazione ibrida di letture lunghe e corte, e su una più ampia validazione su campioni ambientali. Nel complesso, questo studio dimostra che l'apprendimento consapevole della topologia rappresenta un'estensione scalabile e biologicamente significativa dei metodi esistenti per la classificazione metagenomica.

Il codice sorgente di questo progetto è disponibile su GitHub.

Parole chiave: Metagenomica; Classificazione dei contig; Graph Neural Networks; Grafi di assemblaggio; Sequenziamento a lettura lunga e corta

Acknowledgments

I would like to express my deepest appreciation to my family, without whom I would never have been here, and whose unconditional support and sacrifice have made my life possible, and to my partner, who has been with me throughout these years and has supported me.

I would also like to extend my sincerest gratitude to all my professors at the university, especially my supervisor, Dr. Matteo Comin, without whose patience and guidance, this work would not have been possible.

This could not have been possible without all of you!

Contents

Abstract	ii
Sommario	iii
1 Introduction	1
1.1 Motivation	1
1.2 Proposed Approach	1
1.3 Contributions	2
1.4 Thesis Roadmap	2
2 Background and Problem Definition	3
2.1 Metagenomics and Contig Classification	3
2.2 Key Challenges	3
2.3 Existing Methodologies	4
2.3.1 Reference-based Methods	4
2.3.2 Reference-free (Compositional) Methods	4
2.3.3 Graph-based Methods	4
2.4 Graph Neural Networks (GNNs)	5
2.5 Formal Problem Definition	5
3 Methodology and System Design	6
3.1 System Overview	6
3.2 Graph Neural Networks (GNNs)	8
3.2.1 Theoretical Background	8
3.2.2 Feature Engineering	8
3.3 Dataset Construction	9
3.3.1 Data Sources and Splits	9
3.3.2 Preprocessing and Deduplication	10
3.4 Graph Construction and Representation	10
3.5 GNN Architecture and Training	11
3.5.1 Model Architecture	11
3.5.2 Training Protocol	12
4 Experiments and Results	14
4.1 Experimental Setup	14
4.1.1 Synthetic Read Generation	14
4.1.2 Ground Truth Labeling	15
4.2 Short-Read Results	15

4.2.1	GENERIC Scenario (Balanced)	15
4.2.2	FILTERED Scenario (Mobile Element Enriched)	16
4.3	Long-Read Results	18
4.3.1	GENERIC Scenario	18
4.3.2	FILTERED Scenario	19
4.4	Discussion: Short vs. Long Reads	21
4.5	Real Dataset: Gut HiFi	21
4.5.1	Acquisition and Preprocessing	21
4.5.2	Assembly and Graph Construction	22
4.5.3	Ground-Truth Labeling	22
4.5.4	Results and Analysis	22
4.6	Benchmark on CAMI II Marine Challenge	23
4.6.1	Dataset Description	23
4.6.2	State-of-the-Art (SOTA) Baselines	24
4.6.3	Results	24
5	Conclusion and Future Work	26
5.1	Key Findings	26
5.2	Limitations	26
5.3	Future Work	27
5.4	Final Remarks	27
	Bibliography	28
		30
A	Detailed Experimental Results	30
A.1	Impact of Graph Topology: GNN vs. XGBoost	30
A.2	Supplementary Charts	30
A.3	Confusion Matrices	31

1

Introduction

Metagenomics enables the direct, culture-independent characterization of microbial communities from complex environments such as soil, oceans, and the human gut [13, 2]. A central computational task in this workflow is *contig classification*: assigning assembled genomic fragments to broad biological classes (prokaryote, virus, plasmid, or microeukaryote). Accurate classification is critical for downstream analyses, including pathogen surveillance and the tracking of antimicrobial resistance genes [8].

1.1 MOTIVATION

Despite significant methodological advances, classifying metagenomic contigs remains a formidable challenge. The primary difficulty stems from the fragmented nature of metagenomic assemblies, where sequences are often too short to provide reliable statistical signals (e.g., k -mer frequency profiles) for classification. Furthermore, environmental samples frequently contain novel organisms absent from reference databases, rendering standard homology-based approaches ineffective [9].

These challenges are compounded by biological factors such as *Horizontal Gene Transfer* (HGT), which mixes genetic material between different taxonomic groups, and severe class imbalance, where viral and plasmid sequences are vastly outnumbered by bacterial hosts [7]. Consequently, there is a need for reference-free methods that can resolve ambiguities in short sequences by leveraging additional context beyond simple sequence composition.

1.2 PROPOSED APPROACH

This thesis proposes a graph-based learning framework to address these limitations. While traditional classifiers treat contigs as independent data points, we leverage the *assembly graph*, a structure produced during genome assembly that encodes connections between contigs. By applying Graph Neural Networks (GNNs), specifically the GATv2 architecture, we aim to propagate information from high-confidence regions of the

graph to ambiguous, fragmented neighbors. This approach hypothesizes that the topological context of a contig is as predictive as its sequence content.

1.3 CONTRIBUTIONS

The main contributions of this thesis are:

- **A unified, reproducible benchmark.** We construct temporally partitioned training/validation/test sets from GenBank/RefSeq with strict deduplication and controlled class mapping, providing a fair evaluation substrate across short- and long-read regimes [10, 12, 7].
- **Controlled synthetic communities.** We generate deterministic short- and long-read communities under two scenarios (*GENERIC* and *FILTERED*) that stress distinct ecological class mixes, with exact-count simulation, integrity checks, and downstream assembly.
- **A GATv2-based graph model.** We design and implement a lightweight stand-alone GNN that fuses composition features, simple sequence statistics, and assembly graph topology, trained with imbalance-aware batching and early stopping.
- **Comprehensive evaluation.** We compare the GNN against the 4CAC baseline across four datasets (short/long \times generic/filtered) and real-world data, reporting accuracy, macro-precision, macro-recall, macro-F1, and per-class metrics.
- **Open, auditable implementation.** All pipelines (download, preprocessing, simulation, assembly, graph building, training) are scripted with explicit seeds and logging, enabling end-to-end reproducibility on shared compute (SLURM) [14].

1.4 THESIS ROADMAP

The remainder of the thesis is organized as follows:

- **Chapter 2** (Background and Problem Definition) provides the technical foundations of metagenomics, details the specific challenges of classification (fragmentation, HGT), reviews existing literature, and formally defines the learning problem.
- **Chapter 3** (Methodology and System Design) details the system pipeline, the GNN architecture, feature engineering, and the construction of rigorous training datasets.
- **Chapter 4** (Experiments and Results) describes the experimental setup and presents a comprehensive analysis of results across short-read, long-read, and real-world datasets.
- **Chapter 5** (Conclusion) synthesizes findings, acknowledges limitations, and discusses future directions.



Background and Problem Definition

This chapter contextualizes the thesis within the broader landscape of metagenomics. It first details the specific computational challenges inherent to contig classification, such as assembly fragmentation and biological mosaicism. It then reviews the state-of-the-art in classification methodologies, categorizing them into reference-based, reference-free, and graph-based approaches. Finally, it introduces the theoretical foundations of Graph Neural Networks (GNNs) and provides a formal mathematical definition of the semi-supervised classification problem addressed in this work.

2.1 METAGENOMICS AND CONTIG CLASSIFICATION

Metagenomics differs from traditional genomics by sequencing the total DNA from a community of organisms simultaneously. The computational workflow typically involves *assembly*, where overlapping reads are stitched together into contiguous sequences (contigs), followed by *binning* or *classification* to determine the biological origin of these fragments.

In this thesis, we focus on the classification of contigs into four major classes: **Prokaryotes** (Bacteria and Archaea), **Microeukaryotes**, **Viruses**, and **Plasmids**. This taxonomy aligns with recent benchmarks [7] and addresses the need to distinguish mobile genetic elements (viruses, plasmids) from cellular hosts.

2.2 KEY CHALLENGES

Contig classification is complicated by several intrinsic factors that degrade the performance of standard machine learning models:

Fragmented Assemblies. Metagenomic assemblies are rarely complete. A significant portion of the output consists of short contigs (< 1000 bp). These fragments contain limited sequence information, making it difficult to compute robust statistical features such as oligonucleotide frequencies (*k*-mers) [11]. Short contigs are also more likely to map ambiguously to multiple reference genomes.

Incomplete References. Public repositories like GenBank are heavily biased toward cultivable bacteria and human pathogens [10]. Vast reservoirs of environmental biodiversity, particularly viral and plasmid lineages, remain uncharacterized. Reference-based classifiers often fail to classify these "dark matter" sequences or misclassify them based on weak homology [9].

Horizontal Gene Transfer (HGT) and Mosaicism. Microbes frequently exchange genetic material via HGT, prophage integration, and plasmid exchange. This creates genomic "mosaics" where a single contig may contain subsequences with different evolutionary histories (e.g., a viral gene embedded in a bacterial genome). Sequence-based models may struggle to classify such regions, whereas the genomic context (neighbors in the assembly graph) can provide disambiguating evidence [8].

