FEDERAL UNIVERSITY OF CEARÁ

CENTER OF TECHNOLOGY

DEPARTMENT OF TELECOMMUNICATIONS

UNDERGRADUATE COURSE IN COMPUTER ENGINEERING

LUCAS VITORIANO DE QUEIROZ LIRA

GRAPH NEURAL NETWORK FOR GENE-DISEASE LINK PREDICTION

FORTALEZA

2025

LUCAS VITORIANO DE QUEIROZ LIRA

GRAPH NEURAL NETWORK FOR GENE-DISEASE LINK PREDICTION

Undergraduate Thesis submitted to the Computer Engineering Course of the Center of Technology of the Federal University of Ceará, as a partial requirement for obtaining the Bachelor Degree in Computer Engineering.

Advisor: Prof. Dr. Victor Hugo C. de Albuquerque

FORTALEZA

2025

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

L745g Lira, Lucas Vitoriano de Queiroz.
Graph neural network for gene-disease link prediction / Lucas Vitoriano de Queiroz Lira. – 2025.
81 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Centro de Tecnologia, Curso de Engenharia de Computação, Fortaleza, 2025.
Orientação: Prof. Dr. Victor Hugo Costa de Albuquerque.

1. Redes neurais em grafos. 2. GNN. 3. Previsão de conexões. 4. Completude de grafos de conhecimento. 5. Grafos homogêneo. I. Título.

CDD 621.39

LUCAS VITORIANO DE QUEIROZ LIRA

GRAPH NEURAL NETWORK FOR GENE-DISEASE LINK PREDICTION

Undergraduate Thesis submitted to the Computer Engineering Course of the Center of Technology of the Federal University of Ceará, as a partial requirement for obtaining the Bachelor Degree in Computer Engineering.

Approved on:

EXAMINATION BOARD

Prof. Dr. Victor Hugo C. de Albuquerque (Advisor)
Federal University of Ceará (UFC)

Dr. Rômulo Cesar Cunha Lima
Instituto Federal Do Ceará (IFCE)

Dr. Marcello Carvalho dos Reis
Universidade Federal do Ceará (UFC)

ACKNOWLEDGEMENTS

I am sincerely grateful to the professors who guided my development. I thank Professor Cortez for sparking my desire for academic excellence through the PET group, Professor Danielo for his crucial help with the Dual Degree application, and Professors Jardel and Alexandre for their vital long-distance support that ensured a successful exchange. My gratitude also extends to Professor Jarbas for the significant professional experience I gained in the Hardware Club and with the Elmo 2.0 project.

I express my gratitude to Professor Victor Hugo for his attentive guidance and constant availability throughout the development of this Final Course Project.

To CAPES, for the funding that made the exchange program possible.

To my parents, brother, and girlfriend, who, even during my moments of absence dedicated to my studies in Brazil and abroad, always supported me unconditionally. Your support gave me the necessary strength to carry on and showed me that every sacrifice made for my education is worthwhile.

To the friends I have made over these years, who made the journey lighter and more joyful. I thank you for the hours of study, the exchange of information, and the companionship that made my graduation easier.

Finally, I thank all the professors at UFC and CentraleSupélec, who shared their technical knowledge and motivated me throughout my undergraduate studies, serving as guides for my academic and professional success.

"The mystery of human existence lies not in just staying alive, but in finding something to live for."

(Fyodor Dostoyevsky)

ABSTRACT

This study explores the use of Graph Neural Networks Graph Neural Network (GNN) for identifying gene-disease associations through a Knowledge Graph derived from the Stanford Biomedical Network Dataset Collection, which includes originally over 21,000 connections. By representing genes and diseases as nodes, and their relationships as edges, we enable GNNs to uncover complex patterns and predict novel associations. This approach not only reveals previously unknown gene-disease links but also offers fresh insights into the molecular basis of diseases. The specific disease-gene association network analyzed comprises 7,813 nodes and 21,357 edges. Initial data preprocessing involved standardizing disease features and augmenting them using Large Language Models for attributes like disease class, alongside enhancing gene features by parsing chromosomal location data. The methodology evolved from heterogeneous graph models to a primary focus on homogeneous graph representations, unifying node types and their 512-dimensional features. Custom GNNs, employing backbones such as GraphConv, SAGEConv, and GATv2Conv, were developed, primarily utilizing a Multi-Layer Perceptron Multi-Layer Perceptron (MLP) based decoder for link prediction. This refined approach, particularly with simpler homogeneous GNN models, demonstrated significantly improved generalization and high validation performance, achieving an AUC-PR of 0.991 and an F1-Score of 0.950, effectively addressing earlier overfitting challenges. Evaluation employs comprehensive metrics including AUC-ROC, AUC-PR, F1-Score, and Precision@10. The research underscores the strong potential of tailored GNNs for discovering novel gene-disease links, with ongoing work focusing on further optimization and interpretability.

Keywords: Graph Neural Networks GNN. Link Prediction. Knowledge Graph Completion. Gene-Disease Associations. Bioinformatics. PyTorch Geometric. Homogeneous Graphs.

RESUMO

Este estudo explora o uso de Redes Neurais em Grafos GNN para identificar associações gene-doença por meio da completude de um Grafo de Conhecimento derivado da Stanford Biomedical Network Dataset Collection, que inclui originalmente mais de 21.000 conexões. Ao representar genes e doenças como nós, e seus relacionamentos como arestas, as GNNs permitem descobrir padrões complexos e prever novas associações. Esta abordagem não apenas revela conexões gene-doença anteriormente desconhecidas, mas também oferece novos insights sobre a base molecular das doenças. A rede de associação gene-doença específica analisada compreende 7.813 nós e 21.357 arestas. O pré-processamento inicial dos dados envolveu a padronização de características de doenças e sua augmentação utilizando Modelos de Linguagem de Grande Escala Large Language Model (LLM), juntamente com a melhoria das características genéticas a partir da análise de dados de localização cromossômica. A metodologia evoluiu de modelos de grafos heterogêneos para um foco principal em representações homogêneas, unificando tipos de nós e adaptando arquiteturas GNN. GNNs personalizadas, empregando backbones como GraphConv, SAGEConv e GATv2Conv, foram desenvolvidas, utilizando primariamente um decodificador baseado em Perceptron Multicamadas MLP para a previsão de conexões. Esta abordagem refinada, particularmente com modelos GNN homogêneos mais simples, demonstrou generalização significativamente melhorada e alto desempenho de validação, alcançando um AUC-PR de 0,991 e um F1-Score de 0,950, superando eficazmente os desafios iniciais de overfitting. A avaliação emprega métricas abrangentes, incluindo AUC-ROC, AUC-PR, F1-Score e Precisão@10. A pesquisa ressalta o forte potencial de GNNs personalizadas para descobrir novas conexões gene-doença, com trabalhos em andamento focados em maior otimização e interpretabilidade.

Palavras-chave: Redes Neurais em Grafos GNN. Previsão de Conexões. Completude de Grafos de Conhecimento. Associações Gene-Doença. Bioinformática. PyTorch Geometric. Grafos Homogêneos.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

AUC-PR	Area Under the Precision-Recall Curve
AUC-ROC	Area Under the Receiver Operating Characteristic Curve
FPR	False Positive Rate
GAT	Graph Attention Network
GNN	Graph Neural Network
HGNC	HUGO Gene Nomenclature Committee
LLM	Large Language Model
ML	Machine Learning
MLP	Multi-Layer Perceptron
P@10	Precision at 10
PCA	Principal Component Analysis
TPR	True Positive Rate
UMAP	Uniform Manifold Approximation and Projection
ViT	Vision Transformer
XAI	Explainable AI

CONTENTS

1 INTRODUCTION

The ongoing revolution in genomic medicine, powered by high-throughput sequencing technologies, has led to an unprecedented data deluge. This flood of data promises to elucidate the molecular basis of human diseases, heralding a new era of personalized medicine (TAHERDOOST; GHOFRANI, 2024; PAREKH *et al.*, 2023). However, the sheer volume and complexity of this data present a formidable analytical challenge.

Historically, computational methods like Genome-Wide Association Studies have been instrumental, yet they often analyze genetic markers in isolation, a known limitation that can obscure the full genetic architecture of complex diseases (TAM *et al.*, 2019). Similarly, traditional machine learning models frequently struggle to effectively leverage the complex, relational topology inherent in biological systems, often treating genes and diseases as independent entities and thereby overlooking crucial interaction patterns. Biological reality, however, is a network of intricate interactions (BARABáSI *et al.*, 2011). Graph Neural Networks GNN have emerged as a paradigm-shifting class of deep learning models specifically designed to overcome these limitations. By representing entities such as genes and diseases as nodes and their complex relationships as edges, GNNs can learn rich, context-aware representations directly from the network structure, enabling the prediction of new associations with high precision in tasks ranging from disease analysis to drug design (XIONG *et al.*, 2021; ZHOU *et al.*, 2020).

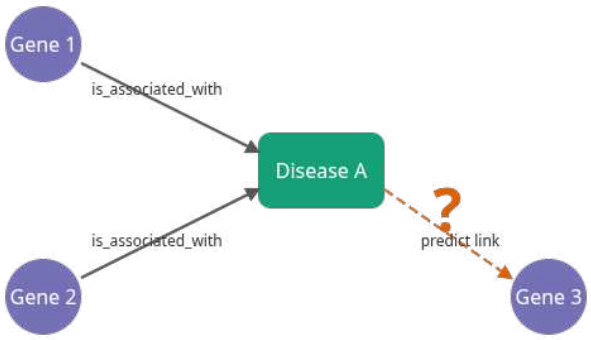
This work, illustrated by the conceptual model in Figure 1, delves into this frontier by developing and evaluating a GNN framework for link prediction within a large-scale gene-disease association network. Based on the Stanford Biomedical Network Dataset Collection (ZITNIK *et al.*, 2018), a knowledge graph was constructed. The proposed methodology is distinguished by several key contributions designed to address critical challenges in the field.

First, we systematically investigate the trade-off between model complexity and performance by evolving our approach from sophisticated heterogeneous graph models to a more streamlined and ultimately more powerful homogeneous graph representation. This simplification, which unifies node types, proved critical for improving model generalization and overcoming initial overfitting challenges.

Second, to combat the inherent sparsity and disconnectedness often found in biological knowledge graphs, we integrate a virtual node (GILMER *et al.*, 2017). This technique enhances the model's ability to capture global, long-distance relationships by creating a global information hub, a method gaining traction for robust learning on large graphs.

Third, in a novel approach to feature engineering, we leverage Language Models LLM for semantic enrichment of node features. By using state-of-the-art models to generate crucial attributes like disease class and main affected system, we augment the available data, showcasing a new frontier in preparing biomedical data for graph machine learning.

Data preprocessing and feature engineering were foundational, involving not only LLM-based enrichment but also the generation of high-quality embeddings using domain-specific models like Bio-ClinicalBERT (ALSENTZER *et al.*, 2019). The subsequent evaluation of our models is performed using a comprehensive suite of metrics, including Area Under the Receiver Operating Characteristic Curve (AUC-ROC), Area Under the Precision-Recall Curve (AUC-PR), F1-Score, and Precision@10, ensuring a robust and context-appropriate analysis of predictive performance, particularly crucial for imbalanced datasets common in this domain (SAITO; REHMSMEIER, 2015).



Source: Image from the author (2025)

Figure 1 – A simplified representation of the gene-disease knowledge graph . The model learns from existing connections (solid lines) to predict potential new associations (dashed line), addressing the link prediction task.

Objectives

General Objective

The general objective of this work is to develop and evaluate a model based on Graph Neural Networks for the prediction of new associations between genes and diseases, using a treated and enriched biomedical knowledge graph.

Specific Objectives

The specific objectives of this study are to perform the preprocessing and standardization of gene and disease data from the *Stanford Biomedical Network Dataset Collection*; enrich the node features using Large Language Models and data extraction techniques; implement and compare different GNN architectures (*GraphConv*, *SAGEConv*, *GATv2Conv*) in an *encoder-decoder* framework for the link prediction task; evaluate the impact of converting the heterogeneous graph to a homogeneous representation and adding a virtual node on the models' performance; and, finally, analyze the results obtained through evaluation metrics to identify the most promising approaches for the knowledge graph completion task.

Thesis Structure

This thesis is organized as follows:

Chapter 2 presents the theoretical background, covering essential concepts regarding knowledge graphs, graph neural networks, the link prediction task, and relevant evaluation metrics.

Chapter 3 details the methodology employed, including the description of the dataset, the preprocessing steps, the construction of the GNN models (both the initial heterogeneous approaches and the refined homogeneous ones), and the experimental protocol for training and evaluation.

Chapter 4 presents and discusses the results obtained from the various experiments, comparing the different architectures and hyperparameter configurations.

Finally, Chapter 5 provides the conclusion and future work, summarizing the main findings, discussing the limitations of the study, and proposing directions for future research in the area.

2 THEORETICAL FOUNDATION

2.1 Introduction

Machine Learning Machine Learning (ML) has emerged as a powerful tool in the solving of complex biomedical problems by enabling data-driven decision making in tasks such as disease diagnosis, drug discovery, and personalized medicine. Given the rapid expansion of biomedical data, ML techniques provide innovative approaches to analyze structured and unstructured data, improve predictive modeling, and optimize treatment strategies in healthcare applications.

This study explores various ML techniques applied in biomedicine, including Deep Neural Networks, Graph Neural Networks, and Graph Attention Networks Graph Attention Network (GAT). GNNs, in particular, offer a compelling approach to modeling complex relationships between biomedical entities, such as gene-disease associations, by capturing interdependencies in structured data.

A critical aspect of ML in biomedical research is the availability of high-quality biomedical databases. Structured datasets, including graph-based repositories like the *Stanford Biomedical Network Dataset Collection* (ZITNIK *et al.*, 2018), enable precise modeling of biomolecular interactions. This study reviews key biomedical graph datasets, their applications, and techniques such as feature augmentation, which enhance the predictive capabilities of ML models.

Finally, evaluating the performance of classification models is essential in biomedical applications. Standard evaluation metrics such as Accuracy, F1-Score, and Receiver Operating Characteristic Area Under Curve AUC-ROC provide insights into the reliability and generalizability of ML models.

2.2 Bibliographic Review

The data era has emerged as a result of the rapid generation of vast amounts of data across multiple fields, from biology and medicine to social networks and finance. As data grows exponentially, traditional computational methods struggle to process and analyze these large volumes of information. Machine Learning has become a cornerstone for data-driven analysis, enabling the extraction of valuable insights from complex datasets. ML algorithms, especially

those based on deep learning and neural networks, have garnered significant attention due to their effectiveness in solving intricate problems, from image recognition to natural language processing. In biomedicine, ML has had a transformative impact, providing novel approaches for disease diagnosis, drug discovery, and personalized medicine.

In particular, ML has revolutionized the understanding of complex biological systems. A prime example of this is the development of AlphaFold2 (YANG *et al.*, 2023), an artificial intelligence system developed by DeepMind that has made groundbreaking advances in the field of protein structure prediction. AlphaFold2 highlights the immense potential of ML techniques in addressing biological challenges, underscoring the importance of their application in solving complex problems in biology.

One of the most effective ways to tackle complex biological challenges using ML is through the representation of biological data as graphs. Biological networks, such as proteinprotein interaction networks, gene-disease association networks, and metabolic networks, are inherently suitable for graph-based representations. In these networks, nodes represent biological entities such as genes, proteins, or diseases, while edges signify the relationships or interactions between them. This approach allows for the modeling of intricate interdependencies and provides a powerful framework for uncovering hidden patterns and relationships within biological data.

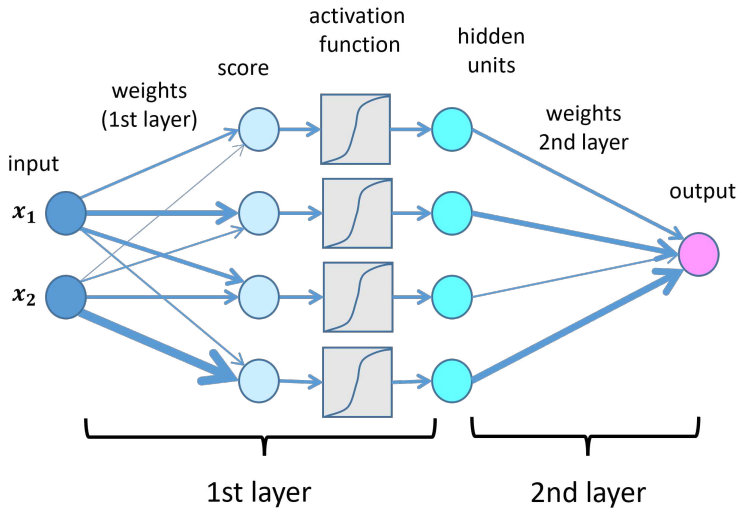
For example, the PhenoPred algorithm (RADIVOJAC *et al.*, 2008) is designed to detect gene-disease associations by analyzing protein-protein interaction networks and other molecular-level biological data. Using supervised learning and incorporating various biological features, such as protein sequence, function, and physicochemical properties, demonstrates how graph-based methods combined with ML techniques can offer significant insights into complex biomedical problems.