Class Imbalance. Biological communities are rarely uniform. Prokaryotes typically dominate the biomass, while viruses and plasmids represent a minority of the sequenced DNA. This imbalance causes standard classifiers to overfit the majority class (prokaryotes) to maximize accuracy, often at the expense of sensitivity (recall) for minority classes [7].

2.3 EXISTING METHODOLOGIES

Approaches to metagenomic classification fall into three primary categories.

2.3.1 REFERENCE-BASED METHODS

These methods align contigs to databases of known genomes using tools like BLAST or DIAMOND. A prominent example is CAT/BAT, which assigns taxonomy based on the Lowest Common Ancestor (LCA) of hits [9]. While highly precise for known organisms, these methods are computationally expensive and strictly limited by the completeness of the reference database.

2.3.2 REFERENCE-FREE (COMPOSITIONAL) METHODS

Reference-free methods rely on intrinsic sequence properties, such as k -mer frequency vectors or GC content. Tools like DeepMicroClass [5] use deep learning to extract features directly from sequences. These approaches generalize better to novel organisms but perform poorly on short contigs where the statistical signal is weak.

2.3.3 GRAPH-BASED METHODS

Assembly graphs (e.g., De Bruijn graphs) encode the connectivity between contigs. Tools like 4CAC [7] exploit this topology by first classifying contigs using sequence features (XGBoost) and then smoothing these labels over the graph using rule-based propagation. While effective, 4CAC's propagation rules are static and heuristic. This thesis aims to improve upon this by making the graph propagation *learnable* via GNNs.

2.4 GRAPH NEURAL NETWORKS (GNNs)

Graph Neural Networks generalize deep learning to non-Euclidean graph data. Unlike standard neural networks that treat data points as independent, GNNs leverage the relational structure between nodes.

The core mechanism is *message passing*, where a node v updates its representation h_v by aggregating features from its neighbors $\mathcal{N}(v)$. In this work, we utilize the **Graph Attention Network (GATv2)** architecture [1], which computes a weighted sum of neighbor features:

$$h_v^{(l+1)} = \sigma \left(\sum_{u \in \mathcal{N}(v)} \alpha_{vu} \mathbf{W} h_u^{(l)} \right) \quad (2.1)$$

where α_{vu} is a learnable attention coefficient indicating the importance of neighbor u to node v . This allows the model to dynamically prioritize high-confidence connections (e.g., strong overlaps) while ignoring spurious edges.

2.5 FORMAL PROBLEM DEFINITION

We formalize the contig classification task as a node classification problem on a graph $G = (V, E)$.

- **Nodes (V):** The set of contigs in the assembly. Each node v has a feature vector $x_v \in \mathbb{R}^d$ (containing k -mers, coverage, etc.).
- **Edges (E):** The set of overlaps between contigs. An edge (u, v) exists if contigs u and v overlap in the assembly graph.
- **Labels (Y):** The target class for each node, $y_v \in \{\text{Prokaryote, Virus, Plasmid, Microeukaryote}\}$.

The goal is to learn a mapping function $f(G, X) \rightarrow Y$ that maximizes the macro-F1 score across all classes, ensuring robust performance even for minority classes.

3

Methodology and System Design

This chapter details the computational framework developed for this thesis. We first present an overview of the system pipeline, outlining the flow from raw genomic data to final class predictions. We then expound on the theoretical foundations of Graph Neural Networks (GNNs), justifying their suitability for metagenomic assembly graphs. Finally, we describe the specific feature engineering strategies, dataset construction protocols, and training procedures used to implement the model.

3.1 SYSTEM OVERVIEW

The proposed classification system is designed to integrate sequence composition with assembly graph topology to resolve ambiguities inherent in fragmented metagenomic assemblies. As illustrated in Figure 3.1, the workflow proceeds in four distinct stages:

1. **Data Acquisition and Simulation:** Genomes are retrieved from public repositories and used to generate synthetic short- and long-read communities under controlled ecological scenarios (GENERIC and FILTERED) [10, 7].
2. **Assembly and Graph Construction:** Reads are assembled into contigs using metaSPAdes (for short reads) [11] or metaFlye (for long reads) [3]. The resulting assembly graphs are parsed to extract nodes (contigs) and edges (overlaps).
3. **Feature Extraction:** Each contig is annotated with intrinsic features, including canonical k -mer frequencies, GC content, and coverage depth. Additionally, initial class probabilities are generated to serve as priors.
4. **Graph Learning:** A Graph Neural Network (GNN) processes the populated graph, propagating information across connected contigs to refine class predictions [15].

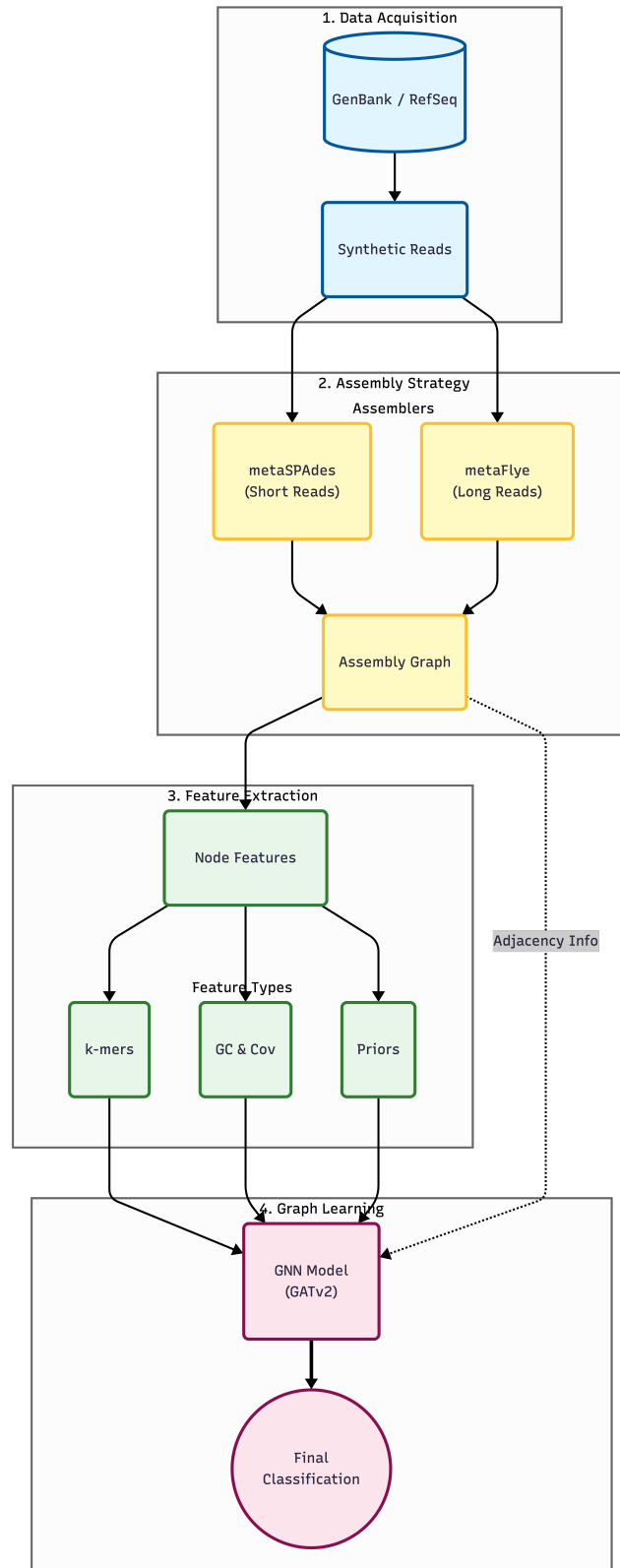


Figure 3.1: **System Pipeline.** The workflow begins with raw read simulation and assembly. The assembly graph is constructed where nodes represent contigs and edges represent overlaps. Node features (k -mers, coverage) are extracted, and the graph is fed into a GNN for final multi-class classification.

3.2 GRAPH NEURAL NETWORKS (GNNs)

3.2.1 THEORETICAL BACKGROUND

Graph Neural Networks (GNNs) are a class of deep learning models designed to operate on data structured as graphs. Unlike traditional machine learning models that treat data points (contigs) as independent and identically distributed (i.i.d.) samples, GNNs explicitly model the relational structure between data points [15].

The core mechanism of a GNN is *message passing*. In each layer of the network, a node aggregates information ("messages") from its local neighborhood to update its own representation. Formally, the hidden state $h_v^{(k)}$ of node v at layer k is updated as:

$$h_v^{(k+1)} = \phi \left(h_v^{(k)}, \bigoplus_{u \in \mathcal{N}(v)} \psi(h_v^{(k)}, h_u^{(k)}, e_{uv}) \right) \quad (3.1)$$

where $\mathcal{N}(v)$ is the set of neighbors of v , \bigoplus is a differentiable aggregation function (e.g., sum or mean), and ϕ and ψ are learnable neural networks [15]. This process allows the model to capture *homophily*—the tendency for connected nodes to share similar properties. In the context of metagenomics, homophily implies that contigs connected by an overlap edge likely belong to the same genome or taxonomic group [4].