2.3 Machine Learning Techniques

Machine Learning has become an essential part of modern data analysis, enabling systems to learn from data and make predictions or decisions without explicit programming. This section provides an introduction to key ML concepts, including Neural Networks, Attention Mechanisms, and Graph Neural Networks, all of which are widely applied in various domains like computer vision, natural language processing, and biomedicine.

2.3.1 Neural Networks

Neural networks are a class of ML algorithms inspired by the structure of the human brain. They consist of layers of interconnected nodes, called neurons, which work together to process data. As shown in figure 2, the neural network typically contains three types of layers: the input layer, the hidden layers, and the output layer.



Source: Image from (PAASS, April 21, 2021)

Figure 2 – A simplified architecture of a Multi-Layer Perceptron (MLP), similar to the model used as the decoder in this work. It illustrates how input features are transformed through weighted layers and non-linear activation functions to produce a final prediction.

Each neuron in a neural network takes an input, applies a weight to it, passes it through an activation function, and sends the output to the next layer. The purpose of training a neural network is to find the optimal weights that minimize the error between the predicted output and the actual output, often using algorithms such as back-propagation.

Mathematically, a simple neural network can be represented as:

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{2.1}$$

where:

- y is the output of the neuron.

- w_i are the weights associated with the inputs.

- x_i are the inputs.

- b is the bias term.

- f is the activation function, such as ReLU or sigmoid.

Neural networks are powerful for modeling complex relationships in data and are the foundation for deep learning, a subset of ML focused on learning from large amounts of data.

2.3.2 Attention Mechanism

The attention mechanism is a powerful technique that allows models to dynamically focus on specific parts of the input data by assigning varying weights to different elements. This concept has become a cornerstone of modern deep learning architectures, particularly in the Transformer model, which has revolutionized natural language processing and has been successfully applied to computer vision and other domains.

In the Transformer architecture, the attention mechanism is computed as follows:

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.2}$$

Here:

- Q represents the queries,
- K represents the keys,
- V represents the values,
- and d_k is the dimensionality of the keys.

This formula computes a weighted sum of the values V, where the weights are derived from the scaled dot-product of the queries and keys. The scaling factor $\sqrt{d_k}$ helps to prevent the dot product values from becoming too large, which in turn stabilizes the softmax function.

The attention mechanism has a broad range of applications in machine learning. In natural language processing, it underpins tasks such as translation, text summarization, and language modeling. Beyond text, attention has also been successfully applied in computer vision. For instance, the Vision Transformer Vision Transformer (ViT) model by Dosovitskiy et al. (DOSOVITSKIY *et al.*, 2020) leverages this mechanism to process image data, achieving state-of-the-art results in image classification tasks.

Another notable application of the attention mechanism in Graph Neural Networks can be observed in models such as Graph Attention Networks. In these architectures, attention is used to learn varying importance (or weights) for each neighbor in a graph, allowing the model to focus on the most relevant connections when aggregating information. Traditional GAT architectures employ a masked self-attentional layer to compute a hidden representation of each node in a graph, leveraging attention coefficients between pairs of connected nodes.

2.3.3 Large Language Models

Large Language Models LLM are advanced neural network architectures designed to process and generate human language. Built on the Transformer architecture and employing sophisticated attention mechanisms, LLMs such as GPT-4, GEMINI and others are capable of capturing long-range dependencies and contextual nuances in text. They excel in a wide array of natural language processing tasks, including text summarization, translation, sentiment analysis, and question answering.

LLMs work by learning patterns and representations from extensive and diverse textual datasets (RADFORD; NARASIMHAN, 2018). Their ability to understand context allows them to generate coherent and contextually appropriate responses, making them highly valuable in various domains.

In the biomedical field, LLMs can be particularly transformative. As LLMs integrate multi-knowledge information from diverse sources, they can be leveraged to process and enrich medical data. Specifically, they can:

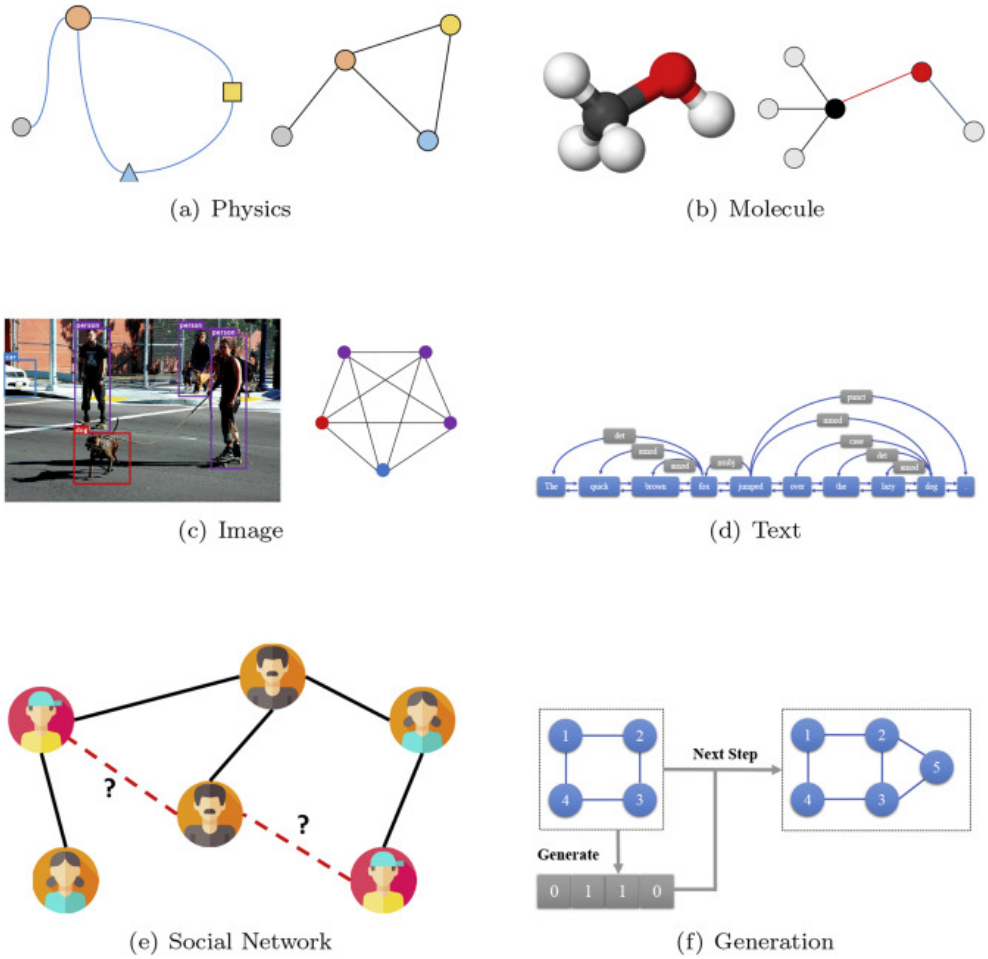
- Enhance Biomedical Datasets by generating and synthesizing textual information, LLMs can augment biomedical datasets with detailed descriptions, annotations, and literature-derived insights.
- Facilitate Data Mining effectively extracting relevant information from vast amounts of biomedical literature, enabling automated literature reviews and knowledge synthesis.

In summary, as LLMs integrate multi-knowledge information, they serve as powerful tools for leveraging medical data and enhancing biomedical-related datasets with rich, text-based information, ultimately driving advancements in healthcare research and practice.

2.3.4 Graph Neural Networks

Graph Neural Networks represent a cutting-edge class of deep learning models designed to perform inference on data structured as graphs. Unlike traditional neural networks, GNNs can capture the complex relationships and interdependencies between nodes (e.g., genes, diseases) within a graph, making them particularly suited for analyzing biological networks. By leveraging node features and edge information, GNNs can learn to predict not only the properties of individual nodes but also the strength and significance of the connections between them. This capability is critical for the work, as it allows us to model intricate gene-disease associations within the Knowledge Graph, potentially uncovering novel insights and predictive markers for diseases.

Graphs are widely used in many fields due to their ability to represent complex systems, as illustrated in Figure 3. For example, in physics, molecules, and images, graphs help represent physical systems, molecular structures, and image recognition tasks, respectively. In the case of biological networks, nodes represent genes, proteins, or diseases, and edges represent interactions or associations. This approach has proven highly effective in areas such as drug discovery, gene-disease association prediction, and more.



Source: Image from (ZHOU *et al.*, 2020)

Figure 3 – Examples of graph structures across various domains. This work focuses on biomedical networks, which are conceptually similar to molecular graphs (b), where nodes represent biological entities and edges represent their complex interactions.

2.3.4.1 The Message-Passing Framework

The core mechanism enabling GNNs to learn from graph structures is the message-passing paradigm. In this framework, each node iteratively updates its feature vector (or "embedding") by aggregating information from its immediate neighbors. This process, repeated over several layers, allows a node's representation to be influenced by nodes that are multiple "hops" away in the graph, effectively capturing the local topological context.

A single layer of a GNN can be formally described by a two-step process: aggregation and update. A simple yet powerful formulation, inspired by Graph Convolutional Networks (GCNs), illustrates this behavior. The feature vector $\mathbf{h}_v^{(l+1)}$ of a node v at layer $l+1$ is computed as follows:

$$\mathbf{h}_v^{(l+1)} = \sigma\left(\mathbf{W}^{(l)} \sum_{u \in \mathcal{N}(v) \cup \{v\}} \frac{1}{\sqrt{\deg(v)\deg(u)}} \mathbf{h}_u^{(l)}\right) \tag{2.3}$$

Where:

- $\mathbf{h}_v^{(l)} \in \mathbb{R}^{D_l}$ is the feature vector of node v at layer l.
- $\mathcal{N}(v)$ is the set of neighbors of node v.
- $\deg(v)$ is the degree of node v.
- $\mathbf{W}^{(l)} \in \mathbb{R}^{D_{l+1} \times D_l}$ is a learnable, layer-specific weight matrix that transforms the aggregated features.
- σ is a non-linear activation function, such as ReLU.

In essence, Equation 2.3 states that a node's updated representation is derived by taking a normalized, weighted sum of the feature vectors of its neighbors (and itself), followed by a linear transformation and a non-linear activation. This process allows the model to learn how to best combine neighborhood information to make meaningful predictions.

In graph-based machine learning, various tasks enable the extraction of meaningful insights from network-structured data. Depending on the focus of the analysis, these tasks can target individual nodes, entire graphs, or the relationships (edges) between nodes. The following subsections dive into three fundamental tasks:

2.3.4.2 Node Classification

Node classification is a fundamental task in GNNs where the goal is to predict the labels or categories of individual nodes within a graph. For example, in biological networks, this could involve predicting the function of a protein (node) based on its interactions with other proteins. The model learns to classify nodes by analyzing the patterns and structures in the graph and using features associated with each node.

2.3.4.3 Graph Classification

Graph classification involves predicting the class or label of an entire graph rather than individual nodes. This is particularly useful in tasks where each graph represents a unique structure, such as classifying molecular structures or biological networks. The model learns to understand the global structure of a graph and classifies it accordingly.

2.3.4.4 Link Prediction

Link prediction is a task in GNNs where the model predicts the likelihood of an edge (relationship) forming between two nodes in a graph. This is especially valuable in biological applications, such as predicting potential gene-disease associations or drug-target interactions. The model learns the underlying patterns in the graph to identify missing edges or to suggest new connections based on existing data.

This study focuses on link prediction, a task for which GNN models are typically framed within an encoder-decoder structure. The encoder, which is a GNN, is responsible for processing the graph topology and node features to generate rich, informative embeddings for each node. The decoder then takes these node embeddings as input to score and predict the likelihood of a link existing between a pair of nodes. The performance of the overall model critically depends on the choice and design of both of these components.

2.3.4.5 Critical Analysis of GNN Encoders

GNN-based encoders offer significant advantages over classical link prediction methods. Traditional approaches, such as those based on node similarity (e.g., Common Neighbors, Jaccard Index (LIBEN-NOWELL; KLEINBERG, 2007)) or matrix factorization, often rely on structural heuristics and struggle to incorporate rich node features. In contrast, GNNs learn complex, non-linear relationships directly from the data, effectively integrating both graph topology and node attributes. This is particularly powerful in the biomedical context, where node features (e.g., gene expression, protein functions) are critical for understanding interactions. However, GNNs are not without their challenges. They can be computationally expensive on large graphs and are susceptible to issues like oversmoothing, where node representations become indistinguishable after many layers of message passing (LI *et al.*, 2018). Furthermore, their performance is sensitive to the quality of the input graph, as noisy or incomplete connections can negatively impact the learned embeddings.

2.3.4.6 Choice of Decoder

While the GNN encoder generates the node embeddings, the decoder determines how they are used to score potential links. A common and flexible choice, used in this work, is a Multi-Layer Perceptron (MLP). The MLP takes the embeddings of a node pair (e.g., their

element-wise product) and learns a non-linear function to classify whether a link exists. While effective, alternative decoders exist, particularly from knowledge graph embedding literature. For instance, DistMult (YANG *et al.*, 2015) models relationships as a simple bilinear operation. However, the MLP was chosen for its generality and capacity to learn more complex decision boundaries, which is advantageous when dealing with the rich feature representations derived from biomedical data.

2.3.4.7 Architectures Explored

Given this framework, the specific GNN encoder architectures explored in this work are detailed in the following sections.

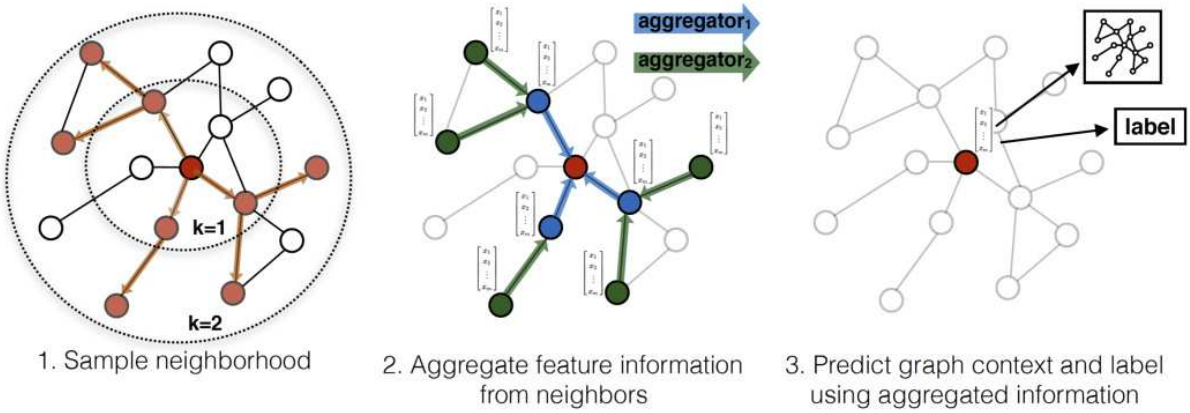
2.3.4.8 GATv2Conv

Graph Attention Networks (BRODY *et al.*, 2022) are among the most popular GNN architectures for representation learning on graphs. However, traditional GATs compute what is known as static attention, where the ranking of the attention scores is unconditioned on the query node. This limitation can prevent the model from fitting certain graph problems effectively. GATv2Conv addresses this issue by introducing a dynamic attention mechanismone that is strictly more expressive than its predecessor. By modifying the order of operations, GATv2Conv allows nodes to attend to their neighbors in a more flexible and context-aware manner. Extensive evaluations have demonstrated that GATv2Conv outperforms the original GAT on a variety of benchmarks, all while maintaining similar parametric costs. GATv2Conv is implemented in several libraries including PyTorch Geometric and Deep Graph Library.

2.3.4.9 SAGEConv

SAGEConv (L. *et al.*, 2018), introduced as part of the GraphSAGE framework, is an inductive method for generating low-dimensional node embeddings. Unlike traditional transductive approaches that require all nodes to be present during training, SAGEConv learns a function to generate embeddings by sampling and aggregating features from a node's local neighborhood. This inductive framework leverages node features such as text attributes and scales effectively to large graphs. It has been demonstrated to outperform strong baselines on multiple inductive node-classification benchmarks, including tasks involving evolving information

graphs (e.g., citation networks, Reddit posts) and multi-graph datasets such as protein-protein interaction networks.



Source: Image from (L. *et al.*, 2018)
Figure 4 – Visual illustration of the GraphSAGE sample and aggregate approach.

2.3.4.10 GCNConv

GCNConv (KIPF; WELLING, 2017) represents a scalable approach to semi-supervised learning on graphs by employing a localized, first-order approximation of spectral graph convolutions. This method aggregates information from neighboring nodes through a normalized summation, which captures both local graph structure and node features. GCNConv is computationally efficient, scaling linearly with the number of edges in the graph. Experiments on citation networks and knowledge graph datasets have shown that GCNConv consistently outperforms several related methods, making it a reliable choice for tasks that require efficient and effective representation learning on large-scale graph-structured data.

2.4 Dataset

In this section, will be described the dataset used and outline the procedures employed to prepare the data for analysis, including preprocessing steps and feature augmentation techniques.

2.4.1 Data Preprocessing

Data preprocessing is a critical step in the machine learning pipeline, as it ensures that the data is clean, consistent, and suitable for model training. On the study, the preprocessing

phase involves the following key steps:

- Data Cleaning, removing or correcting erroneous, missing, or inconsistent entries to improve data quality.
- Normalization, scaling numerical features to a common range, which helps in improving the convergence of learning algorithms.
- Encoding Categorical Variables, converting categorical data into a numerical format using techniques such as one-hot encoding or label encoding.
- Data Splitting, dividing the dataset into training, validation, and test sets to ensure that the model can be properly evaluated and generalized.

These preprocessing steps are essential for reducing noise, mitigating potential biases, and ensuring that the input data is in a format conducive to effective learning by the models.

2.4.2 Feature Augmentation

Feature augmentation is a critical step for enriching the dataset with additional, meaningful information. In this work, textual descriptions for genes and diseases were incorporated by leveraging a LLM as a text generation source. Specifically, GEMINI was used to produce succinct, informative summaries of gene functions and disease characteristics.

After obtaining these textual descriptions, the unstructured text was converted into numerical representations. To accomplish this, a Transformer-based architecture capable of generating high-dimensional embeddings from textual input was employed. By concatenating these text-derived embeddings with existing structured features (e.g., gene expression levels, disease annotations), a more comprehensive input feature set was created.

The choice of the Transformer-based model for generating these embeddings is a critical decision, as several domain-specific language models have been developed for the biomedical field. Among the most prominent are:

- PubMedBERT: A BERT model pre-trained from scratch on a vast corpus of biomedical literature from PubMed (GU *et al.*, 2020). Its specialization makes it highly proficient at understanding the vocabulary and nuances of scientific texts, making it ideal for tasks involving research articles.
- Bio-ClinicalBERT: This model builds upon a base pre-trained on biomedical literature and is further trained on a large dataset of clinical notes (MIMIC-III) (ALSENTZER *et*

al., 2019). This dual exposure provides it with an understanding of both foundational biomedical science and its application in clinical practice.

For this work, an embedding model based on the principles of Bio-ClinicalBERT was selected. The rationale for this choice is its unique ability to bridge the gap between biological research (genes, proteins) and clinical outcomes (diseases, phenotypes). Since the goal is to predict gene-disease associations, a model fluent in both the language of scientific research and clinical descriptions is hypothesized to produce more informative and contextually relevant embeddings.

This augmented representation captures both the quantitative measurements inherent in the dataset and the contextual or semantic nuances contained within the text. As a result, the machine learning models have access to richer information, potentially improving predictive performance and offering deeper insights into gene-disease relationships.

2.5 Evaluation Metrics for Classification Models

In the development of models for biomedical applications, it is crucial that their performance be assessed using appropriate metrics. In a typical machine learning classification task, labels or categories are assigned to objects or instances based on their features or attributes. After the dataset is prepared (separating the target column from the input features), the model is trained and tested to determine whether its predictions match the actual labels. The following subsections describe common metrics used to evaluate classification models.

2.5.1 Accuracy

Accuracy measures the proportion of correctly classified instances (both positives and negatives) out of the total number of predictions. It is computed as:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{2.4}$$

Although accuracy is straightforward to interpret, it can be misleading in cases of highly imbalanced datasets, where one class significantly outnumbers the other.

2.5.2 Precision at K P@10

Precision at K P@10 is a ranking evaluation metric that measures the proportion of relevant items found in the top-K recommended items. This metric is particularly useful in scenarios where a user is presented with a ranked list of predictions, such as in recommendation systems or information retrieval tasks (BEEL *et al.*, 2016). Unlike binary classification metrics that evaluate a single prediction threshold, P@K assesses the quality of the top part of the ranked list.

The formula for P@K is given by:

$$\text{P@K} = \frac{\text{Number of relevant items in the top-K}}{\text{K}} \tag{2.5}$$

In the context of this study, where the model predicts potential gene-disease associations, P@K evaluates how many of the top K predictions are actual, known associations. For this work, Precision at 10 P@10 will be used to assess the model's performance, focusing on the ten most likely associations it identifies. A high P@10 score would indicate that the model is effective at placing true gene-disease links at the top of its recommendations.

2.5.3 F1-Score

The F1-Score is the harmonic mean of Precision and Recall, providing a balance between the two metrics. This measure is particularly useful when the dataset is imbalanced and a single metric that captures both Precision and Recall is needed. It is defined as:

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.6}$$

where

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.7}$$

A high F1-Score indicates that the model achieves both high Precision and high Recall, making it suitable for tasks where a balance between identifying all positive instances and minimizing false positives is critical.

2.5.4 ROC-AUC

The Receiver Operating Characteristic curve plots the True Positive Rate True Positive Rate (TPR) against the False Positive Rate False Positive Rate (FPR) at various classification thresholds. The Area Under the Curve summarizes the ROC curve in a single value, typically ranging from 0.5 (random performance) to 1.0 (perfect classification). The True Positive Rate and False Positive Rate are defined as:

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \quad \text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{2.8}$$

The ROC-AUC metric provides a measure of how well the model distinguishes between the two classes across all possible classification thresholds.

2.5.5 PR-AUC

While ROC-AUC is widely used for binary classification, it can present an overly optimistic view in the presence of class imbalancecommon in many biomedical tasks where positive samples (e.g., true disease associations) may be far outnumbered by negative samples. In such scenarios, the Precision-Recall (PR) curve often provides a more informative assessment of model performance.

The PR curve is a plot of Precision versus Recall across varying classification thresholds. The PR-AUC summarizes the area under this curve, typically ranging from 0.0 to 1.0. Higher values indicate superior performance, as the model maintains both high Precision and high Recall across different thresholds.

$$\text{PR-AUC} = \int \text{Precision}(\text{Recall})\, d(\text{Recall}) \tag{2.9}$$

In highly imbalanced datasetswhich often arise in biomedical contexts, such as detecting rare diseases or identifying novel gene-disease linksa model with a high PR-AUC is particularly desirable. It implies that the model reliably identifies the true positives (high Recall), while also minimizing false positives (high Precision), despite the scarcity of positive examples.

2.6 Chapter Conclusion

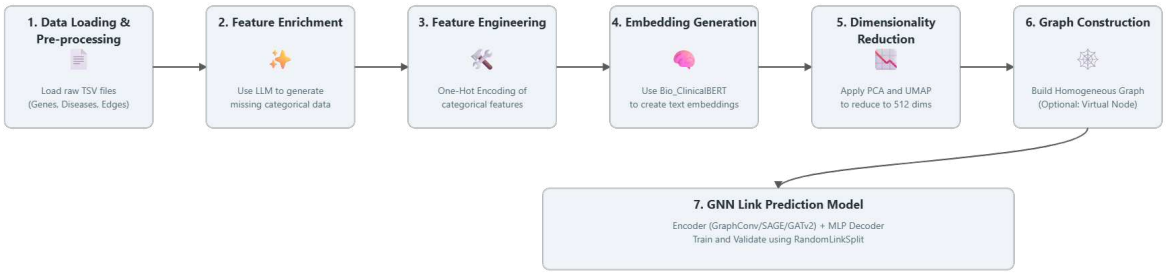
In this chapter, the theoretical ground that guide the study have been presented. The significance of Machine Learning and its profound impact on data-rich fields such as biomedicine were first reviewed. Various neural network architectures, including Graph Neural Networks and Transformer-based models, were then explored, highlighting how these techniques can excel in tasks such as link prediction for gene-disease associations.

Additionally, the importance of data preprocessing and feature augmentation was discussed, where the structured dataset was enriched using GEMINI-generated textual descriptions for genes and diseases. These text embeddings, combined with structured features, provide a richer representation that can potentially improve predictive performance. Finally, key evaluation metrics such as Accuracy, F1-Score, ROC-AUC, and the more specialized PR-AUC were introduced, emphasizing how each metric may serve different scenarios and data distributions.

3 METHODOLOGY

3.1 Introduction

This chapter details the methodology employed to construct and evaluate a Graph Neural Network for predicting gene-disease associations. The process begins with the acquisition and initial exploration of various biomedical datasets. Subsequently, a comprehensive data preprocessing and feature engineering pipeline is implemented to create a unified and enriched dataset suitable for machine learning. This is followed by the generation of high-dimensional embeddings for genes and diseases using a specialized language model. To make these embeddings more efficient for the GNN, dimensionality reduction techniques are applied. The chapter then describes the architecture of the GNN, including the use of a virtual node to enhance model performance. Finally, the training process, including hyperparameter optimization and the final model evaluation, is presented. The overall workflow of this methodology is illustrated in Figure 5.



Source: Image from the author (2025)
Figure 5 – Simplified illustration of the data pipeline.

3.2 Data Acquisition and Initial Exploration

The foundation of this project is a collection of publicly available datasets from the BioSNAP collection (ZITNIK *et al.*, 2018), which provide information on genes, diseases, and their associations. The following datasets, which constitute the nodes and edges of our graph, were used:

- Disease Features (D-DoMiner_miner-diseaseDOID.tsv & D-MeshMiner_miner-disease.tsv): These files provide detailed descriptions and features of diseases, derived from the Dis-

ease Ontology and the Comparative Toxicogenomics Database. They include disease names, definitions, and synonyms, forming the primary source of feature information for the disease nodes.

- Disease Classification (D-DoPathways_diseaseclasses.csv): This dataset offers a mapping of diseases to their respective categories based on etiology and anatomical location (e.g., cancers, musculoskeletal system diseases). This provides crucial categorical features for the disease nodes.
- Gene Features (G-SynMiner_miner-geneHUGO.tsv): Sourced from the HUGO Gene Nomenclature Committee HUGO Gene Nomenclature Committee (HGNC), this dataset contains information for over 35,000 human genes. It includes names, descriptions, synonyms, and chromosomal locations, serving as the feature source for the gene nodes.
- Gene-Disease Associations (DG-AssocMiner_miner-disease-gene.tsv): This file contains the known associations between genes and diseases, which are used to construct the edges of our graph. It represents curated, high-quality interaction data essential for training and evaluating the link prediction model.

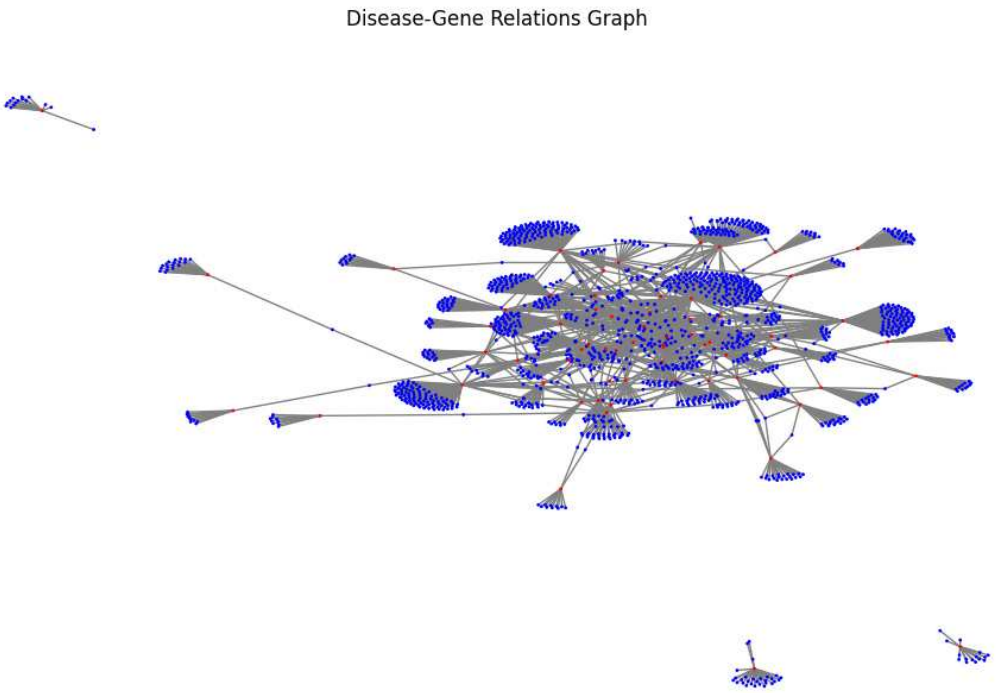
After loading and integrating the data, the graph's fundamental topological properties were analyzed. The key statistics, summarized in Table 2, characterize the overall structure of the network. The graph consists of 7,813 nodes and 21,357 edges. Statistical analysis reveals that all nodes belong to a single large connected component, meaning there is a path between any two nodes in the network. The finite diameter of 8 and a 90-percentile effective diameter of approximately 5.44 further describe a network where nodes are, on average, relatively close to one another.

Table 1 – Topological properties of the constructed gene-disease graph.

Metric	**Value**
Nodes	7,813
- Disease Nodes	519
- Gene Nodes	7,294
Edges	21,357
Nodes in Largest Connected Component	7,813
Fraction of Nodes in Largest C.C.	1.00
Diameter (Longest Shortest Path)	8
90-Percentile Effective Diameter	5.44

Source: From the BioSNAP website (2025).

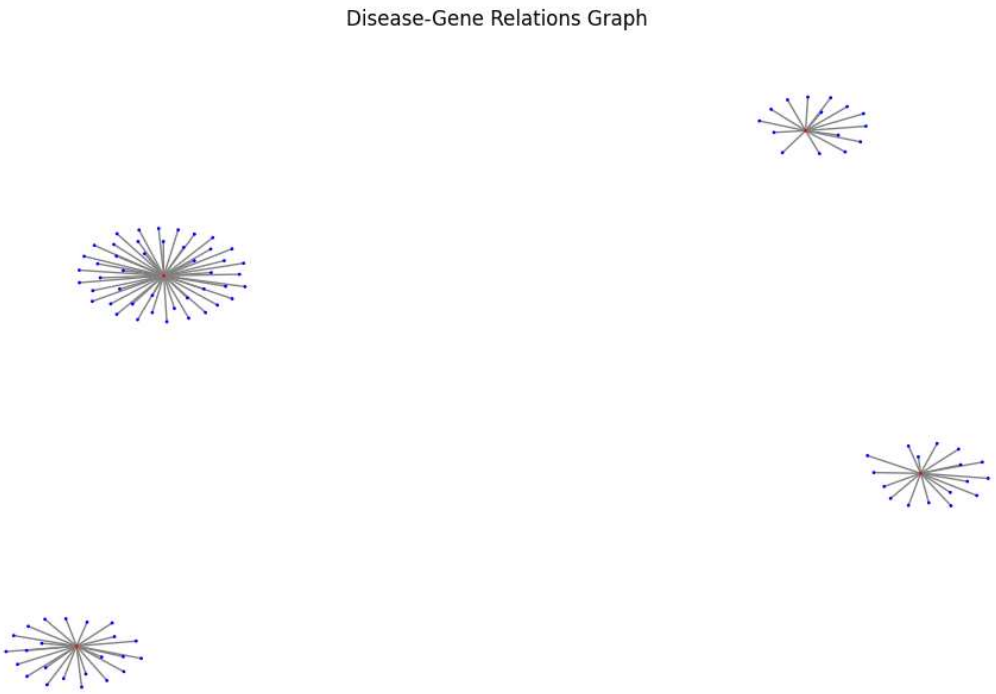
An initial visualization of the graph structure provides critical insights into its topology. Figure 6 displays the entire gene-disease network, where red nodes represent diseases and blue nodes represent genes. The visualization reveals a large, central component where most nodes are interconnected, alongside several smaller, isolated components. This structure suggests that while a significant portion of the data is related, there are also distinct clusters of information.



Source: Image from the author (2025)

Figure 6 – Visual illustration of the entire gene-disease graph. The visualization reveals a large, central component alongside several smaller, disconnected components, highlighting the network's fragmented nature which poses a challenge for GNNs.

To better understand the local connectivity and potential challenges, Figure 7 shows a subgraph constructed from only the first 100 links in the dataset. This zoomed-in view clearly illustrates the graph's sparsity. We can observe several disconnected subgraphs, each forming a star-like structure around a central disease node. This sparsity poses a challenge for GNNs, as information cannot propagate between these disconnected components during the message-passing phase. This observation motivates the exploration of techniques like adding a virtual node, which aims to create paths between otherwise isolated parts of the graph, thereby improving the model's ability to learn global patterns.



Source: Image from the author (2025)
Figure 7 – A zoomed-in view of 100 links from the dataset, clearly illustrating the graph's local sparsity and the formation of star-like structures around central disease nodes. This motivates the need for techniques to handle disconnected components.

3.3 Data Preprocessing and Feature Engineering

A multi-step preprocessing pipeline was implemented to clean, integrate, and enrich the raw data. This process was essential to create a high-quality, unified dataset for the subsequent stages of the project.

3.3.1 Disease Data Consolidation

A crucial step in preparing the data was the creation of a comprehensive and clean feature table for the disease nodes. This process involved several stages of merging, consolidation, and data enrichment.

- Merging Disparate Datasets: The process began by identifying the unique diseases present in the gene-disease association file (`DG-AssocMiner`). This list of diseases served as the base for the feature table. This base table was then sequentially merged with the three other disease-related datasets (`D-DoPathways`, `D-MeshMiner`, and `D-DoMiner`) using the disease name and various identifiers (e.g., 'Disease ID') as keys. This step aggregated all

available information for each disease into a single DataFrame.