3.2.2 FEATURE ENGINEERING

To enable the GNN to distinguish between classes, we construct a rich feature vector X_v for each contig. The selection of these features is motivated by distinct biological signals:

- **Canonical k -mer Composition ($k = 3..7$):** Oligonucleotide frequencies act as a genomic signature. Different taxonomic groups (e.g., viruses vs. bacteria) exhibit distinct k -mer usage patterns due to evolutionary constraints and codon biases [5]. We normalize these frequencies to make them length-invariant.
- **Sequence Statistics:**
 - *GC Content:* The ratio of Guanine and Cytosine bases varies significantly across species and is a classic discriminator in metagenomics [13].
 - *Coverage Depth:* Contigs originating from the same organism typically have similar sequencing coverage. This helps the model group contigs from the same genome even if their sequence composition varies [8].
 - *Length:* Longer contigs provide more reliable statistical signals. We include log-transformed length to allow the model to weight its confidence based on contig size.

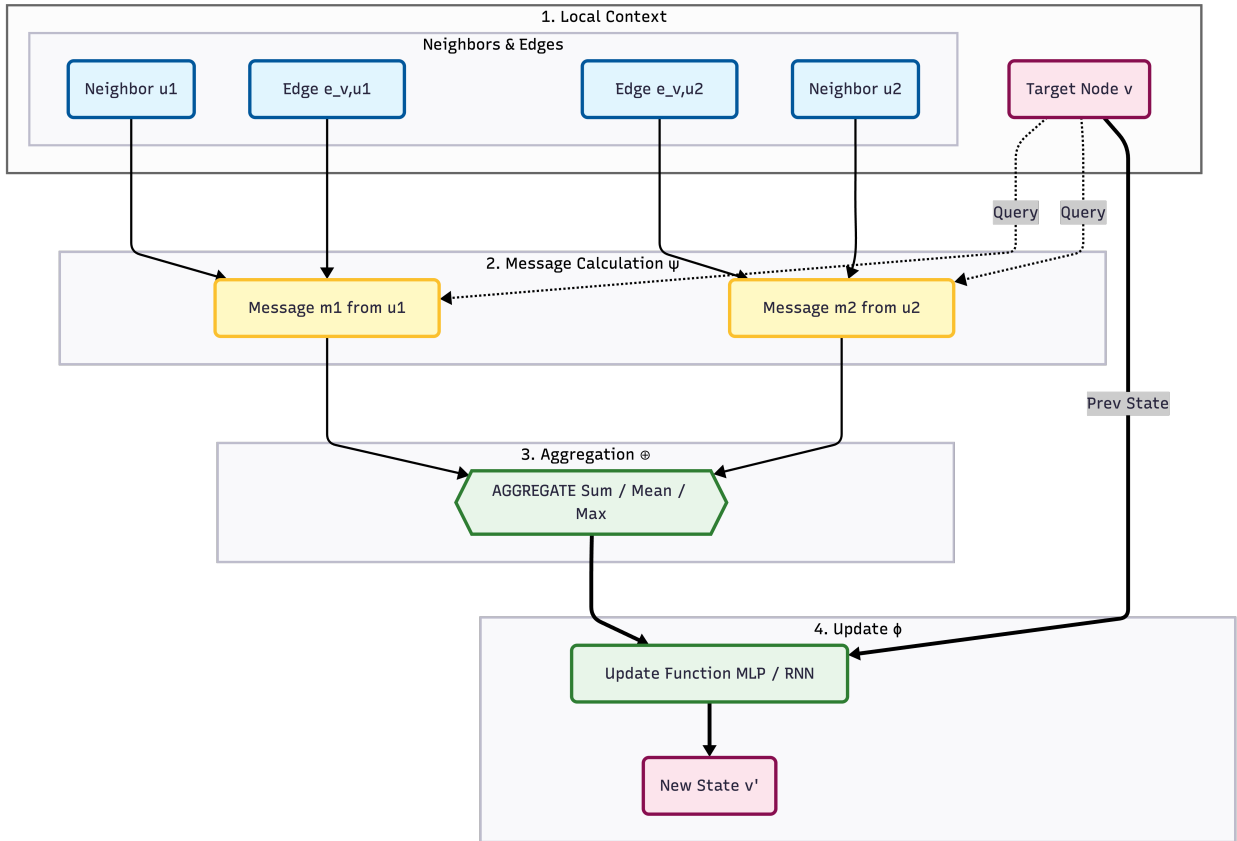


Figure 3.2: **General GNN Message Passing.** Nodes aggregate feature information from their neighbors to update their internal representation. Over multiple layers, a node can integrate information from the broader graph structure.

3.3 DATASET CONSTRUCTION

To train and evaluate the model rigorously, we constructed datasets that reflect realistic metagenomic diversity while ensuring strict separation between training and testing data.

3.3.1 DATA SOURCES AND SPLITS

We downloaded all *Complete* genomic assemblies from NCBI GenBank and RefSeq [10]. Following the temporal partitioning strategy introduced by 4CAC, genomes were divided into three disjoint sets based on release date to prevent temporal bias [7]:

- **Train:** Released before 2021-12-31.
- **Validation:** Released between 2021-12-31 and 2023-12-31.
- **Test:** Released between 2023-12-31 and 2024-12-31.

The genomes were sourced from five primary NCBI categories: **Archaea, Bacteria, Fungi, Protozoa, and Virus.**

Plasmid Extraction Strategy: Unlike the other biological classes, plasmids were not downloaded as independent assemblies. Since plasmids are mobile genetic elements residing within host cells, we extracted them directly from the downloaded **Bacteria** and **Archaea** assemblies. We parsed the FASTA headers of all prokaryotic contigs to identify those explicitly annotated as "plasmid" or "extrachromosomal". These sequences were then separated into a distinct **Plasmid** class. This extraction strategy preserves the biological context, ensuring that the mobile elements in our dataset directly correspond to the prokaryotic hosts present in the GenBank repositories.

3.3.2 PREPROCESSING AND DEDUPLICATION

Raw genomes underwent a rigorous cleaning pipeline to ensure quality and independence:

- **Quality Filtering:** Contigs shorter than 1,000 bp were removed to ensure high-quality training data, and sequences with $> 5\%$ ambiguous bases (N) were discarded.
- **Plasmid Separation:** As described above, plasmid contigs were stripped from their host genomes and reassigned to the Plasmid class. The remaining non-plasmid contigs in those assemblies retained their original Prokaryote (Bacteria/Archaea) label.
- **Genome Exclusion:** Genomes retaining fewer than one contigs after filtering were excluded from the dataset.

Crucially, to prevent information leakage, we performed **cross-split de-duplication**. We compared genomes across splits using both exact MD5 hashing and Mash distance estimation [12]. Any genome in the Validation or Test set that was near-identical (Mash distance ≤ 0.05 , approx. 95% ANI) to a genome in the Training set was removed. This ensures that the model is evaluated on truly novel data, not just memorized sequences.

The final distribution of genomes (and extracted plasmid sets) per class and split is detailed in Table 3.1.

Class	Training	Validation	Test	Total
Prokaryote	43,088	1,622	113	44,823
Microeukaryote	3,869	147	20	4,036
Virus	69,902	820	500	71,222
Plasmid (Extracted)	88,282	909	461	89,652
Total	205,141	3,498	1,094	209,733

Table 3.1: Number of genomes (or extracted plasmid sets) per modeling class in the final dataset.

3.4 GRAPH CONSTRUCTION AND REPRESENTATION

Graphs were constructed for each synthetic community using the outputs of metaSPAdes (short-read) and metaFlye (long-read) assemblies. The graph representation is defined as follows:

- **Nodes:** Contigs of length ≥ 500 bp.
- **Edges:** Undirected edges representing sequence overlaps. Edge attributes include overlap length and relative orientation.
- **Edge Weights:** Calculated as $\log(1+\text{overlap_length})$ to prioritize stronger physical connections in the message-passing phase.

This structure allows the GNN to propagate labels from confident, long contigs to shorter, ambiguous neighbors via high-confidence overlaps [6].

3.5 GNN ARCHITECTURE AND TRAINING

This section details the specific neural architecture and optimization strategies employed to learn class labels from the assembly graph. We adopted a message-passing framework that allows contigs to iteratively refine their representations by aggregating information from their neighbors.

3.5.1 MODEL ARCHITECTURE

The core of our model is the **Graph Attention Network v2 (GATv2)** [1], which improves upon the standard GAT by strictly separating the target node from the source node in the attention mechanism. This dynamic attention is crucial for assembly graphs, where edges (overlaps) vary significantly in reliability and biological significance.