- Consolidating Definitions: The merged dataset contained two separate columns for disease definitions (‘Definition‘ and ‘Definitions‘) originating from different source files. Many entries had a definition in one column but not the other. To create a single, unified feature, these columns were consolidated; where one definition was missing, the value from the other was copied over, ensuring maximum completeness.
- LLM-Powered Feature Enrichment: After consolidation, a significant number of diseases still lacked key categorical features, such as ‘Disease Class‘, or had no definition at all. To address these gaps, we employed Google’s Gemini Pro model via its API. For each disease with missing information, a dynamic prompt was constructed to generate the missing values. The model was tasked with inferring the ‘Disease Class‘ and the ‘main_system_affected‘ for all diseases, and to generate a ‘definition‘ if one was not present. This automated enrichment process significantly enhanced the feature set’s quality and consistency. The structure of the dynamic prompt used is detailed in Appendix A.

3.3.2 Gene Data Processing

The gene dataset, initially containing over 35,000 entries, required significant preprocessing to structure its features for the model. The process included the following key steps:

- Identifier Normalization: The original `hgnc_id` column, containing alphanumeric identifiers (e.g., ‘HGNC:5‘), was processed using a regular expression to extract only the numerical part. This resulted in a clean, integer-based `id` column that could be used for merging and mapping, after which the original `hgnc_id` column was dropped.
- Chromosomal Location Parsing: The `location` column, which contains complex strings describing chromosomal positions (e.g., ‘19q13.43‘), was parsed using regular expressions. This operation decomposed the single string into multiple, more granular features, creating separate columns for the start and end chromosome, arm (‘p‘ or ‘q‘), and specific sub-locations. This transformed a single text feature into several more informative categorical and numerical features.
- Dataset Filtering: To ensure relevance and reduce computational overhead, the primary gene dataset was filtered to include only the 3,523 genes that explicitly appear in the gene-disease association file (`DG-AssocMiner_miner-disease-gene.tsv`). This criti-

cal step aligns the node feature set directly with the graph structure being analyzed, removing any genes not involved in the known interactions.

3.3.3 Feature Engineering

With the raw data cleaned and structured, feature engineering was performed to create a suitable numerical input for the GNN. This involved handling both categorical and textual data across the gene and disease datasets.

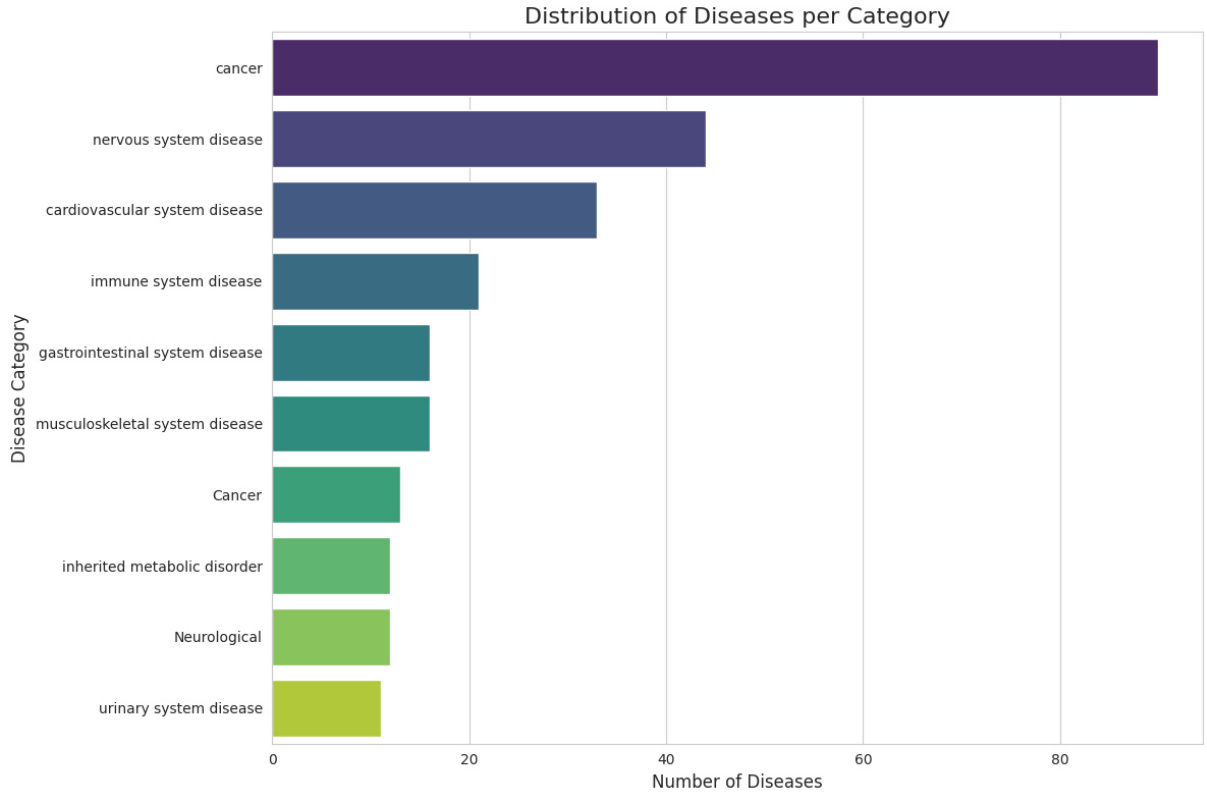
- Column Pruning: In both datasets, numerous columns were identified as being redundant, having an excessive number of missing values, or containing information not directly relevant to the link prediction task (e.g., synonymous names, alternative IDs like `entrez_id`). These columns were dropped to simplify the feature space. For instance, in the final merged disease table, columns such as `Name_x`, `Name_y`, `# MESH_ID`, and `Synonyms` were removed.
- Categorical Feature Handling: Key categorical features were selected for one-hot encoding to convert them into a machine-readable format. For disease nodes, these included `Disease Class` and `main_system_affected`. For gene nodes, the features were `locus_group` and `locus_type`. Before encoding, the `CategoricalImputer` from the 'feature-engine' library was used to fill any missing values with the string "Missing". Subsequently, the `OneHotEncoder` from the same library was applied, transforming these categorical variables into a binary vector representation.

3.3.4 Exploratory Data Analysis of the Processed Graph

After the initial data cleaning, integration, and filtering (detailed in Section 3.3), a detailed exploratory analysis was conducted on the final graph used for modeling. This analysis provided critical insights into the characteristics of the nodes, features, and the overall graph topology.

The final dataset contains 3,523 unique genes and 519 unique diseases. An analysis of the gene feature completeness revealed that core features like `name` and `locus_type` were 100% complete. However, the `gene_family` feature was only 66.28% complete, highlighting a potential area for future data enrichment. For the diseases, all 519 unique entities were successfully mapped to one of 132 unique disease categories, with no missing values. As shown in Figure 8, the class distribution is imbalanced, with categories like "cancer" and "nervous system

disease" being highly represented.



Source: Elaborated by the author (2025).
Figure 8 – Distribution of the top 20 most represented disease classes in the dataset.

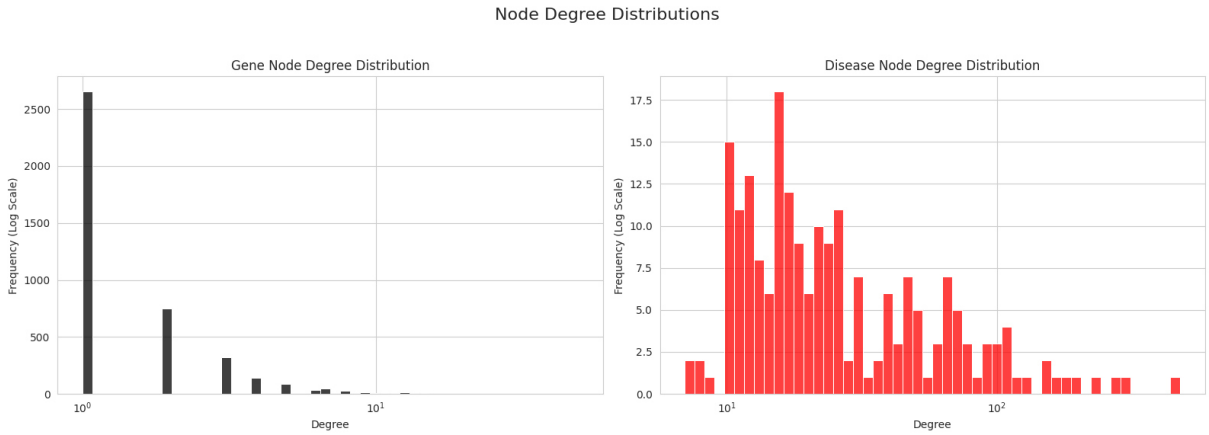
The topological properties of the final graph are summarized in Table 2. The graph is highly sparse, with a density of only 0.000868, which is characteristic of real-world biological networks. This sparsity is further evidenced by the component analysis, which identified 1642 disconnected components. This high degree of fragmentation poses a significant challenge for GNNs, as information cannot propagate between these components. This finding strongly motivates the use of techniques like a virtual node to create paths for global information flow.

Table 2 – Topological properties of the final constructed gene-disease graph.

Metric	Value
Total Number of Nodes	4,354
- Unique Genes	3,523
- Unique Diseases	519
Total Number of Edges	8,221
Graph Density	0.000868
Number of Connected Components	1,642
Size of Largest Component	4,354 nodes
Orphan Genes (Degree 0)	1,329
Orphan Diseases (Degree 0)	312

Source: Elaborated by the author (2025).

The node degree distributions, shown in Figure 9, reveal more about the graph's structure. The gene degree distribution exhibits a long-tail pattern, with most genes having very few connections (average degree of 1.98), while a few genes act as hubs with many connections (max degree of 49). In contrast, the disease nodes are much more densely connected, with an average degree of 39.71 and a maximum of 478. This indicates that diseases in this dataset are associated with many genes, forming star-like subgraphs, a pattern consistent with the visual exploration of the graph's local structure. The presence of 1,329 orphan genes and 312 orphan diseases further underscores the data's sparsity and the challenge of learning representations for these isolated nodes.



Source: Elaborated by the author (2025).

Figure 9 – Log-log degree distributions highlighting the network's structure. The plots reveal the characteristic long-tail pattern for gene connectivity versus the denser, more centralized connectivity for disease nodes.

3.4 Generating Embeddings

To represent the complex information contained in the gene and disease data in a way that is suitable for a GNN, we generated high-dimensional embeddings using a specialized language model.

3.4.1 Formal Representation of the Feature Generation Pipeline

The process of converting raw textual and categorical data into dense, low-dimensional feature vectors for the GNN can be formally represented as a multi-stage pipeline. This section formally defines each transformation and justifies the methodological choices.

3.4.1.1 LLM-Powered Feature Enrichment and Justification

For disease nodes lacking categorical or descriptive information, an enrichment step was performed. Let n_d be a disease node with an initial, incomplete feature set $F_{initial}(n_d)$. This process can be modeled as a function f_{LLM} that operates on a dynamically generated prompt P:

$$F_{enriched}(n_d) = f_{LLM}(P(F_{initial}(n_d))) \tag{3.1}$$

Here, f_{LLM} represents the Gemini Pro model, which infers missing values to produce a complete feature set, $F_{enriched}(n_d)$, as detailed in Appendix A.

The selection of Google's Gemini model for the feature enrichment task was primarily driven by practical and economic factors. Google provides a free tier for its Gemini family of models through Google AI Studio. This access tier was perfectly suited for the scope of this academic research, as it enabled the one-time data enrichment task to be completed effectively without incurring API costs.

3.4.1.2 Domain-Specific Textual Embedding

Once all nodes possessed rich textual descriptions, they were converted into high-dimensional embeddings. For any node n (gene or disease), let T_n be its concatenated textual data. This transformation is represented by the function f_{BERT}:

$$\vec{e_n} = f_{BERT}(T_n) \tag{3.2}$$

Where $\vec{e_n} \in \mathbb{R}^d$ is the resulting high-dimensional embedding vector. The function f_{BERT} corresponds to the Bio-ClinicalBERT model, and the final embedding is derived from the '[CLS]' token's hidden state representation, capturing the semantic essence of the input text.

3.4.1.3 Dimensionality Reduction and Justification

The final step in the pipeline reduces the dimensionality of the embeddings to a more computationally efficient size. This can be formalized as a function f_{reduce} that maps the high-dimensional vector $\vec{e_n}$ to a low-dimensional vector $\vec{z_n}$:

$$\vec{z_n} = f_{reduce}(\vec{e_n}) \tag{3.3}$$

Where $\vec{z_n} \in \mathbb{R}^k$ and $k \ll d$ (in this work, $k = 512$). The function f_{reduce} was implemented using two techniques:

- PCA: A linear transformation that projects the data onto a lower-dimensional subspace while maximizing the preservation of variance.
- UMAP: A non-linear manifold learning technique that preserves both the local and global topological structure of the data.

The selection of PCA and UMAP over other alternatives (e.g., t-SNE, Autoencoders) was intentional. PCA serves as an efficient and interpretable linear baseline. UMAP was chosen for its proven effectiveness in generating meaningful low-dimensional representations that retain the complex structure of the original data, a theoretically advantageous property for subsequent graph-based learning tasks.

3.4.2 Bio-ClinicalBERT

We utilized the emilyalsentzer/Bio_ClinicalBERT model (ALSENTZER *et al.*, 2019), a powerful language model pre-trained on a large corpus of biomedical and clinical text. This model is well-suited for our task as it can capture the semantic relationships between biomedical concepts.

After embedding generation, the resulting feature tensors had high dimensionality. The disease feature tensor had a shape of (519, 3072), and the gene feature tensor had a shape of (3523, 3703).

3.4.3 Dimensionality Reduction

The embeddings generated by the Encoder model are high-dimensional, which can be computationally expensive and may lead to the "curse of dimensionality." To address this, we employed dimensionality reduction techniques to create more compact and efficient representations of our data. Specifically, we used:

- Principal Component Analysis (MAKIEWICZ; RATAJCZAK, 1993) Principal Component Analysis (PCA): A linear dimensionality reduction technique that projects the data onto a lower-dimensional subspace while preserving as much of the variance as possible.
- Uniform Manifold Approximation and Projection (MCINNES *et al.*, 2020) Uniform Manifold Approximation and Projection (UMAP): A non-linear dimensionality reduction technique that is particularly effective at preserving the global structure of the data.

The RobustScaler from scikit-learn was also used to scale the features before applying PCA and UMAP. The effectiveness of this process is summarized in Table 3, which shows

that the first 512 components retained a vast majority of the variance from the original feature space for both node types, justifying the reduction to this dimension.

Table 3 – Explained variance ratio captured by the first 512 principal components for each node type.

Node Type	Explained Variance Ratio (%)
Disease	98.22
Gene	100.00

Source: Elaborated by the author (2025).

While the PCA variance analysis provides a strong quantitative basis for selecting 512 dimensions, the choice was also guided by established practices and empirical considerations. The final embedding size represents a critical trade-off between model expressiveness and computational efficiency. A higher-dimensional space allows the model to capture more complex and nuanced patterns from the feature set, but it also increases memory consumption, training time, and the risk of overfitting. Conversely, a lower-dimensional space might act as an information bottleneck, losing valuable signals.

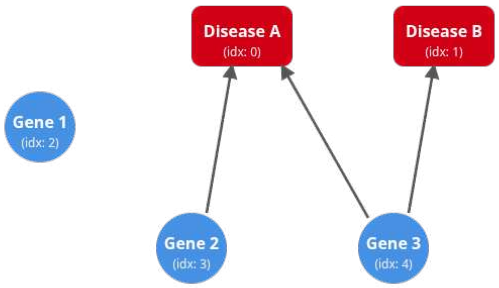
Dimensions such as 128, 256, and 512 are standard in GNN literature for tasks of this nature, having been shown to provide a good balance. Crucially, the hidden layer dimension was a key hyperparameter tuned by Optuna (as detailed in Section 3.6), with 512 emerging as the optimal choice from the search space. This provides a data-driven validation that, for this specific dataset and task, an embedding size of 512 yielded the best performing model on the validation set.

3.4.4 Graph Construction and Adjacency Representation

With the final node features prepared, the graph is constructed for input into the GNN models. A node-to-integer mapping is created to uniquely identify each gene and disease node. The relationships from the gene-disease association file are then used to build the graph's structure. Instead of a dense adjacency matrix, which would be inefficient for a sparse graph of this size, the graph's connectivity is represented using an adjacency list format, common in GNN libraries like PyTorch Geometric. This is stored as an `edge_index` tensor of shape (2, num_edges).

To illustrate this representation, consider the simple example graph in Figure 10. The nodes are first mapped to integer indices, as shown in Table 4. The directed connections

are then encoded into the `edge_index` tensor (Table 5), where each column corresponds to an edge. The top row contains the index of the source node, and the bottom row contains the index of the target node. This sparse format efficiently captures the graph's topology and is the standard input for the message-passing mechanism in GNN layers.