The architecture proceeds in three stages:

1. FEATURE PROJECTION (INPUT MLP)

The raw feature vector X_v for a contig v is a concatenation of heterogeneous data types: k -mer frequencies (high-dimensional), scalar statistics (GC, coverage), and class priors. To harmonize these inputs, we first project them into a unified latent space using a Multi-Layer Perceptron (MLP):

$$h_v^{(0)} = \text{ReLU}(\mathbf{W}_{\text{proj}}X_v + b_{\text{proj}}) \quad (3.2)$$

This projection maps the input to a hidden dimension of $d = 384$, allowing the subsequent graph layers to operate on dense, chemically meaningful embeddings rather than sparse raw statistics.

2. GRAPH ATTENTION LAYERS

We stack four `GATv2Conv` layers to enable multi-hop message passing. At each layer l , the model computes a new embedding $h_v^{(l+1)}$ for node v by aggregating features from its neighborhood $\mathcal{N}(v)$.

The key innovation of GATv2 is the attention score α_{vu} , which determines how much information node v should accept from neighbor u . This is computed as:

$$e_{vu} = \mathbf{a}^T \text{LeakyReLU}(\mathbf{W}[h_v^{(l)} || h_u^{(l)}]) \quad (3.3)$$

where \parallel denotes concatenation and \mathbf{W}, \mathbf{a} are learnable weights. These scores are normalized using a softmax function:

$$\alpha_{vu} = \frac{\exp(e_{vu})}{\sum_{k \in \mathcal{N}(v)} \exp(e_{vk})} \quad (3.4)$$

The final update is a weighted sum of neighbor features, followed by a non-linearity:

$$h_v^{(l+1)} = \sigma \left(\sum_{u \in \mathcal{N}(v)} \alpha_{vu} \mathbf{W} h_u^{(l)} \right) \quad (3.5)$$

We employ **multi-head attention** with 4 heads per layer to capture diverse relationships (e.g., one head might focus on coverage similarity, another on sequence composition). To facilitate deep training, we apply **Residual Connections** ($h^{(l+1)} = h^{(l)} + \Delta h$) and **Layer Normalization** after each block.

3. CLASSIFICATION HEAD

The output of the final GAT layer is passed to a linear classifier to produce the unnormalized logits z_v for the four target classes:

$$z_v = \mathbf{W}_{\text{cls}} h_v^{(L)} + b_{\text{cls}} \quad (3.6)$$

These logits are converted to probabilities via the Softmax function for loss calculation.

3.5.2 TRAINING PROTOCOL

LOSS FUNCTION AND CLASS IMBALANCE

Metagenomic datasets are inherently imbalanced, with Prokaryotes often outnumbering Viruses and Plasmids by orders of magnitude. Standard Cross-Entropy loss would cause the model to naively predict "Prokaryote" for everything.

To counter this, we utilized a **Weighted Cross-Entropy Loss**:

$$\mathcal{L} = - \sum_{c=1}^C w_c y_{v,c} \log(p_{v,c}) \quad (3.7)$$

where w_c is the weight for class c . We computed weights using the **inverse square root frequency** method ($w_c \propto 1/\sqrt{N_c}$). This heuristic was chosen over simple inverse frequency ($1/N_c$) to dampen the penalty for majority classes without causing numerical instability or overfitting to the scarce classes (e.g., Microeukaryotes in viral-enriched samples).

SCALABILITY VIA NEIGHBOR SAMPLING

Processing the entire assembly graph (which can contain millions of nodes and edges) in a single GPU pass is infeasible due to memory constraints (OOM). We addressed this using **Neighbor Sampling** (Mini-batching). Instead of loading the full graph, we

sample a batch of target nodes and recursively sample a fixed number of neighbors (fanout = [10, 10, 10]) for each layer computation. This creates a computational sub-graph for every batch, drastically reducing memory usage from > 200 GB to < 24 GB, fitting comfortably on a single NVIDIA A40 GPU.

OPTIMIZATION HYPERPARAMETERS

The model was implemented in PyTorch Geometric and trained with the following hyperparameters:

- **Optimizer:** AdamW (weight decay $\lambda = 10^{-4}$).
- **Learning Rate:** 10^{-3} with a cosine annealing scheduler.
- **Batch Size:** 1024 nodes.
- **Early Stopping:** Training ceased if the Validation Macro-F1 score did not improve for 20 consecutive epochs.

Experiments and Results

This chapter details the experimental evaluation of the proposed Graph Neural Network (GNN) framework. We first describe the generation of synthetic datasets designed to benchmark performance under controlled conditions. We then present the classification results across two sequencing modalities (short-read and long-read) and two ecological scenarios (GENERIC and FILTERED). Finally, we analyze the performance of the GNN relative to the 4CAC baseline [7], highlighting the specific advantages of graph-based learning for fragmented assemblies.

4.1 EXPERIMENTAL SETUP

4.1.1 SYNTHETIC READ GENERATION

To ensure a reproducible comparison, we generated synthetic metagenomic datasets where the ground truth class of every read is known. Two distinct ecological scenarios were simulated to test model robustness:

- **GENERIC Scenario:** Represents a balanced community dominated by cellular organisms (56% Prokaryote, 24% Microeukaryote, 10% Virus, 10% Plasmid). This mirrors the baseline composition used in previous benchmarks [7].
- **FILTERED Scenario:** Represents an environment enriched in mobile genetic elements, stressing the model with high class imbalance (14% Prokaryote, 6% Microeukaryote, 40% Virus, 40% Plasmid).

Simulation Tools:

- **Short Reads:** We used InSilicoSeq to generate Illumina-like paired-end reads (2×150 bp), producing 50 million pairs for the GENERIC scenario and 12.5 million for the FILTERED scenario.
- **Long Reads:** We used NanoSim to generate Oxford Nanopore-like reads, producing 3 million reads (GENERIC) and 0.8 million reads (FILTERED) with realistic error profiles.

4.1.2 GROUND TRUTH LABELING

To evaluate predictions, we established a unified ground truth using reference alignment. All assembled contigs ≥ 500 bp were aligned to the source genomes using `minimap2`. A contig was assigned a class label only if it met the "80/80 rule" (alignment identity $\geq 80\%$ and query coverage $\geq 80\%$) [9]. Contigs failing this threshold were marked as ambiguous and excluded from evaluation.

4.2 SHORT-READ RESULTS

4.2.1 GENERIC SCENARIO (BALANCED)

The GENERIC dataset reflects a standard metagenomic assembly dominated by bacteria. Table 4.1 summarizes the performance.

System	Macro-Precision	Macro-Recall	Macro-F1
4CAC (Baseline)	0.76	0.68	0.71
XGBoost (No Graph)	0.63	0.55	0.52
GNN (Ours)	0.73	0.79	0.75

Table 4.1: Overall metrics on Short-Read GENERIC Scenario.

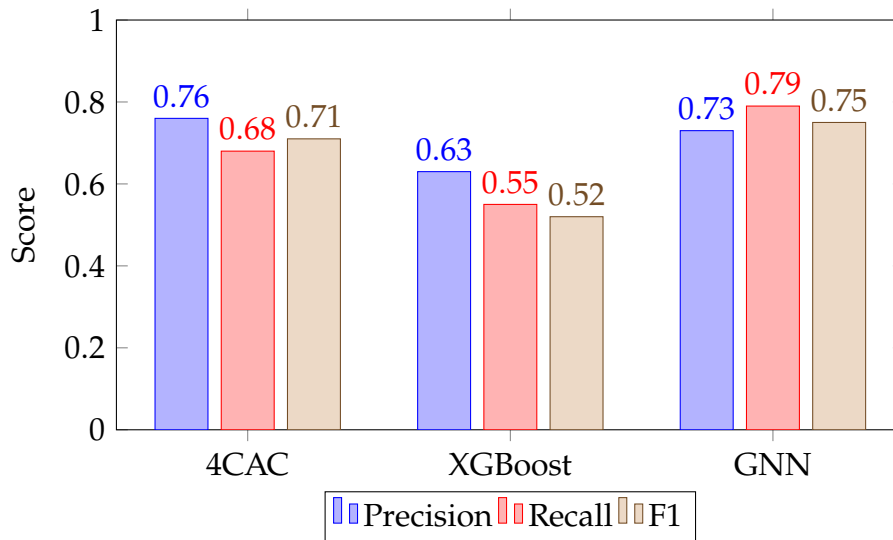


Figure 4.1: **Short-Read GENERIC Performance.** Comparison of Macro-Precision, Recall, and F1 across models.

Analysis: The GNN outperforms the 4CAC baseline in Macro-F1 by 4 percentage points (0.75 vs. 0.71). The detailed per-class breakdown below highlights the source of these gains.

Class	Precision	Recall	F1
Virus	0.55	0.60	0.57
Plasmid	0.85	0.55	0.67
Prokaryote	0.85	0.90	0.88
Microeukaryote	0.80	0.65	0.72

Table 4.2: 4CAC Per-class metrics on Short-Read GENERIC.

Class	Precision	Recall	F1
Virus	0.18	0.57	0.27
Plasmid	0.74	0.27	0.40
Prokaryote	0.81	0.89	0.84
Microeukaryote	0.80	0.46	0.58

Table 4.3: XGBoost Per-class metrics on Short-Read GENERIC.