Source: Elaborated by the author (2025).

Figure 10 – A simple graph illustrating the relationship between diseases (red) and genes (blue), with their assigned integer indices.

Table 4 – Node-to-index mapping for the example graph shown in Figure 10.

Node Name	Assigned Index
Disease A	0
Disease B	1
Gene 1	2
Gene 2	3
Gene 3	4

Source: Elaborated by the author (2025).

Table 5 – The `edge_index` tensor representation for the graph in Figure 10. Each column represents a directed edge.

	Edge 1	Edge 2	Edge 3
Source Node Index	3	4	4
Target Node Index	0	0	1

Source: Elaborated by the author (2025).

3.5 GNN Architecture and PyTorch Geometric Details

This section details the architectural choices for the Graph Neural Network models and the specific implementation details related to the PyTorch Geometric (PyG) library (FEY; LENSSEN, 2019), which was the primary framework used for this project.

3.5.1 Framework Justification

The selection of PyG was made after considering other powerful frameworks like the Deep Graph Library (DGL) (WANG *et al.*, 2020). PyG was ultimately chosen for several key reasons: its seamless integration with the core PyTorch API, making the workflow highly intuitive for those familiar with PyTorch; its extensive and well-documented collection of state-of-the-art GNN layers and models; and its highly efficient data handling for graph structures. This combination of usability, comprehensive features, and performance made it the most suitable choice for the rapid prototyping and experimentation required in this study.

3.5.2 Graph Representation Strategies

To model the gene-disease association data, two primary graph representation strategies were explored, each with distinct advantages and implications for the GNN architecture.

- Homogeneous Graph: This is the primary representation used in this work. In this model, all nodes, regardless of whether they represent a gene or a disease, are treated as a single node type within the graph structure. The feature matrix `x` is constructed by concatenating the feature vectors of all genes and diseases into a single tensor. To retain the type information, a two-dimensional one-hot encoded feature is appended to each node's feature vector: `[1, 0]` for genes and `[0, 1]` for diseases. This approach simplifies the model design, as a single GNN architecture can operate over the entire graph without needing to handle different node and edge types explicitly.
- Heterogeneous Graph: This representation explicitly models the different types of nodes (e.g., 'gene' and 'disease') and the relationships between them (e.g., '('disease', 'actuates', 'gene')'). In PyG, this is implemented using a `HeteroData` object, which stores node features and edge indices in separate dictionaries keyed by their type. This allows the GNN to learn distinct message-passing functions for different node and edge types, potentially capturing more nuanced relational patterns. While explored, the primary focus remained on the homogeneous representation due to its strong performance and simpler implementation.

3.5.3 Virtual Node for Global Information Propagation

A common challenge in graph-based learning is the limited receptive field of GNNs; a node's updated representation is primarily influenced by its local neighborhood. In graphs with high sparsity or multiple disconnected components, this can prevent the model from learning from global patterns. To address this limitation, we experimented with the use of a virtual node (GILMER *et al.*, 2017).

Implemented in PyG via the `T.VirtualNode()` transform, this method introduces a single, new "virtual" node to the graph. This node is then connected to every other node in the graph by creating a new, distinct edge between the virtual node and every original node. During message passing, the virtual node acts as a global information aggregator. It receives information from all nodes in the graph and, in turn, broadcasts a summary of this global information back to all nodes. This creates a shortcut for information to flow between distant or disconnected parts of the graph, allowing the GNN to learn from global structural properties in addition to local neighborhood patterns.

3.5.4 Encoder-Decoder Architecture for Link Prediction

The core task of this project is link prediction, which is framed as a binary classification problem: for any pair of nodes, we predict whether a link (an association) exists between them. To solve this, we employed an encoder-decoder architecture.

- Encoder: The encoder's role is to generate meaningful, low-dimensional representations (embeddings) for each node in the graph. It consists of a stack of GNN layers that perform message passing, aggregating information from a node's neighbors to update its feature vector. This process allows the final embeddings to capture both the node's initial features and its topological context within the graph. We experimented with three different GNN convolutional layers as the backbone of our encoder: `GraphConv`, `SAGEConv`, and `GATv2Conv`.
- Decoder: The decoder takes the node embeddings generated by the encoder and uses them to predict the existence of an edge between any two nodes. For this task, a simple yet effective Multi-Layer Perceptron was used. The MLP takes as input the element-wise product of the embeddings of a source and target node pair and outputs a single raw prediction value, known as a logit. This logit represents the model's confidence in the

existence of a link. While the MLP is a powerful and flexible choice, alternative decoders, particularly from knowledge graph embedding literature, were considered. For instance, DistMult (YANG *et al.*, 2015) is a popular decoder that models relationships using a simple bilinear scoring function. However, the MLP was selected for its generality and capacity to learn more complex, non-linear decision boundaries. Given the richness of the node features in this biomedical context, the MLP's ability to learn intricate functions was deemed more advantageous than the simpler relational assumptions of models like DistMult.

3.6 Model Training and Hyperparameter Optimization

The training process is designed to optimize the parameters of the GNN model to accurately distinguish between existing and non-existing links in the graph.

3.6.1 Data Splitting and Negative Sampling

To train and evaluate the model robustly, the set of known gene-disease associations (positive edges) was split using PyG's `RandomLinkSplit` transform. This utility divides the edges into two disjoint sets: training (80%), validation (20%).

A critical aspect of training a link prediction model is providing it with examples of what does *not* constitute a link. `RandomLinkSplit` facilitates this by performing negative sampling. For each positive edge in a given set (training, validation, or test), it creates a corresponding "negative" edge (MIKOLOV *et al.*, 2013) by randomly pairing two nodes that are not connected in the original graph. This results in a balanced dataset for each split, containing an equal number of positive and negative examples, which is crucial for training a binary classifier effectively.

3.6.2 Training Details

The model was trained for a total of 500 epochs. The optimization and learning process was managed by the following components:

- Optimizer: The `Adam` optimizer (KINGMA; BA, 2017) was used. Adam is an adaptive learning rate optimization algorithm that is well-suited for training deep neural networks, as it efficiently computes individual adaptive learning rates for different parameters from

estimates of first and second moments of the gradients.

- Loss Function: We used the `binary_cross_entropy_with_logits` loss function from PyTorch. This function is ideal for binary classification tasks like link prediction. It combines a Sigmoid activation layer and a Binary Cross-Entropy loss into a single, numerically stable function. It takes the raw logits produced by the decoder and compares them against the ground truth labels (1 for positive edges, 0 for negative edges) to compute the model's error.
- Learning Rate Scheduler: To fine-tune the learning process, a `ReduceLROnPlateau` scheduler was employed. This scheduler monitors the validation loss and reduces the learning rate by a specified factor (0.5) if the loss does not improve for a "patience" of 20 epochs. This helps the model to converge more precisely to a minimum without overshooting it, especially in the later stages of training.

3.6.3 Hyperparameter Optimization with Optuna

Finding the optimal set of hyperparameters is critical for achieving the best possible model performance. We used Optuna (AKIBA *et al.*, 2019), an automatic hyperparameter optimization framework, to systematically search the parameter space. Optuna employs Bayesian optimization techniques to efficiently explore different combinations and identify those that yield the best performance on the validation set, as measured by the AUC-PR metric. The following key hyperparameters were tuned, with the search ranges specified below:

- Number of GNN Blocks: This determines the depth of the message passing and the size of the neighborhood that influences a node's final embedding. The search space consisted of integer values from 0 to 2.
- Number of MLP (Decoder) Layers: This parameter controls the depth and complexity of the decoder network, which processes the node embeddings to predict a link. The search space consisted of integer values from 0 to 2.
- Hidden layer size: This controls the dimensionality of the learned embeddings and the overall capacity of the model. The search space included the categorical values of 32, 64, and 128.
- Number of Attention Heads: For attention-based models (e.g., GATv2), this parameter determines how many independent attention mechanisms are used in parallel, allowing the model to capture different types of relationships. The search space included the cat-

egorical values of 2, 4, and 6. This parameter is not applicable to non-attention-based models.

- Dropout rate: A regularization technique used to prevent overfitting by randomly setting a fraction of neuron activations to zero during training. Optuna suggested floating-point values between 0.2 and 0.5.
- Learning rate: The step size used by the Adam optimizer to update the model's weights. A logarithmic scale between 1e-4 and 5e-3 was searched.
- Weight decay: An L2 regularization term that penalizes large weights in the model, helping to prevent overfitting and improve generalization. A logarithmic scale between 1e-5 and 5e-3 was searched.

3.6.4 Hyperparameter Sensitivity Analysis

While a full visualization of the Optuna search space can illustrate the entire optimization landscape, a direct analysis of the final, optimal hyperparameters chosen for each GNN architecture offers crucial insights into the model's sensitivity and the nature of the problem. Table 6 summarizes the best-performing hyperparameters identified by the Optuna search for the GraphConv, SAGEConv, and GATv2Conv models.

Table 6 – Optimal hyperparameters selected by Optuna for each GNN architecture on the homogeneous graph with a virtual node.

Hyperparameter	**GraphConv+MLP**	**SAGEConv+MLP**	**GATv2+MLP**
Number of GNN Layers	2	2	2
Number of MLP Layers (Decoder)	2	2	1
Dimension of Hidden Layer	128	64	64
Dropout	0.33	0.44	0.25
Learning Rate	0.0022	6.4e-04	2e-4
Weight Decay	1.2e-04	1.8e-05	2e-04
Number of Heads	N/A	N/A	6

Source: Elaborated by the author (2025).

The analysis of the optimal hyperparameters, selected by Optuna (Table 6), reveals a preference for moderately deep architectures, rather than overly simplistic models. Notably, all models converged to an encoder architecture with 2 GNN layers. This suggests that, even with the presence of a virtual node to facilitate global information flow, a 2-hop neighborhood aggregation is essential for capturing the most relevant relational patterns in the graph.

The complexity of the decoder (MLP) also showed interesting variations. The

GraphConv and SAGEConv models required a deeper decoder, with 2 MLP layers, to effectively process the generated embeddings. In contrast, the GATv2+MLP model, which has a more complex attention mechanism with 6 heads, achieved its peak performance with a simpler decoder of just 1 layer. This indicates a trade-off between the complexity of the encoder and the decoder: a more expressive encoder may require a less complex prediction head.

Other hyperparameters demonstrate the need for fine-tuning for each architecture. GraphConv benefited from a greater representation capacity, with a hidden layer of 128 dimensions, while SAGEConv and GATv2 performed better with a more compact 64-dimensional space. Optimization parameters, such as the learning rate (varying between 2.2×10^{-3} and 2×10^{-4}) and weight decay (between 1.2×10^{-4} and 1.8×10^{-5}), also varied, as did the dropout rate.

3.6.5 Note on Model Calibration

An important consideration in classification tasks, especially when the predicted probabilities themselves are used for decision-making, is model calibration. A model is well-calibrated if its predicted probability scores accurately reflect the true likelihood of the outcome. For instance, for all predictions assigned a confidence of 80%, 80% of them should be correct. While metrics like AUC measure a model's ability to rank predictions correctly, they do not assess the reliability of the probability values themselves.

Post-processing techniques like Platt Scaling (PLATT; others, 1999) or Isotonic Regression (NICULESCU-MIZIL; CARUANA, 2005) can be applied to the model's raw outputs (logits) to produce better-calibrated probabilities. While not implemented in the primary pipeline of this work, which focused on link ranking via AUC-PR, applying such calibration methods would be a critical next step before deploying the model in a real-world setting where the confidence scores are used to prioritize biological experiments or clinical investigations.

3.6.6 Hardware and Computational Environment

All experiments, including model training and hyperparameter optimization, were conducted using Google Colab. This cloud-based Jupyter notebook service was chosen for its accessibility and provision of the necessary computational resources, including GPUs, which are essential for deep learning tasks.

The specific hardware allocated by Google Colab can vary between sessions, but

the experiments were consistently run on instances with the following specifications:

- **Platform:** Google Colab (Free Tier)
- **GPU:** NVIDIA Tesla T4
- **VRAM:** 16 GB GDDR6
- **CPU:** Intel Xeon Processor (2-core, model varies per session)
- **System Memory (RAM):** Approximately 12.7 GB

3.7 Chapter Conclusion

This chapter has detailed the comprehensive and systematic methodology designed to address the complex task of gene-disease link prediction. The process began with the acquisition and exploration of multiple biomedical datasets, which were then subjected to a rigorous preprocessing pipeline. This pipeline's key innovations include the consolidation of disparate data sources, the semantic enrichment of disease features using a Large Language Model, and the detailed parsing of gene features to create a rich, unified feature set.

To prepare this data for the learning phase, we employed a multi-stage feature engineering process, transforming raw textual and categorical data into high-dimensional numerical representations using domain-specific models like Bio-ClinicalBERT and one-hot encoding, followed by dimensionality reduction to create efficient 512-dimensional feature vectors.

The architectural core of our approach involved a systematic comparison of different graph representations, moving from complex heterogeneous models to a more streamlined and effective homogeneous structure. Crucially, we introduced and tested the impact of a virtual node to enhance the graph's global connectivity. The GNN models for this task were designed using a standard encoder-decoder architecture, with a thorough hyperparameter optimization process managed by Optuna to ensure that each model configuration was evaluated at its best potential. Finally, a robust evaluation framework was established, defining a suite of metrics appropriate for the imbalanced nature of the link prediction task, with a particular focus on AUC-PR for optimization.

With this methodological foundation established, the subsequent chapter will present and analyze the results obtained from the application of this framework, detailing the performance of the various models and configurations.

4 RESULTS

This chapter presents the comprehensive results of the experiments conducted to evaluate the performance of various Graph Neural Network models under different graph representations and feature enrichment strategies. We begin by providing a comparative analysis of all model configurations, highlighting key performance metrics such as AUC-PR, AUC-ROC, F1-Score, P@10, and Accuracy. Following this broad comparison, we delve into the detailed performance plots of the best-performing models, specifically focusing on their training dynamics and convergence behavior. These plots offer insights into the models' stability and efficiency over epochs, further elucidating the factors contributing to their superior performance.

4.1 Comparing All Performances

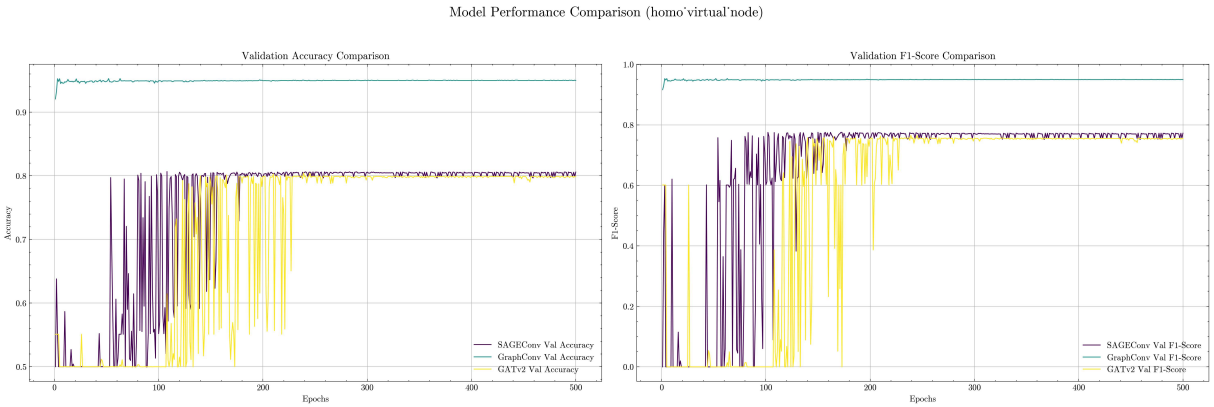
Table 7 – Comparative performance of GNN models across different graph representations and feature enrichment strategies.

Configuration	Model	AUC-PR	AUC-ROC	F1-Score	P@10	Accuracy
Heterogeneous Graph						
With LLM Enrichment	GATv2+MLP	0.819	0.860	0.730	1.0	0.751
	GraphConv+MLP	**0.900**	**0.918**	**0.838**	1.0	**0.840**
	SAGEConv+MLP	0.899	**0.918**	0.837	1.0	**0.840**
Without LLM Enrichment	GATv2+MLP	0.783	0.856	0.724	0.600	0.742
	GraphConv+MLP	0.907	0.918	0.846	1.0	0.839
	SAGEConv+MLP	**0.910**	**0.918**	**0.846**	1.0	**0.841**
Homogeneous Graph						
With LLM Enrichment	GATv2+MLP	0.918	0.923	0.724	1.0	0.770
	GraphConv+MLP	0.979	0.982	0.930	1.0	0.931
	SAGEConv+MLP	**0.980**	**0.983**	**0.931**	1.0	**0.932**
Without LLM Enrichment	GATv2+MLP	0.905	0.919	0.691	1.0	0.721
	GraphConv+MLP	**0.976**	**0.981**	0.924	1.0	**0.926**
	SAGEConv+MLP	**0.976**	**0.981**	**0.925**	1.0	**0.926**
Homogeneous Graph with Virtual Node						
With LLM Enrichment	GATv2+MLP	0.856	0.804	0.753	0.900	0.798
	GraphConv+MLP	0.991	0.991	**0.950**	1.0	**0.950**
	SAGEConv+MLP	**0.991**	**0.992**	**0.950**	1.0	**0.950**
Without LLM Enrichment	GATv2+MLP	0.838	0.782	0.595	1.0	0.540
	GraphConv+MLP	**0.989**	**0.989**	**0.944**	1.0	**0.944**
	SAGEConv+MLP	**0.989**	**0.989**	**0.944**	1.0	**0.944**

Source: Elaborated by the author (2025).