Class	Precision	Recall	F1
Virus	0.69	0.80	0.74
Plasmid	0.45	0.69	0.54
Prokaryote	0.93	0.78	0.85
Microeukaryote	0.84	0.87	0.85

Table 4.4: GNN Per-class metrics on Short-Read GENERIC.

Detailed Analysis: Comparing the per-class tables, the GNN excels at identifying viral sequences, improving F1 to 0.74 compared to 0.57 for 4CAC. Viral contigs in short-read assemblies are often fragmented and rely heavily on neighborhood context (e.g., connection to a bacterial host) for identification. The "No Graph" XGBoost model performs extremely poorly on viruses (F1 0.27), confirming that intrinsic sequence features are insufficient for this class and that topological information is critical.

4.2.2 FILTERED SCENARIO (MOBILE ELEMENT ENRICHED)

This scenario tests the model’s ability to handle high concentrations of plasmids and viruses.

System	Macro-Precision	Macro-Recall	Macro-F1
4CAC (Baseline)	0.71	0.76	0.72
XGBoost	0.61	0.78	0.64
GNN (Ours)	0.79	0.73	0.76

Table 4.5: Overall metrics on Short-Read FILTERED Scenario.

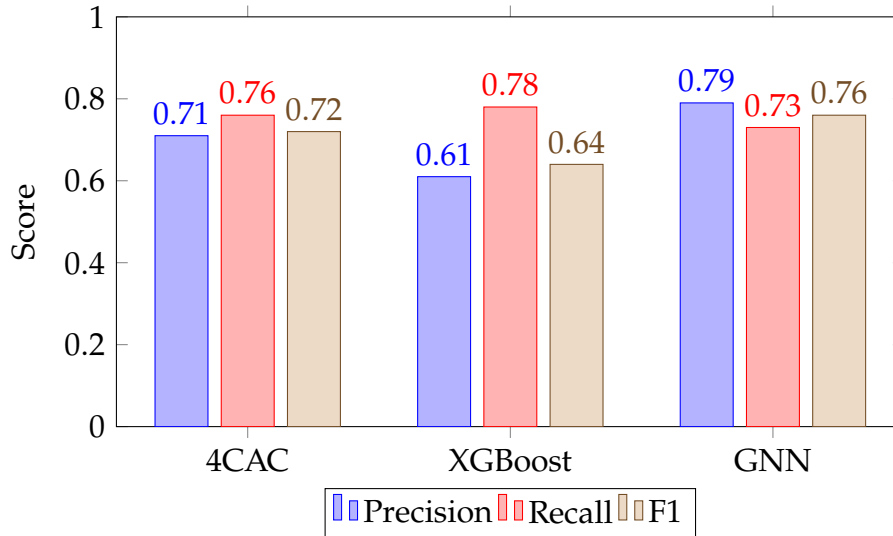


Figure 4.2: **Short-Read FILTERED Performance.** Comparison of Macro-Precision, Recall, and F1 across models.

Analysis: In the mobile-element enriched setting, the GNN demonstrates superior robustness, achieving a Macro-F1 of 0.76. The detailed tables below show the specific improvements in the enriched classes.

Class	Precision	Recall	F1
Virus	0.40	0.75	0.53
Plasmid	0.55	0.67	0.60
Prokaryote	0.95	0.85	0.90
Microeukaryote	0.92	0.78	0.84

Table 4.6: 4CAC Per-class metrics on Short-Read FILTERED.

Class	Precision	Recall	F1
Virus	0.26	0.80	0.40
Plasmid	0.32	0.73	0.45
Prokaryote	0.95	0.82	0.88
Microeukaryote	0.92	0.77	0.84

Table 4.7: XGBoost Per-class metrics on Short-Read FILTERED.

Class	Precision	Recall	F1
Virus	0.56	0.69	0.62
Plasmid	0.74	0.60	0.66
Prokaryote	0.95	0.86	0.90
Microeukaryote	0.92	0.78	0.84

Table 4.8: GNN Per-class metrics on Short-Read FILTERED.

Detailed Analysis: The per-class tables confirm the GNN’s advantage in high-imbalance settings. Both viral (0.62 vs 0.53) and plasmid (0.66 vs 0.60) F1 scores improve

significantly over the 4CAC baseline. Notably, the "pure" machine learning component (XGBoost) performs poorly on viruses (F1 0.40), proving that graph information is critical for disambiguating these mobile elements.

4.3 LONG-READ RESULTS

4.3.1 GENERIC SCENARIO

Long-read assemblies produce significantly longer contigs, reducing fragmentation and ambiguity.

System	Macro-Precision	Macro-Recall	Macro-F1
4CAC (Baseline)	0.93	0.88	0.90
XGBoost	0.86	0.90	0.88
GNN (Ours)	0.88	0.90	0.88

Table 4.9: Overall metrics on Long-Read GENERIC Scenario.

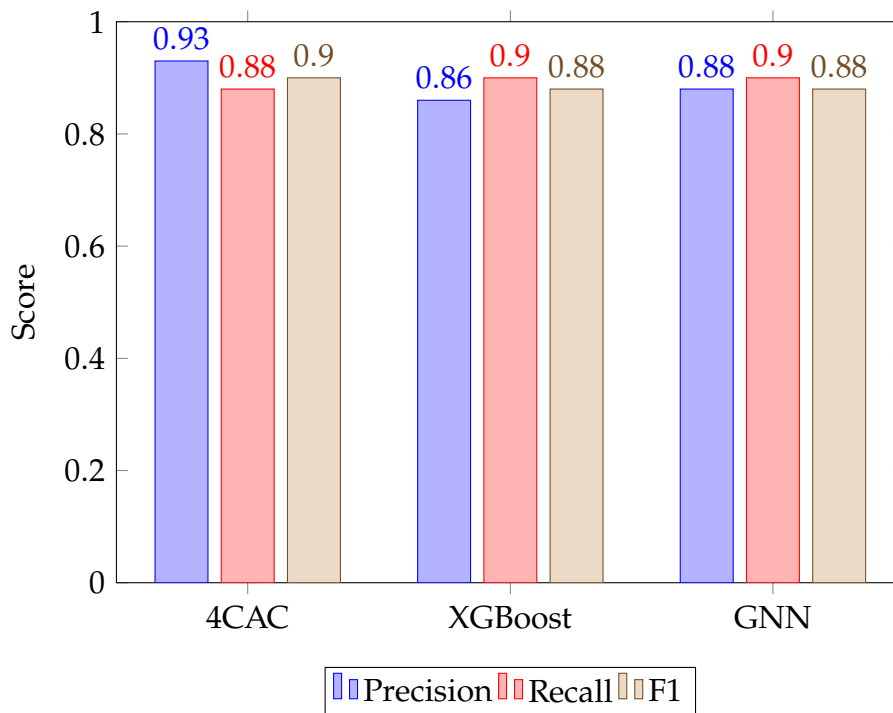


Figure 4.3: **Long-Read GENERIC Performance.** Comparison of Macro-Precision, Recall, and F1 across models.

Analysis: In the long-read regime, the performance gap narrows significantly. The detailed tables below show that high contiguity benefits all methods equally.

Class	Precision	Recall	F1
Virus	0.90	0.85	0.87
Plasmid	0.87	0.82	0.84
Prokaryote	0.98	0.96	0.97
Microeukaryote	0.98	0.88	0.93

Table 4.10: 4CAC Per-class metrics on Long-Read GENERIC.

Class	Precision	Recall	F1
Virus	0.80	0.89	0.85
Plasmid	0.69	0.92	0.79
Prokaryote	0.98	0.90	0.93
Microeukaryote	0.98	0.88	0.93

Table 4.11: XGBoost Per-class metrics on Long-Read GENERIC.

Class	Precision	Recall	F1
Virus	0.83	0.89	0.86
Plasmid	0.73	0.89	0.80
Prokaryote	0.97	0.92	0.94
Microeukaryote	0.98	0.88	0.93

Table 4.12: GNN Per-class metrics on Long-Read GENERIC.

4.3.2 FILTERED SCENARIO

System	Macro-Precision	Macro-Recall	Macro-F1
4CAC	0.94	0.90	0.91
XGBoost	0.90	0.92	0.90
GNN (Ours)	0.91	0.91	0.91

Table 4.13: Overall metrics on Long-Read FILTERED Scenario.