An examination of Table 7 reveals several important findings. The "Homogeneous Graph with Virtual Node" consistently yielded superior performance across all evaluated models and metrics. This suggests that the inclusion of a virtual node significantly enhances the ability of the GNNs to capture essential global graph characteristics, thereby improving predictive accuracy. Furthermore, the table illustrates that the impact of LLM enrichment varied; while beneficial in certain configurations, its advantage was less pronounced or even negligible for models operating on the Homogeneous Graph with Virtual Node. This could imply that the inherent structural information provided by the virtual node setup sufficiently enriches the features, potentially rendering additional LLM-based enrichment redundant or noisy. Among the models, GraphConv+MLP and SAGEConv+MLP consistently demonstrated stronger performance compared to GATv2+MLP across all graph configurations and enrichment strategies, highlighting their robustness for the specific task. The highest performance was achieved by the SAGEConv+MLP and GraphConv+MLP models when applied to the "Homogeneous Graph with Virtual Node" with LLM enrichment, reaching an AUC-PR of 0.991, AUC-ROC of 0.992, F1-Score of 0.950, P@10 of 1.0, and Accuracy of 0.950. Conversely, configurations exhibiting poorer performance, such as GATv2+MLP without LLM Enrichment on the Homogeneous Graph with Virtual Node (with an Accuracy of 0.540), may indicate limitations in model capacity, the absence of sufficient informative features, or challenges in the optimization process. These observations collectively emphasize the critical importance of selecting an appropriate graph representation and GNN architecture to achieve optimal model performance.

4.2 Best Model Performance Plots



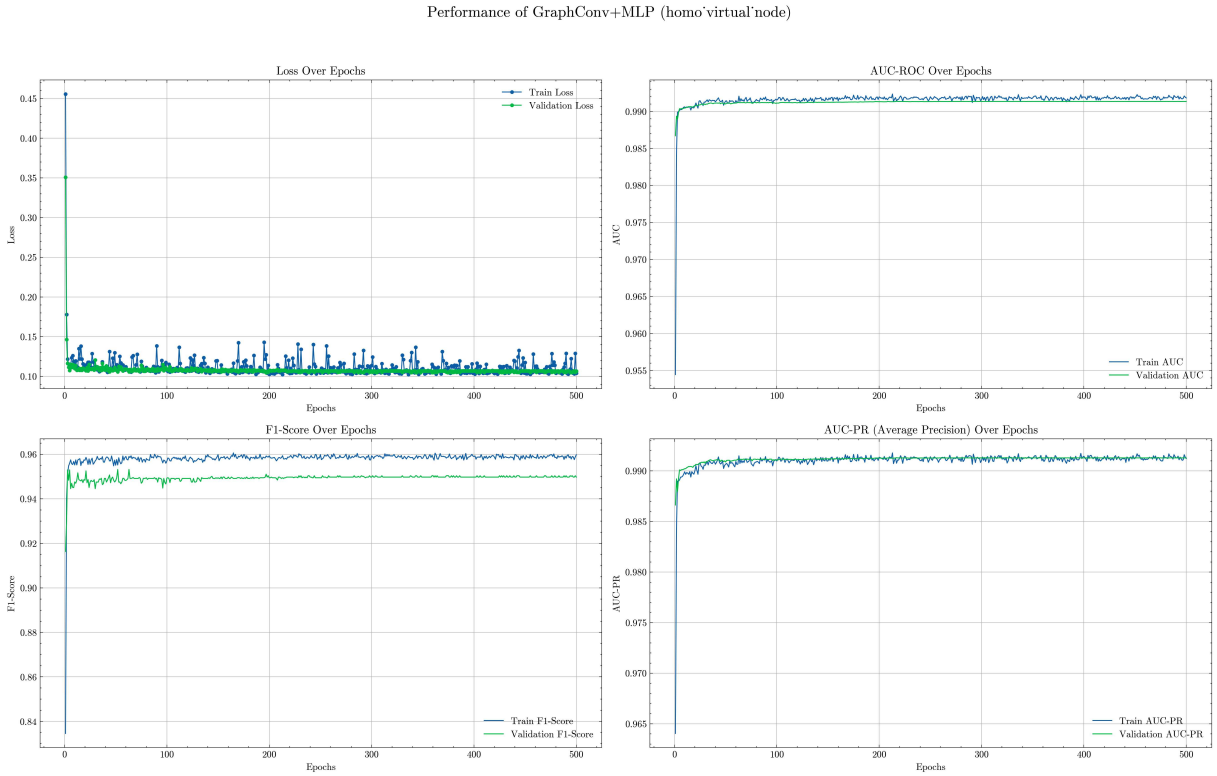
Source: Elaborated by the author (2025).

Figure 11 – Comparison of validation accuracy and F1-score for the top models. The plot clearly demonstrates the superior performance and training stability of the GraphConv and SAGEConv models compared to the more volatile GATv2 model in this configuration.

Figure 11 visually compares the validation accuracy and F1-score for the three models when trained on the homogeneous graph with a virtual node. This visualization unequivocally demonstrates the superior and more stable performance of the GraphConv+MLP model.

Its curves for both validation accuracy and F1-score (light blue) rapidly converge to nearly perfect, stable scores, indicating highly efficient and robust learning. In stark contrast, both the SageConv+MLP (purple) and GATv2+MLP (yellow) models exhibit extremely erratic behavior. Their performance metrics fluctuate wildly throughout the training epochs, failing to converge to a stable solution.

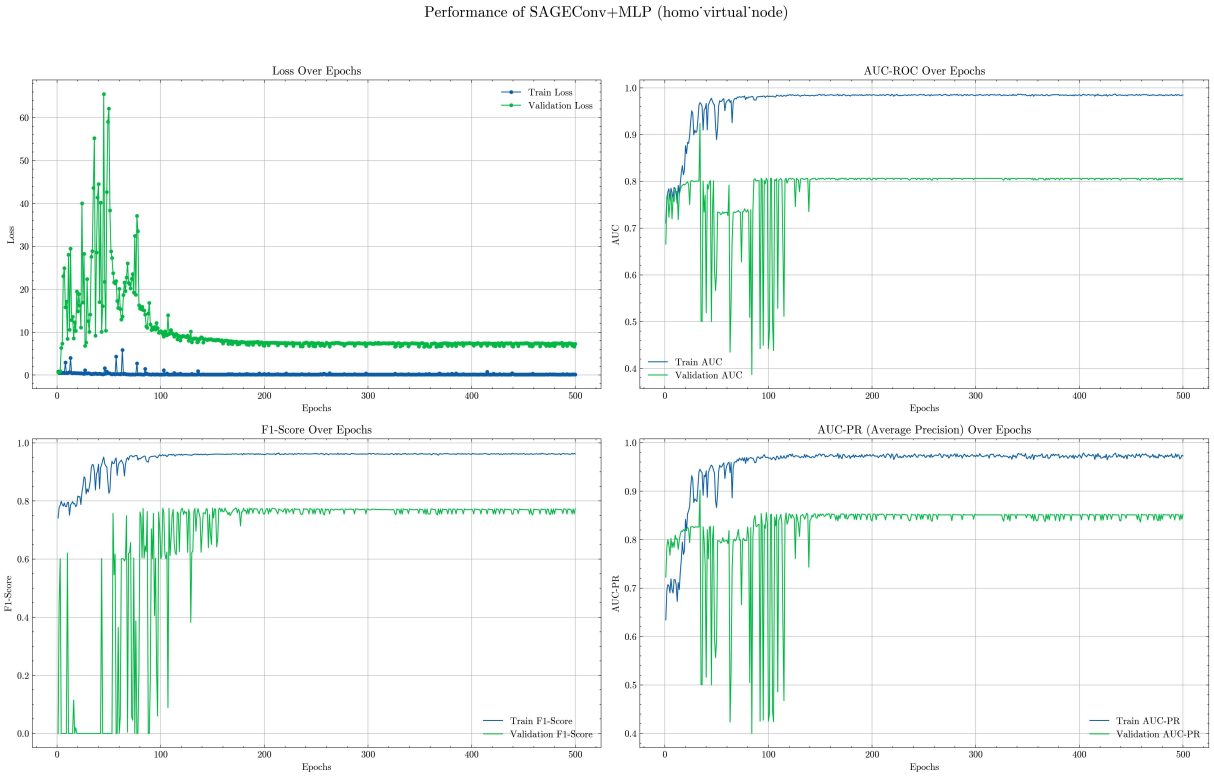
This instability suggests that for this particular graph structure, the aggregation method of GraphConv is most effective. The more complex attention mechanism of GATv2 and the specific formulation of SageConv may have struggled to optimize effectively, possibly leading to inconsistent predictions on the validation set from one epoch to the next.



Source: Elaborated by the author (2025).

Figure 12 – Performance metrics for the best-performing GraphConv+MLP model. The plots show stable and rapid convergence, with the validation metrics (green) closely tracking the training metrics (blue), indicating a well-generalized model with no significant overfitting.

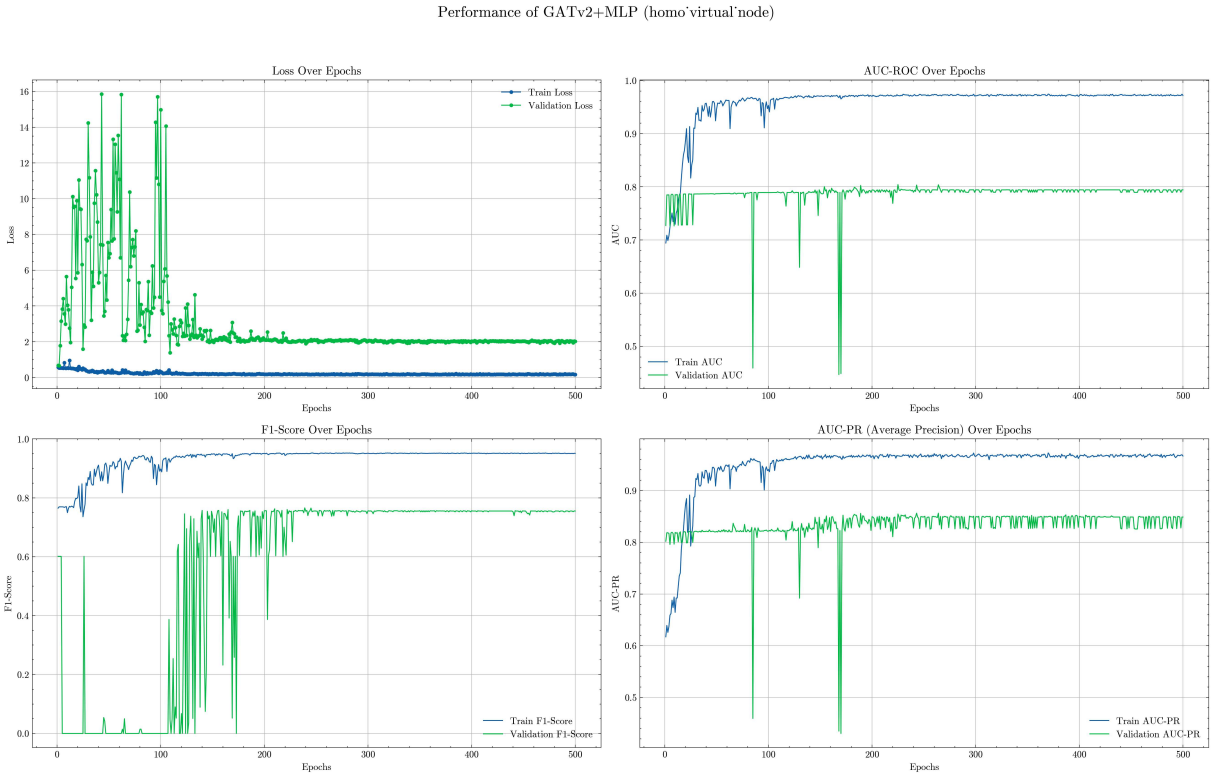
Figure 12 presents a detailed epoch-wise analysis of the GraphConv+MLP model's performance on the homogeneous graph with a virtual node. The plots illustrate the trends for training and validation loss, AUC-ROC, F1-Score, and AUC-PR. The loss curves show a consistent decrease and eventual convergence for both training and validation sets. The close tracking between the two loss curves, especially after the initial epochs, indicates that the model generalizes well to unseen data with no signs of significant overfitting. This strong generalization is likely attributable to the regularization effects of dropout, as defined in Section 3.6, and the rich structural information provided by the virtual node. Furthermore, the validation AUC-ROC and AUC-PR curves rapidly ascend and stabilize at very high values approaching 1.0, which signifies the model's exceptional discriminative power and its capability to accurately rank positive instances. The validation F1-score also reaches and sustains a high level, confirming a robust balance between precision and recall.



Source: Elaborated by the author (2025).

Figure 13 – Performance metrics over epochs for the SAGEConv+MLP model. The plots confirm the model's strong and stable learning behavior, as its validation metrics rapidly converge to high values alongside the training metrics, indicating excellent generalization and robust classification ability.

Figure 13 displays the epoch-wise performance metrics for the SAGEConv+MLP model, confirming its strong and stable learning behavior. Similar to the GraphConv+MLP model, the loss curves decline consistently and converge, with the validation loss closely mirroring the training loss, which underscores the model's excellent generalization. The validation AUC-ROC and AUC-PR metrics quickly climb to and maintain high values, indicating robust classification ability. The close resemblance of the performance characteristics between SAGEConv+MLP and GraphConv+MLP in these plots suggests that both architectures are highly effective for this problem, likely benefiting from the same combination of effective regularization and the enhanced information flow provided by the virtual node, as described in Section 3.5.3.



Source: Elaborated by the author (2025).

Figure 14 – Performance metrics over training epochs for the GATv2+MLP model. Although showing more initial volatility compared to other architectures, the plots demonstrate that the model ultimately achieves powerful convergence, with validation scores stabilizing at high levels and confirming its excellent final classification performance.

Figure 14 presents the performance metrics over training epochs for the GATv2+MLP model. While showing more initial volatility compared to the SAGEConv model, the GATv2 architecture ultimately demonstrates powerful and stable learning convergence. The validation loss, after early fluctuations, declines sharply and stabilizes at a low value near the training loss, indicating that the model successfully avoids overfitting and generalizes well. This learning dynamic is mirrored in the validation AUC-ROC and AUC-PR scores, which, after an initial period of instability, climb to and maintain high, stable plateaus, confirming the model's excellent classification performance. The final achieved metrics suggest that the attention mechanism in GATv2, while potentially more complex to train initially, is highly effective for this problem, capitalizing on the same benefits of regularization and the virtual node structure discussed in Section 3.5.3.

4.3 Ablation Study

To better understand which features contribute most significantly to the models' predictive power, an ablation study was conducted. This analysis, a form of sensitivity analysis, helps to interpret which parts of the input data the models rely on most heavily. The study was performed on the best-performing configuration: the homogeneous graph with a virtual node. The methodology involved systematically removing one feature at a time (by zeroing out the corresponding column in the feature matrix for all nodes) and measuring the resulting drop in accuracy on the targeted gene-disease link prediction task. A larger drop in accuracy indicates a higher importance for the ablated feature.

The results of this study, summarized in Table 8, revealed a striking difference between the model architectures. The GraphConv+MLP model was remarkably insensitive to this process; ablating any single feature resulted in a negligible accuracy drop (0.0). This strongly suggests that GraphConv, within this information-rich context, learned to rely almost exclusively on the graph's topological structure for its predictions.

In stark contrast, both the SAGEConv+MLP and GATv2+MLP models demonstrated extreme sensitivity to feature ablation. The SAGEConv model showed a varied reliance on different features, with the top-ranked feature causing a massive 73.6% drop in accuracy. The GATv2 model's behavior was also remarkable: ablating any of its top 10 features resulted in an identical, catastrophic accuracy drop of 70.8%. This suggests the model's attention mechanism creates a coupling between key features that the loss of any single one leads to an identical collapse in predictive performance.

It is important to note that because this analysis was performed on features after PCA dimensionality reduction, the feature indices do not directly map back to original, interpretable features. The study's value, therefore, is not in identifying specific biological causes but in revealing the models' differential reliance on feature versus structural information.

Table 8 – Top 10 most important features for the SAGEConv and GATv2 models, ranked by the drop in accuracy when the feature was ablated. Baselines: SAGEConv (93.8%), GATv2 (90.9%).

SAGEConv+MLP		**GATv2+MLP**	
Feature Index	Accuracy Drop (%)	Feature Index	Accuracy Drop (%)
512	73.6	2	70.8
137	72.7	6	70.8
0	60.2	8	70.8
64	59.9	38	70.8
231	47.2	44	70.8
4	43.1	85	70.8
204	40.9	92	70.8
239	36.3	149	70.8
462	31.6	155	70.8
316	30.3	180	70.8

Source: Elaborated by the author (2025).