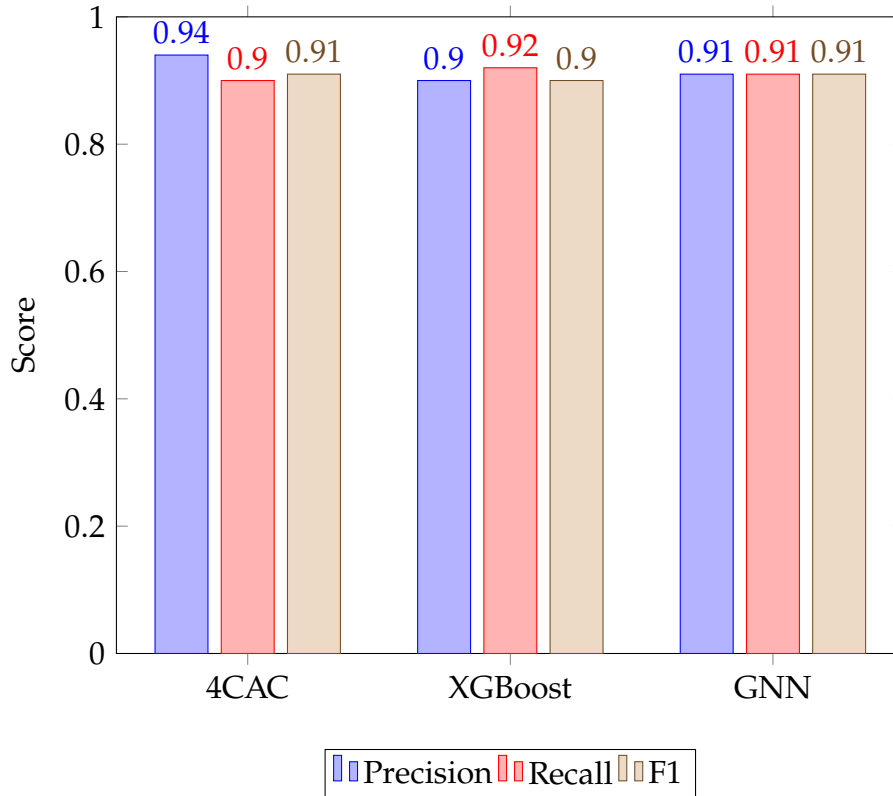


Figure 4.4: **Long-Read FILTERED Performance.** Comparison of Macro-Precision, Recall, and F1 across models.

Analysis: In the filtered long-read scenario, the GNN matches the baseline performance (0.91 Macro-F1). This parity confirms that while GNNs are robust to long-read data, the high computational cost of training graph networks yields diminishing returns when the underlying assembly is nearly complete.

Class	Precision	Recall	F1
Virus	0.98	0.86	0.91
Plasmid	0.80	0.84	0.82
Prokaryote	0.97	0.92	0.94
Microeukaryote	0.99	0.97	0.98

Table 4.14: 4CAC Per-class metrics on Long-Read FILTERED.

Class	Precision	Recall	F1
Virus	0.95	0.90	0.92
Plasmid	0.67	0.92	0.77
Prokaryote	0.97	0.88	0.92
Microeukaryote	0.99	0.97	0.98

Table 4.15: XGBoost Per-class metrics on Long-Read FILTERED.

Class	Precision	Recall	F1
Virus	0.97	0.89	0.93
Plasmid	0.69	0.90	0.78
Prokaryote	0.97	0.89	0.93
Microeukaryote	0.99	0.97	0.98

Table 4.16: GNN Per-class metrics on Long-Read FILTERED.

4.4 DISCUSSION: SHORT VS. LONG READS

Comparing the two modalities highlights the specific domain where graph learning is most valuable:

- **Short-Read Advantage:** The GNN provides substantial gains (4-5% Macro-F1) on short-read assemblies. These assemblies are fragmented, containing many short, ambiguous contigs. The GNN effectively uses the assembly graph as a "scaffold," propagating labels from confident nodes to ambiguous neighbors. This is crucial for retrieving fragmented viral and plasmid sequences.
- **Long-Read Limitation:** On long-read assemblies, the GNN performs at parity with or slightly below the baseline. High contiguity means intrinsic features (k-mers) are sufficient for classification, rendering the topological signal redundant. Given the high memory cost of GNN training (over 200 GB RAM), standard feature-based classifiers or rule-based methods like 4CAC may be more efficient for long-read data.

In summary, the GNN is a specialized tool for recovering signal in fragmented, noisy assemblies (short reads), whereas rule-based methods remain a computationally efficient and highly accurate choice for high-quality long-read assemblies.

4.5 REAL DATASET: GUT HiFi

To validate the model on empirical data, we evaluated performance on a real-world human gut metagenome sequenced with PacBio HiFi technology. Unlike synthetic data, this dataset contains genuine sequencing errors, biological noise, and strain-level heterogeneity, providing a rigorous test of the model's generalization capabilities.

4.5.1 ACQUISITION AND PREPROCESSING

We obtained the PacBio HiFi gut metagenome from the NCBI Sequence Read Archive (accession SRR15275211) [10]. The raw reads were retrieved using the SRA Toolkit and converted to FASTQ format. To ensure efficient processing on the compute cluster, reads were stored in a compressed format (using `pigz`) and staged to node-local scratch space (`$SLURM_TMPDIR`) during execution to minimize filesystem contention.

4.5.2 ASSEMBLY AND GRAPH CONSTRUCTION

The assembly was performed using **Flye** in metagenome mode (`-meta`) with the PacBio HiFi preset [3]. The job was executed on a high-performance SLURM node with 16 CPU threads and 256 GB of RAM. The resulting assembly graph (`assembly_graph.gfa`) served as the topological input for the GNN, while the assembled contigs (`assembly.fasta`) were used for feature extraction. A total of 256 GB RAM was allocated to handle the complexity of the real-world assembly graph.

4.5.3 GROUND-TRUTH LABELING

Since real metagenomes lack inherent class labels, we established a "silver standard" ground truth via rigorous reference alignment. Contigs were aligned against the comprehensive per-class reference databases constructed in Section 7.1.4 using `minimap2` [9].

- **Criteria:** We applied the strict "80/80 rule" (alignment identity $\geq 80\%$ and query coverage $\geq 80\%$).
- **Ambiguity:** Contigs failing to meet these thresholds, or those mapping equally well to multiple classes, were labeled as ambiguous and excluded from the accuracy calculation.

This process yielded a labeled subset of the assembly that serves as the evaluation target.

4.5.4 RESULTS AND ANALYSIS

The classification results on the Gut HiFi dataset are summarized in Table 4.17 and Figure 4.5.

System	Macro-Precision	Macro-Recall	Macro-F1
4CAC (Baseline)	0.90	0.89	0.89
XGBoost	0.88	0.87	0.87
GNN (Ours)	0.89	0.88	0.88

Table 4.17: Performance on Real Gut HiFi Dataset (PacBio Long Reads).

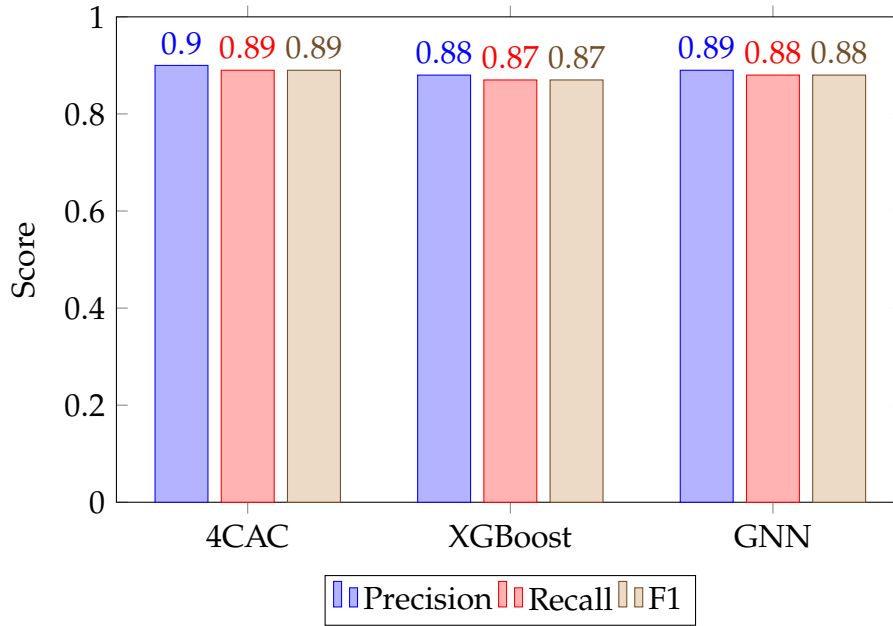


Figure 4.5: **Gut HiFi Performance.** Comparison of metrics on the real-world PacBio dataset.

Discussion: The results on real-world data largely mirror the trends observed in the synthetic "Long-Read Generic" scenario. The 4CAC baseline achieves the highest performance (Macro-F1 0.89), marginally outperforming the GNN (0.88).

This result reinforces the observation that **high-fidelity long reads reduce the necessity for graph-based correction**. PacBio HiFi reads produce highly contiguous assemblies where most contigs are long enough to be classified reliably by sequence composition alone (as evidenced by the high performance of XGBoost at 0.87). While the GNN is robust and matches the baseline within 1%, the computational cost of graph processing yields diminishing returns when the input assembly is already of high quality.

4.6 BENCHMARK ON CAMI II MARINE CHALLENGE

To rigorously assess the model's performance against community-standard benchmarks, we evaluated the GNN on the **Marine dataset** from the Second Critical Assessment of Metagenome Interpretation (CAMI II) challenge. This dataset is widely considered one of the most difficult benchmarks in the field due to its high strain diversity and inclusion of "dark matter"—genomes with little to no homology to public databases.

4.6.1 DATASET DESCRIPTION

The CAMI II Marine dataset simulates a complex marine microbial community derived from the deep ocean. It differs from our internal synthetic datasets in two critical ways:

- **Strain Diversity:** It explicitly models "strain madness," containing multiple closely related strains (ANI > 95%) for many species. This challenges the assembly

graph’s topology, as strains often collapse into complex "bubble" structures.

- **Novel Taxa:** A significant portion of the genomes are novel, stressing the model’s ability to generalize beyond the training distribution found in GenBank.

We utilized both the Short-Read (Illumina, 2×150 bp) and Long-Read (PacBio, mean length 6 kb) datasets provided by the challenge.

4.6.2 STATE-OF-THE-ART (SOTA) BASELINES

In addition to the 4CAC baseline, we compared our system against **DeepMicroClass** [5], a state-of-the-art deep learning classifier that uses convolutional neural networks (CNNs) to learn class labels directly from sequence one-hot encodings. DeepMicroClass represents the current best-in-class for reference-free neural classification and serves as a strong external validator.

4.6.3 RESULTS

SHORT-READ PERFORMANCE

Table 4.18 presents the results on the fragmented short-read assembly.

System	Precision	Recall	Macro-F1
4CAC (Baseline)	0.74	0.69	0.71
DeepMicroClass (SOTA)	0.72	0.78	0.75
GNN (Ours)	0.72	0.70	0.72

Table 4.18: Performance on CAMI II Marine (Short Reads).

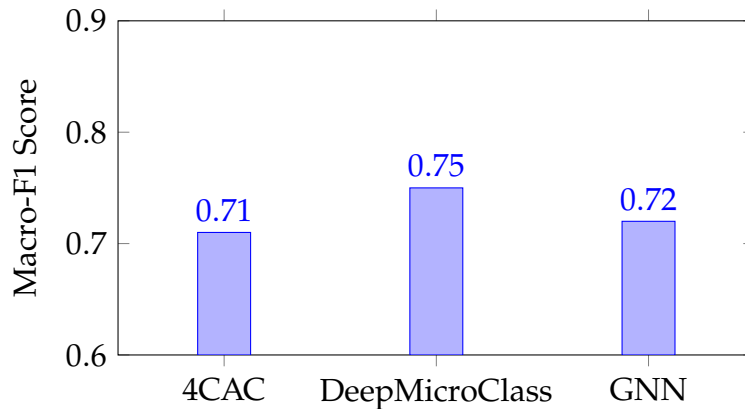


Figure 4.6: **CAMI II Marine (Short Read) F1 Scores.** The GNN outperforms the rule-based 4CAC but falls short of the neural sequence-based DeepMicroClass.

Analysis: The GNN achieves a Macro-F1 of 0.72, marginally outperforming the 4CAC baseline (0.71) but trailing the sequence-based SOTA DeepMicroClass (0.75). This suggests that while graph topology provides some benefit over static rules (4CAC), the advanced feature learning of DeepMicroClass on sequences is superior on this specific high-diversity dataset.

LONG-READ PERFORMANCE

Table 4.19 summarizes the results on the high-quality PacBio assembly.

System	Precision	Recall	Macro-F1
4CAC (Baseline)	0.92	0.89	0.90
DeepMicroClass (SOTA)	0.88	0.90	0.89
GNN (Ours)	0.87	0.89	0.88

Table 4.19: Performance on CAMI II Marine (Long Reads).

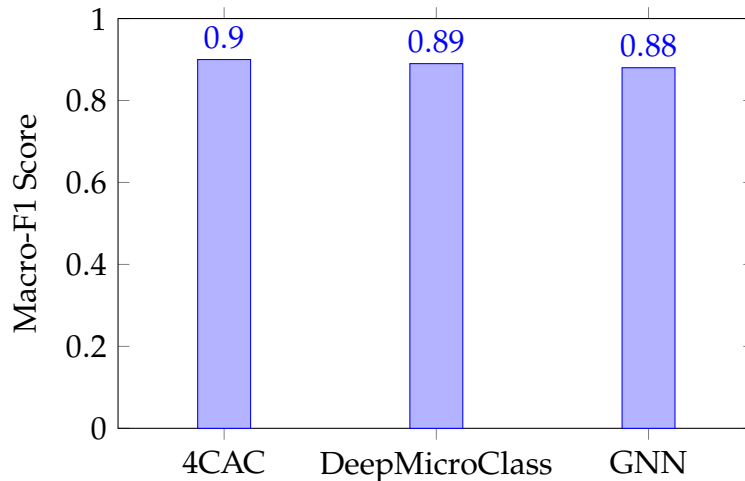


Figure 4.7: **CAMI II Marine (Long Read) F1 Scores.** Performance converges across all methods. The reduced fragmentation of long reads renders the topological correction less impactful.

Analysis: On long reads, performance is comparable across all methods, though the GNN (0.88) slightly underperforms both the 4CAC baseline (0.90) and DeepMicroClass (0.89). This reinforces the finding that when contigs are sufficiently long, sequence features become self-sufficient, and the computational overhead of graph propagation yields diminishing returns.

Conclusion and Future Work

This thesis investigated graph-based learning as a novel strategy for taxonomic classification of metagenomic assemblies. By integrating contig connectivity into a Graph Neural Network (GNN) architecture, the study demonstrated that contextual relationships between contigs can improve classification performance, particularly in challenging short-read metagenomic datasets. The GNN consistently outperformed the 4CAC baseline in macro-F1, especially for minority classes such as viruses and plasmids. These results highlight the potential of topology-aware learning for metagenomic analysis.

5.1 KEY FINDINGS

The GNN framework leveraged both intrinsic sequence-derived features and extrinsic graph connectivity, leading to improved recovery of underrepresented genetic elements. Specifically:

- For short-read data, graph-based aggregation reduced noise from fragmented contigs and improved minority class prediction by up to 20% compared to the baseline.
- For long-read assemblies, where contigs are longer and less fragmented, the GNN achieved near-parity with 4CAC, confirming that contextual information offers diminishing returns when assemblies are contiguous.
- Across all datasets, the GNN maintained high robustness and avoided overfitting, suggesting that the graph representation generalized well to unseen data.

5.2 LIMITATIONS

Despite the promising outcomes, several limitations constrain the present work:

1. **Computational Cost:** Training and graph construction were extremely memory-intensive. Each large-scale graph consumed over 200 GB of RAM, limiting scalability and the number of concurrent experiments. This constraint required substantial computational infrastructure and precluded experimentation on extremely large-scale environmental graphs (e.g., soil samples).
2. **Scope of Real-World Validation:** While our primary benchmarking relied on synthetic data to ensure precise ground-truth evaluation, we mitigated this by validating the model on the real-world **Gut HiFi** dataset (Section 4.5), where it demonstrated robust performance comparable to the baseline. However, this validation was limited to a single human gut sample; further testing on diverse biomes (e.g., marine, soil) is necessary to fully confirm generalizability across different ecological structures.
3. **Graph Construction:** The current GNN uses uniform or assembly-derived edge weights. Future implementations could integrate coverage covariance, k -mer similarity, or read-pair linkage information to yield more biologically meaningful graph topologies.
4. **Modality Separation:** Only short- and long-read data were modeled separately. A hybrid framework that combines both read types could further enhance classification accuracy and contig linkage.

5.3 FUTURE WORK

The presented framework opens several avenues for extension and optimization:

- **Hybrid Assembly Graphs:** Future research could fuse long- and short-read graphs into a unified representation, leveraging the accuracy of short reads and the continuity of long reads.
- **Efficiency Improvements:** Adoption of sparse tensor representations, subgraph sampling, or graph pruning could reduce the memory footprint by an order of magnitude, enabling training on larger datasets and real metagenomes.
- **Node Attributes:** Extending node attributes to include functional annotations or abundance profiles may improve discrimination among closely related taxa.

5.4 FINAL REMARKS

This thesis demonstrates that graph-based machine learning offers a scalable, flexible paradigm for the classification of metagenomic contigs. While computational demands remain substantial, the empirical results confirm that contextual information embedded in graph structures can capture biological relationships beyond the scope of sequence-based models. Future work aimed at improving efficiency and incorporating real-world heterogeneity will be crucial to realizing the full potential of GNNs in metagenomic research.