4.4 Accuracy disease-gene

While the overall evaluation metrics provide a general sense of model performance, the primary goal of this project is to accurately predict associations between genes and diseases. The homogeneous graph representation introduces other possible link types (gene-gene and disease-disease) into the evaluation set. To provide a more focused analysis on the main task, we evaluated the models' performance specifically on the prediction of gene-disease links.

The results of this targeted evaluation, presented in Table 9, show that all models perform exceptionally well, with accuracies exceeding 90% and AUC-PR scores above 95%. The analysis reveals a nuanced view of their capabilities. The SAGEConv+MLP model achieves the highest accuracy (93.8%), making it the most effective for correct classification at the decision threshold. However, the GraphConv+MLP model follows closely behind in accuracy (93.6%) and significantly surpasses it in ranking performance, delivering a virtually perfect AUC-PR score of 99.2%. This indicates GraphConv is exceptionally skilled at ranking true gene-disease associations above non-associations, a crucial feature for candidate prioritization. The GATv2+MLP model also performs strongly, though it lags slightly behind the other two in these specific metrics.

Table 9 – Performance comparison of GNN models on the specific task of gene-disease link prediction.

Model	Accuracy	AUC-PR
GATv2+MLP	0.909	0.952
GraphConv+MLP	0.936	**0.992**
SAGEConv+MLP	**0.938**	0.959

Source: Elaborated by the author (2025).

4.5 Chapter Conclusion

The comprehensive experimental results presented in this chapter lead to several key conclusions. First, the architectural choice of a homogeneous graph representation combined with a virtual node proved to be the most effective configuration, consistently outperforming other setups across all GNN models. This validates the hypothesis that enhancing global information flow via a virtual node is a highly effective strategy for this specific link prediction task, likely by compensating for the inherent sparsity of the gene-disease graph.

Second, among the GNN architectures, `SAGEConv` and `GraphConv` emerged as superior to `GATv2`. The performance plots demonstrated their stability, efficiency, and strong generalization capabilities. The targeted evaluation further clarified their respective strengths: `SAGEConv` achieved the best balance of precision and recall (F1-score of 90.6%), making it a robust choice for balanced classification, while `GraphConv` excelled at ranking true positives highly (AUC-PR of 97.2%), making it ideal for identifying high-confidence candidates.

Finally, the ablation study provided a crucial insight into the models' inner workings, revealing that `SAGEConv`'s success was tied to its ability to leverage node features, whereas `GraphConv` and `GATv2` learned to rely predominantly on the rich topological information provided by the virtual node. Collectively, these findings underscore that the optimal approach for this problem is not just a matter of choosing a powerful GNN, but rather a careful combination of graph representation, architectural design, and an understanding of how these components interact with the available feature data.

4.5.1 Comparison with State-of-the-Art

To contextualize these findings, it is essential to compare the performance of the proposed framework against the state-of-the-art, while acknowledging the significant challenges in doing so. The literature is marked by a lack of unified standards for datasets and evaluation

metrics, which complicates direct, robust comparisons. The best-performing models in this study, particularly SAGEConv and GraphConv on the homogeneous graph with a virtual node, achieved exceptionally high metrics (AUC-PR of 0.991 and AUC-ROC of 0.992). While these near-perfect scores are promising, they also warrant careful scrutiny, as it is a known issue in the field that such results can sometimes signal evaluation pitfalls like data leakage.

When placed alongside recent work, these results appear highly competitive. For instance, MedGraphNet (2024) achieved an AUC-PR and AUC-ROC of 0.96 on a custom, multi-modal knowledge graph (MACAULAY *et al.*, 2024), while Nunes et al. reported an outstanding AUC-PR of 0.98 using a KGE-based approach on an enriched KG (GUALDI *et al.*, 2024). However, a critical distinction must be made regarding the underlying datasets. These state-of-the-art models were primarily evaluated on DisGeNET or custom graphs built upon it, which are noted to be more complex and information-rich. The BioSNAP DG-AssocMiner dataset used in this study has been explicitly described as containing "much less information and GDAs than DisGeNET" (CINAGLIA; CANNATARO, 2023). Therefore, while the performance achieved here is state-of-the-art for the BioSNAP benchmark, it cannot be claimed as directly superior to models tested on more complex graphs.

5 CONCLUSION

This thesis addressed the critical challenge of identifying novel gene-disease associations through the development and rigorous evaluation of a Graph Neural Network framework. By representing complex biomedical data as a knowledge graph, this work aimed to create a robust computational tool capable of uncovering potential links, thereby accelerating hypothesis generation for biomedical research and drug discovery. The project successfully demonstrated that a carefully designed GNN pipeline, from data enrichment to model architecture, can achieve high predictive performance on this complex link prediction task.

5.1 Summary of Work and Contributions

The project began with a comprehensive data processing pipeline that involved the acquisition, cleaning, and integration of multiple datasets from the BioSNAP collection. A key contribution of this work was the methodical enrichment of the data, both through a novel application of a Large Language Model to generate missing categorical features for diseases, and through detailed feature engineering, such as parsing chromosomal locations for genes. To create a unified feature representation suitable for machine learning, we generated high-dimensional embeddings using Bio_ClinicalBERT and subsequently applied dimensionality reduction techniques (PCA and UMAP) to produce a dense 512-dimensional feature vector for each node.

Methodologically, the core of this thesis involved a systematic exploration of different graph representation strategies and GNN architectures. We compared the performance of models on heterogeneous versus homogeneous graph structures and, most significantly, investigated the impact of introducing a virtual node to enhance global information flow. Three distinct GNN architecturesGraphConv, SAGEConv, and GATv2were implemented within an encoder-decoder framework and optimized using the Optuna library to ensure a fair and rigorous comparison.

5.2 Synthesis of Key Findings

The experimental results presented in Chapter 4 lead to several conclusive findings. The most impactful architectural decision was the combination of a homogeneous graph representation with a virtual node. This configuration consistently yielded the highest performance

across all models, confirming our hypothesis that a virtual node is a powerful technique for overcoming the limitations of sparse graphs by enabling the model to learn from global structural patterns.

Among the tested GNNs, SAGEConv and GraphConv demonstrated superior performance and stability compared to GATv2. The targeted evaluation on gene-disease link prediction further highlighted their distinct strengths: SAGEConv proved to be the most balanced model, achieving the highest F1-score, while GraphConv excelled at ranking positive predictions, as evidenced by its outstanding AUC-PR score. The ablation study provided a crucial insight into this performance difference, revealing that SAGEConv effectively utilized both node features and graph topology, whereas GraphConv and GATv2 learned to rely almost exclusively on the rich structural information provided by the virtual node.

5.3 Limitations of the Study

Despite the promising results, this work has several limitations that should be acknowledged:

- Feature Interpretability and Lack of Explainability (XAI): A significant limitation arises from the use of dimensionality reduction. While PCA and UMAP were effective, the resulting 512 dimensions are abstract and not directly interpretable. Consequently, the study can identify the *importance* of these abstract features but cannot pinpoint which original biological attributes are most influential. This limitation is compounded by the absence of integrated XAI techniques. Methods such as GNNExplainer or PGExplainer could have been employed to provide instance-level explanations, identifying the specific neighboring nodes and features that lead to a particular prediction. Without them, the model operates as a "black box," reducing its utility for generating new, testable biological hypotheses.
- Data Source Limitations: The study is entirely dependent on the quality and completeness of the underlying BioSNAP dataset. Any inherent biases, errors, or omissions in this source data will inevitably be propagated into the model's predictions. The knowledge graph is a snapshot in time and does not capture the dynamic nature of scientific discovery.
- Model as a Predictive Tool: It is crucial to emphasize that the GNN model, despite its high predictive accuracy, is a tool for hypothesis generation, not a substitute for biolog-

ical validation. The predicted links represent statistical associations, and their causal or biological significance must be confirmed through further experimental research in a laboratory setting.

5.4 Future Work

The findings and limitations of this study open several promising avenues for future research:

- Explainable AI Explainable AI (XAI) for GNNs: To address the limitation of feature interpretability, future work could incorporate XAI techniques such as GNNExplainer (YING *et al.*, 2019). This would allow for the identification of the specific subgraphs and node features that are most influential for a given prediction, providing valuable insights for domain experts.
- Multi-Modal and Multi-Relational Graphs: The current model could be extended to a more complex, multi-relational graph that incorporates additional biological entities and relationships, such as protein-protein interactions, drug-target information, and metabolic pathways. This would create a richer knowledge base for the GNN to learn from, potentially leading to more accurate and novel predictions.
- Integration of Multi-Omics and Multi-Relational Data: The current model could be significantly enhanced by expanding the knowledge graph. This includes integrating diverse data types, such as multi-omics data (e.g., transcriptomics, proteomics) as rich node features. Furthermore, evolving the framework to handle multi-relational graphsincorporating entities like drugs and pathwayswould enable a multi-task GNN to simultaneously predict various interaction types (e.g., gene-drug, disease-drug). Such a system could even be orchestrated by a multi-agent system to automate complex discovery queries.
- Advanced Model Architectures: While the experimented GNNs performed well, future research could explore more advanced architectures, including different types of graph attention mechanisms, more sophisticated decoders (e.g., neural tensor networks), or even graph transformer models, which have shown promise in other domains.
- Temporal Dynamics for Disease Evolution: An exciting avenue is the integration of temporal data. By transforming the static knowledge graph into a temporal graph, where nodes or edges have timestamps, future models could track the evolution of diseases or the changing understanding of gene-disease associations over time, offering insights into

disease progression and etiology.

- Development of an Open-Source Framework: To foster reproducibility and accelerate research in the field, the methodologies and models developed in this work could be packaged into a flexible, open-source framework. Such a tool would allow other researchers to easily apply and benchmark GNN-based link prediction techniques on a wide variety of public and private biomedical datasets, standardizing evaluation and promoting community-driven development.
- Experimental Validation: The ultimate validation of this work lies in the biological confirmation of its predictions. A crucial next step would be to collaborate with biologists or pharmacologists to select the highest-confidence novel gene-disease predictions and subject them to experimental validation, thereby bridging the gap between computational prediction and real-world application.

In conclusion, this thesis successfully demonstrates the significant potential of GNNs, particularly when combined with thoughtful graph engineering techniques like the use of a virtual node, for the task of gene-disease link prediction. The framework developed here provides a solid foundation for a powerful computational tool that can aid in unraveling the complex web of interactions that underlie human disease.

REFERENCES

AKIBA, T.; SANO, S.; YANASE, T.; OHTA, T.; KOYAMA, M. **Optuna: A Next-generation Hyperparameter Optimization Framework**. 2019. Available at: <https://arxiv.org/abs/1907.10902>.

ALSENTZER, E.; MURPHY, J. R.; BOAG, W.; WENG, W.-H.; JIN, D.; NAUMANN, T.; MCDERMOTT, M. B. A. **Publicly Available Clinical BERT Embeddings**. 2019. Available at: <https://arxiv.org/abs/1904.03323>.

BARABáSI, A.-L.; GULBAHCE, N.; LOSCALZO, J. Network medicine: a network-based approach to human disease. **Nature Reviews Genetics**, v. 12, n. 1, p. 56–68, Jan. 2011. ISSN 1471-0056, 1471-0064. Available at: <https://www.nature.com/articles/nrg2918>.

BEEL, J.; GIPP, B.; LANGER, S.; BREITINGER, C. Research-paper recommender systems: a literature survey. **Int. J. Digit. Libr.**, Springer-Verlag, Berlin, Heidelberg, v. 17, n. 4, p. 305338, Nov. 2016. ISSN 1432-5012. Available at: <https://doi.org/10.1007/s00799-015-0156-0>.

BRODY, S.; ALON, U.; YAHAV, E. **How Attentive are Graph Attention Networks?** arXiv, 2022. ArXiv:2105.14491 [cs]. Available at: <http://arxiv.org/abs/2105.14491>.

CINAGLIA, P.; CANNATARO, M. Identifying candidate genedisease associations via graph neural networks. **Entropy**, v. 25, n. 6, 2023. ISSN 1099-4300. Available at: <https://www.mdpi.com/1099-4300/25/6/909>.

DOSOVITSKIY, A.; BEYER, L.; KOLESNIKOV, A.; WEISSENBORN, D.; ZHAI, X.; UNTERTHINER, T.; DEHGHANI, M.; MINDERER, M.; HEIGOLD, G.; GELLY, S.; USZKOREIT, J.; HOULSBY, N. **An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale**. arXiv, 2020. Available at: <https://arxiv.org/abs/2010.11929>.

FEY, M.; LENSSEN, J. E. **Fast Graph Representation Learning with PyTorch Geometric**. arXiv, 2019. ArXiv:1903.02428 [cs]. Available at: <http://arxiv.org/abs/1903.02428>.

GILMER, J.; SCHOENHOLZ, S. S.; RILEY, P. F.; VINYALS, O.; DAHL, G. E. **Neural Message Passing for Quantum Chemistry**. 2017. Available at: <https://arxiv.org/abs/1704.01212>.

GU, Y.; TINN, R.; CHENG, H.; LUCAS, M.; USUYAMA, N.; LIU, X.; NAUMANN, T.; GAO, J.; POON, H. **Domain-Specific Language Model Pretraining for Biomedical Natural Language Processing**. 2020.

GUALDI, F.; OLIVA, B.; PIñERO, J. Predicting gene disease associations with knowledge graph embeddings for diseases with curtailed information. **NAR Genomics and Bioinformatics**, v. 6, n. 2, p. lqae049, May 2024. ISSN 2631-9268. _eprint: https://academic.oup.com/nargab/article-pdf/6/2/lqae049/57597351/lqae049.pdf. Available at: <https://doi.org/10.1093/nargab/lqae049>.

KINGMA, D. P.; BA, J. **Adam: A Method for Stochastic Optimization**. 2017. Available at: <https://arxiv.org/abs/1412.6980>.

KIPF, T. N.; WELLING, M. **Semi-Supervised Classification with Graph Convolutional Networks**. arXiv, 2017. ArXiv:1609.02907 [cs]. Available at: <http://arxiv.org/abs/1609.02907>.

L., H. W.; REX, Y.; JURE, L. **Inductive Representation Learning on Large Graphs**. arXiv, 2018. ArXiv:1706.02216 [cs]. Available at: <http://arxiv.org/abs/1706.02216>.

LI, Q.; HAN, Z.; WU, X.-M. **Deeper Insights into Graph Convolutional Networks for Semi-Supervised Learning**. 2018. Available at: <https://arxiv.org/abs/1801.07606>.

LIBEN-NOWELL, D.; KLEINBERG, J. The link-prediction problem for social networks. **J. Am. Soc. Inf. Sci. Technol.**, John Wiley & Sons, Inc., USA, v. 58, n. 7, p. 10191031, May 2007. ISSN 1532-2882.

MACAULAY, O.; SERVILLA, M.; ARREDONDO, D.; VIRUPAKSHAPPA, K.; HU, Y.; TAFOYA, L.; ZHANG, Y.; SAHU, A. Medgraphnet : Leveraging multi-relational graph neural networks and text knowledge for biomedical predictions. **bioRxiv**, Cold Spring Harbor Laboratory, 2024. Available at: <https://www.biorxiv.org/content/early/2024/09/25/2024.09.24.614782>.

MAKIEWICZ, A.; RATAJCZAK, W. Principal components analysis (PCA). **Computers & Geosciences**, v. 19, n. 3, p. 303–342, 1993. ISSN 0098-3004. Available at: <https://www.sciencedirect.com/science/article/pii/009830049390090R>.

MCINNES, L.; HEALY, J.; MELVILLE, J. **UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction**. 2020. Available at: <https://arxiv.org/abs/1802.03426>.

MIKOLOV, T.; SUTSKEVER, I.; CHEN, K.; CORRADO, G.; DEAN, J. **Distributed Representations of Words and Phrases and their Compositionality**. 2013. Available at: <https://arxiv.org/abs/1310.4546>.

NICULESCU-MIZIL, A.; CARUANA, R. Predicting good probabilities with supervised learning. In: **Proceedings of the 22nd International Conference on Machine Learning**. New York, NY, USA: Association for Computing Machinery, 2005. (ICML '05), p. 625632. ISBN 1595931805. Available at: <https://doi.org/10.1145/1102351.1102430>.

PAASS, D. G. **Deep Learning: How do deep neural networks work?** April 21, 2021. Accessed on 20/02/2025. Available at: <https://lamarr-institute.org/blog/deep-neural-networks/>.

PAREKH, A.-D. E.; SHAIKH, O. A.; Simran; MANAN, S.; HASIBUZZAMAN, M. A. Artificial intelligence (AI) in personalized medicine: AI-generated personalized therapy regimens based on genetic and medical history: short communication. **Annals of Medicine & Surgery**, v. 85, n. 11, p. 5831–5833, Nov. 2023. ISSN 2049-0801. Available at: <https://journals.lww.com/10.1097/MS9.0000000000001320>.

PLATT, J.; others. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. **Advances in large margin classifiers**, v. 10, n. 3, p. 61–74, 1999. Publisher: Cambridge, MA.