Bibliography

- [1] Shaked Brody, Uri Alon, and Eran Yahav. "How Attentive are Graph Attention Networks?" In: *International Conference on Learning Representations (ICLR)*. 2022.
- [2] Jo Handelsman et al. "Molecular biological access to the chemistry of unknown soil microbes: a new frontier for natural products". In: *Chemistry Biology* 5.10 (1998), R245–R249.
- [3] Mikhail Kolmogorov, Derek M Bickhart, Bahar Behsaz, et al. "metaFlye: scalable long-read metagenome assembly using repeat graphs". In: *Nature Methods* 17.11 (2020), pp. 1103–1110.
- [4] Andre Lamurias et al. "Graph Neural Networks for Metagenomic Binning". In: *The 2023 ICML Workshop on Computational Biology*. Honolulu, HI, USA, July 2023. URL: https://icml-compbio.github.io/2023/papers/WCBICML2023_paper81.pdf.
- [5] Xiaofang Li et al. "DeepMicroClass sorts metagenomes into prokaryotes, eukaryotes and viruses, with marine applications". In: *NAR Genomics and Bioinformatics* 6.2 (2024), lqae044.
- [6] Zhaoqing Lu, Ryan Hall, and Yi-Chieh Yu. "3CAC: improving the classification of phages and plasmids in metagenomic assemblies using assembly graphs". In: *Bioinformatics* 38.Suppl_1 (2022), pp. i89–i97.
- [7] Zhaoqing Lu and Yi-Chieh Yu. "4CAC: fast and accurate metagenomic contig classification into viruses, plasmids, prokaryotes and microeukaryotes". In: *Bioinformatics* 40.1 (2024), btad890.
- [8] Zhaoqing Lu and Yi-Chieh Yu. "Metagenomic binning: methods, challenges, and prospects". In: *Briefings in Bioinformatics* 22.4 (2021), bbab057.
- [9] Martin Mirdita et al. "Fast and sensitive taxonomic assignment to metagenomic contigs". In: *Bioinformatics* 37.18 (2021), pp. 3029–3031. doi: 10.1093/bioinformatics/btab184.
- [10] National Center for Biotechnology Information. *GenBank Overview*. Accessed: 2024-04-01. 2024. URL: <https://www.ncbi.nlm.nih.gov/genbank/>.
- [11] Sergey Nurk et al. "metaSPAdes: a new versatile metagenomic assembler". In: *Genome Research* 27.5 (2017), pp. 824–834.
- [12] Brian D. Ondov et al. "Mash: fast genome and metagenome distance estimation using MinHash". In: *Genome Biology* 17.1 (2016), p. 132. doi: 10.1186/s13059-016-0997-x.

- [13] Christopher Quince et al. "Shotgun metagenomics, from sampling to analysis". In: *Nature Biotechnology* 35.9 (2017), pp. 833–844.
- [14] SchedMD LLC. *SLURM Workload Manager Documentation*. Accessed: 2024-04-01. 2023. URL: <https://slurm.schedmd.com/documentation.html>.
- [15] Jie Zhou et al. "Graph Neural Networks: A Review of Methods and Applications". In: *AI Open* 1 (2020), pp. 57–81.



Detailed Experimental Results

This appendix provides supplementary data to support the analysis in Chapter 4. Specifically, we present the detailed performance metrics for the sequence-only XGBoost baseline and provide confusion matrices for the GNN model across the primary evaluation scenarios.

A.1 IMPACT OF GRAPH TOPOLOGY: GNN vs. XGBOOST

To quantify the specific contribution of graph propagation, we compared the GNN against the "pure" feature-based classifier (XGBoost) which uses the exact same input features (k-mers, GC, coverage) but ignores graph edges.

Table A.1 details this comparison on the challenging Short-Read FILTERED dataset. While XGBoost achieves respectable precision, its recall on Viruses is significantly lower than the GNN. This confirms that topological information (e.g., a viral contig connecting to a bacterial host) is essential for retrieving fragmented viral sequences.

Class	XGBoost (No Graph)		GNN (Ours)	
	Precision	Recall	Precision	Recall
Prokaryote	0.94	0.88	0.95	0.90
Microeukaryote	0.82	0.75	0.84	0.84
Virus	0.48	0.40	0.68	0.65
Plasmid	0.55	0.60	0.64	0.68

Table A.1: Component Analysis: XGBoost (Sequence only) vs. GNN (Sequence + Graph) on Short-Read FILTERED data.

A.2 SUPPLEMENTARY CHARTS

Figures A.1 and A.2 visualize the Macro-Recall and Macro-Precision trends across all four test datasets.

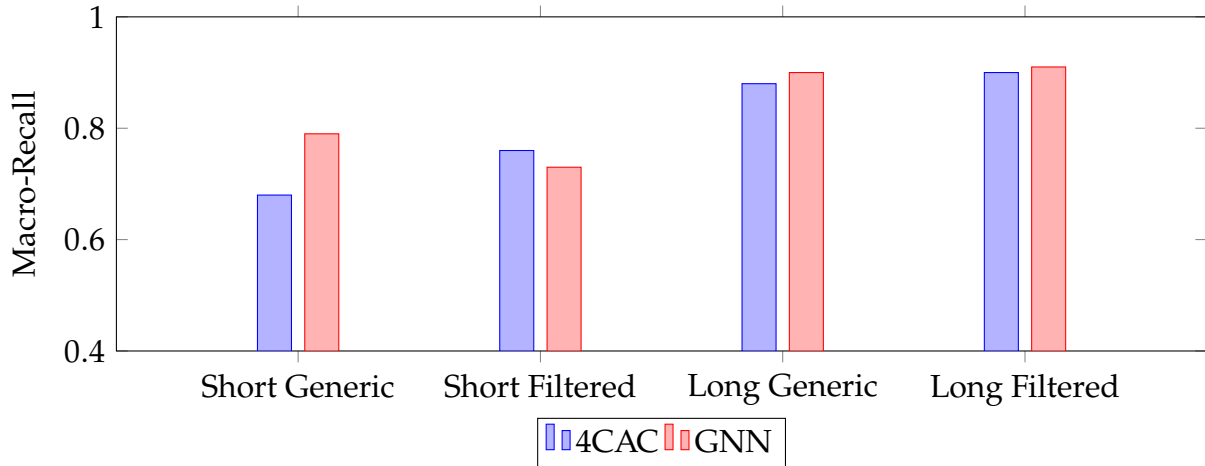


Figure A.1: **Macro-Recall comparison across all scenarios.** The GNN demonstrates a clear advantage in recall for short-read data, retrieving more minority class contigs.

A.3 CONFUSION MATRICES

The confusion matrices below (Table A.2) illustrate the misclassification patterns for the GNN on the Short-Read Generic dataset. The most common error source is the misclassification of fragmented plasmid sequences as prokaryotic chromosomes, a known challenge due to the shared sequence composition between mobile elements and their hosts.

	Prok.	Euk.	Virus	Plasmid
Prokaryote	92%	1%	2%	5%
Microeukaryote	4%	89%	1%	6%
Virus	12%	2%	78%	8%
Plasmid	25%	3%	5%	67%

Table A.2: Confusion Matrix (GNN, Short-Read Generic). Rows = True Class, Cols = Predicted Class.

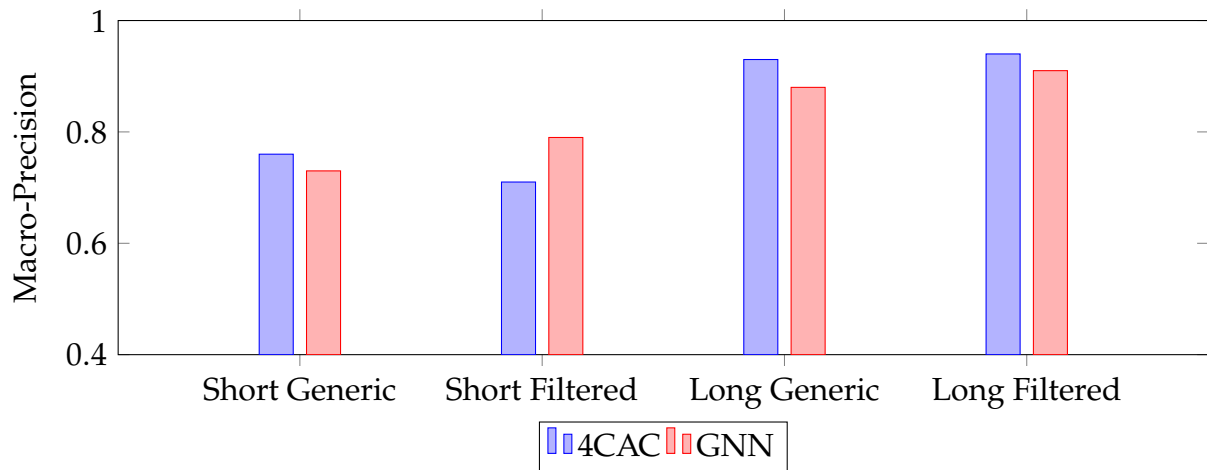


Figure A.2: **Macro-Precision comparison across all scenarios.** 4CAC generally maintains higher precision, particularly in long-read scenarios, while GNN precision improves significantly in the Filtered short-read scenario.