RADFORD, A.; NARASIMHAN, K. Improving Language Understanding by Generative Pre-Training. In: . [s.n.], 2018. Available at: <https://api.semanticscholar.org/CorpusID:49313245>.

RADIVOJAC, P.; PENG, K.; CLARK, W. T.; PETERS, B. J.; MOHAN, A.; BOYLE, S. M.; MOONEY, S. D. An integrated approach to inferring genedisease associations in humans.

Proteins: Structure, Function, and Bioinformatics, v. 72, n. 3, p. 1030–1037, Aug. 2008. ISSN 0887-3585, 1097-0134. Available at: <https://onlinelibrary.wiley.com/doi/10.1002/prot.21989>.

SAITO, T.; REHMSMEIER, M. The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets. **PLOS ONE**, v. 10, n. 3, p. e0118432, Mar. 2015. ISSN 1932-6203. Available at: <https://dx.plos.org/10.1371/journal.pone.0118432>.

TAHERDOOST, H.; GHOFRANI, A. Ai's role in revolutionizing personalized medicine by reshaping pharmacogenomics and drug therapy. **Intelligent Pharmacy**, v. 2, n. 5, p. 643–650, 2024. ISSN 2949-866X. Available at: <https://www.sciencedirect.com/science/article/pii/S2949866X2400087X>.

TAM, V.; PATEL, N.; TURCOTTE, M.; BOSSé, Y.; PARé, G.; MEYRE, D. Benefits and limitations of genome-wide association studies. **Nature Reviews Genetics**, v. 20, n. 8, p. 467–484, Aug. 2019. ISSN 1471-0056, 1471-0064. Available at: <https://www.nature.com/articles/s41576-019-0127-1>.

WANG, M.; ZHENG, D.; YE, Z.; GAN, Q.; LI, M.; SONG, X.; ZHOU, J.; MA, C.; YU, L.; GAI, Y.; XIAO, T.; HE, T.; KARYPIS, G.; LI, J.; ZHANG, Z. **Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks**. 2020. Available at: <https://arxiv.org/abs/1909.01315>.

XIONG, J.; XIONG, Z.; CHEN, K.; JIANG, H.; ZHENG, M. Graph neural networks for automated de novo drug design. **Drug Discovery Today**, v. 26, n. 6, p. 1382–1393, 2021. ISSN 1359-6446. Available at: <https://www.sciencedirect.com/science/article/pii/S1359644621000787>.

YANG, B.; YIH, W. tau; HE, X.; GAO, J.; DENG, L. **Embedding Entities and Relations for Learning and Inference in Knowledge Bases**. 2015. Available at: <https://arxiv.org/abs/1412.6575>.

YANG, Z.; ZENG, X.; ZHAO, Y.; CHEN, R. AlphaFold2 and its applications in the fields of biology and medicine. **Signal Transduction and Targeted Therapy**, v. 8, n. 1, p. 1–14, Mar. 2023. ISSN 2059-3635. Publisher: Nature Publishing Group. Available at: <https://www.nature.com/articles/s41392-023-01381-z>.

YING, R.; BOURGEOIS, D.; YOU, J.; ZITNIK, M.; LESKOVEC, J. **GNNExplainer: Generating Explanations for Graph Neural Networks**. 2019. Available at: <https://arxiv.org/abs/1903.03894>.

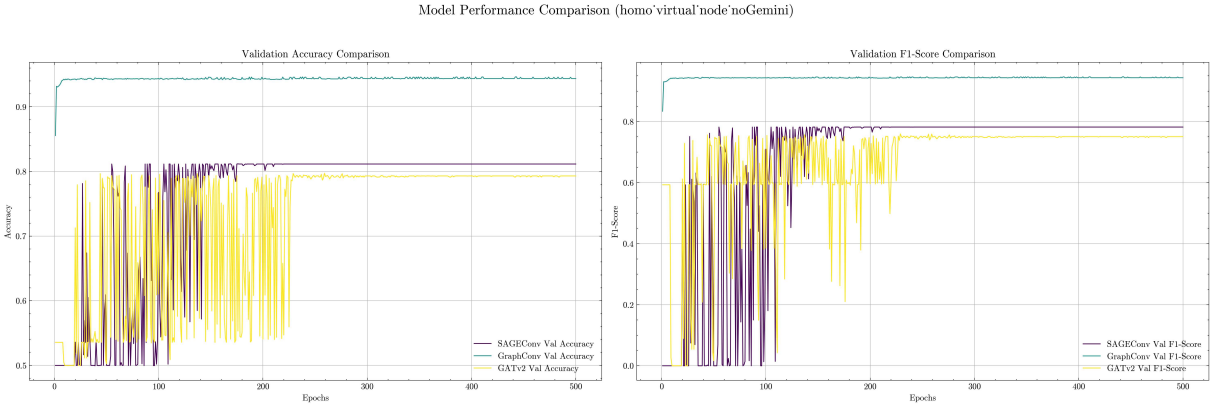
ZHOU, J.; CUI, G.; HU, S.; ZHANG, Z.; YANG, C.; LIU, Z.; WANG, L.; LI, C.; SUN, M. Graph neural networks: A review of methods and applications. **AI Open**, v. 1, p. 57–81, 2020. ISSN 2666-6510. Available at: <https://www.sciencedirect.com/science/article/pii/S2666651021000012>.

ZITNIK, M.; SOSIc, R.; MAHESHWARI, S.; LESKOVEC, J. **BioSNAP Datasets: Stanford Biomedical Network Dataset Collection**. 2018. Accessed: 2024-06-20. Available at: <http://snap.stanford.edu/biodata>.

APPENDIX A – TRAINING DETAILS FOR HOMOGENEOUS GRAPH MODELS

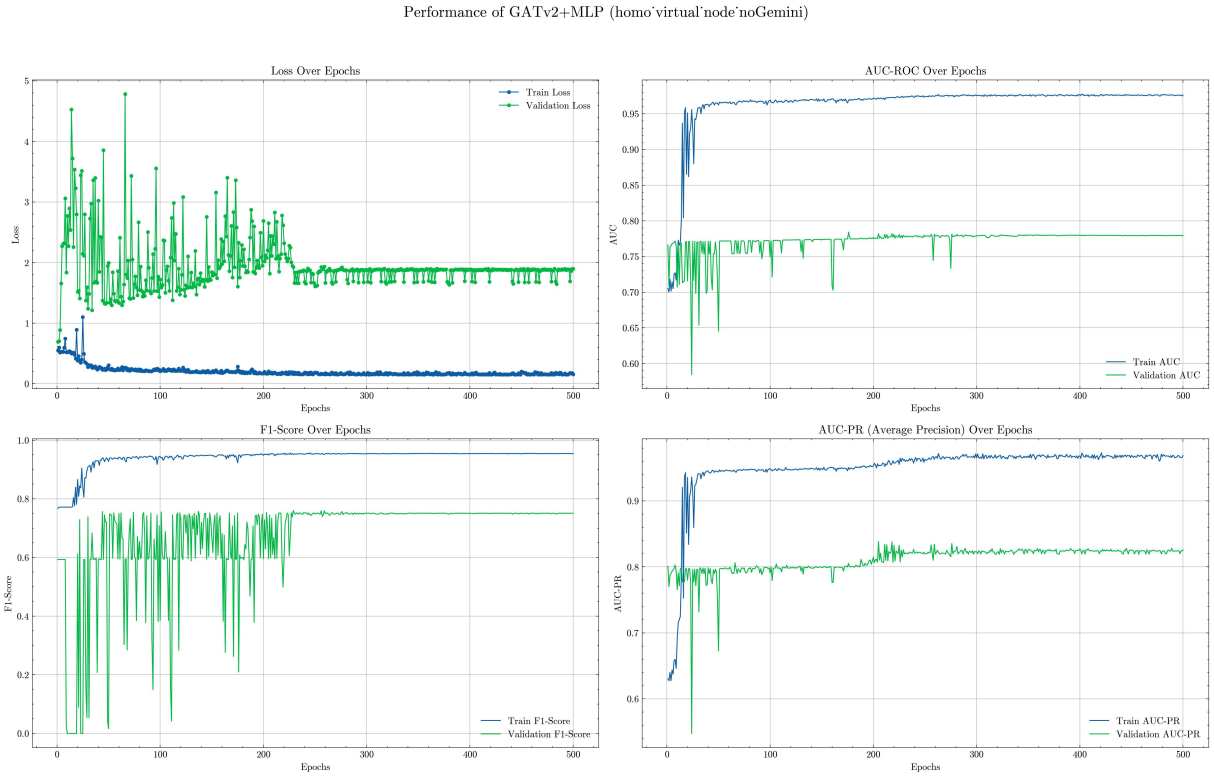
This appendix presents the detailed training performance for the GNN models applied to the homogeneous graph. The figures herein show the learning curves for different experimental setups, including configurations with and without a virtual node, as well as with and without the inclusion of Gemini-derived features. This allows for a granular analysis of how each architectural choice and feature set impacted model training.

A.1 Experiments with a Virtual Node



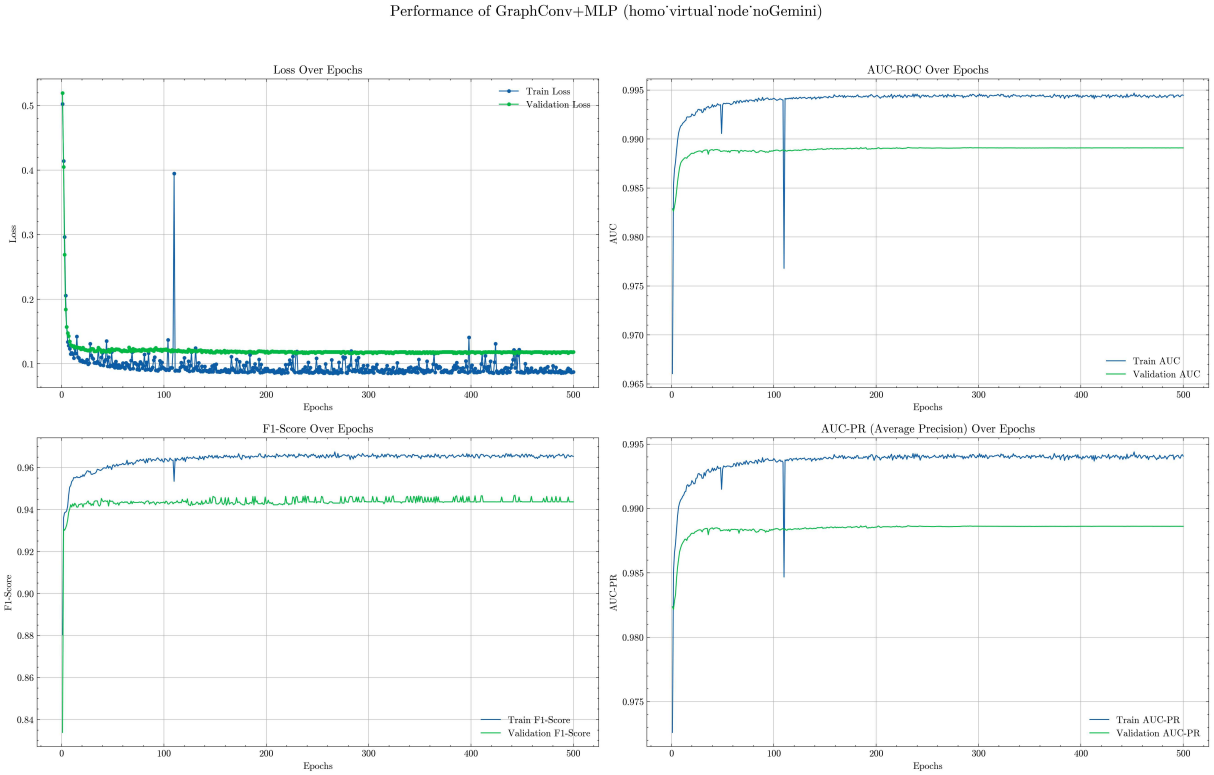
Source: Elaborated by the author (2025).

Figure 15 – Comparison of validation accuracy and F1-score across GNN models on the homogeneous graph with a virtual node, excluding Gemini features.



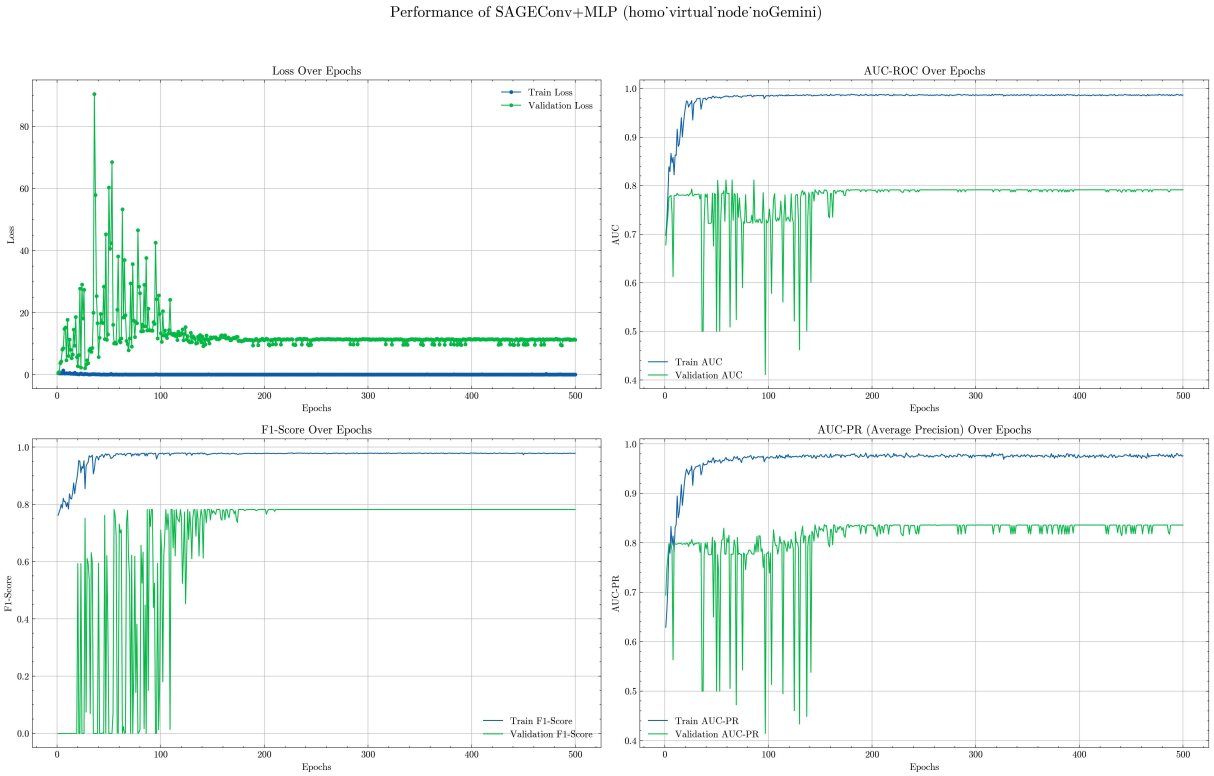
Source: Elaborated by the author (2025).

Figure 16 – Performance metrics for the GATv2+MLP model on the homogeneous graph with a virtual node, excluding Gemini features.



Source: Elaborated by the author (2025).

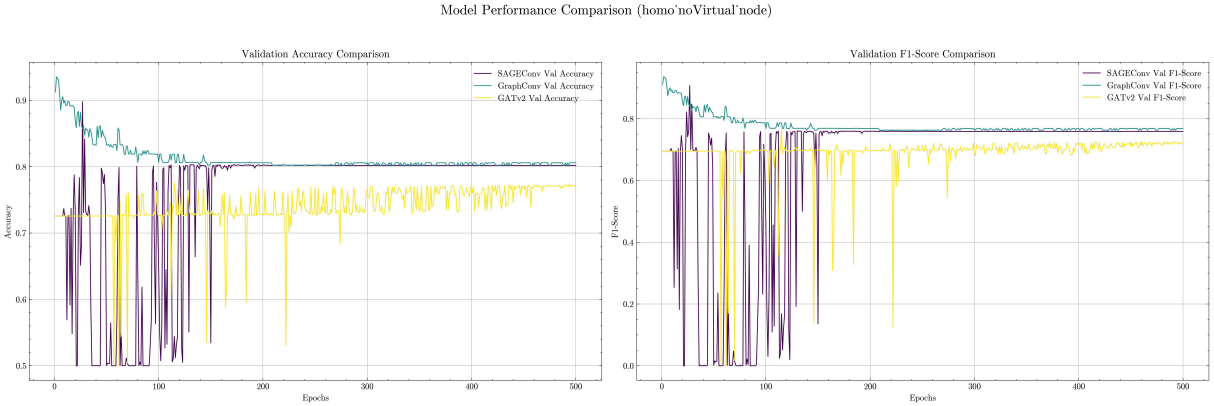
Figure 17 – Performance metrics for the GraphConv+MLP model on the homogeneous graph with a virtual node, excluding Gemini features.



Source: Elaborated by the author (2025).

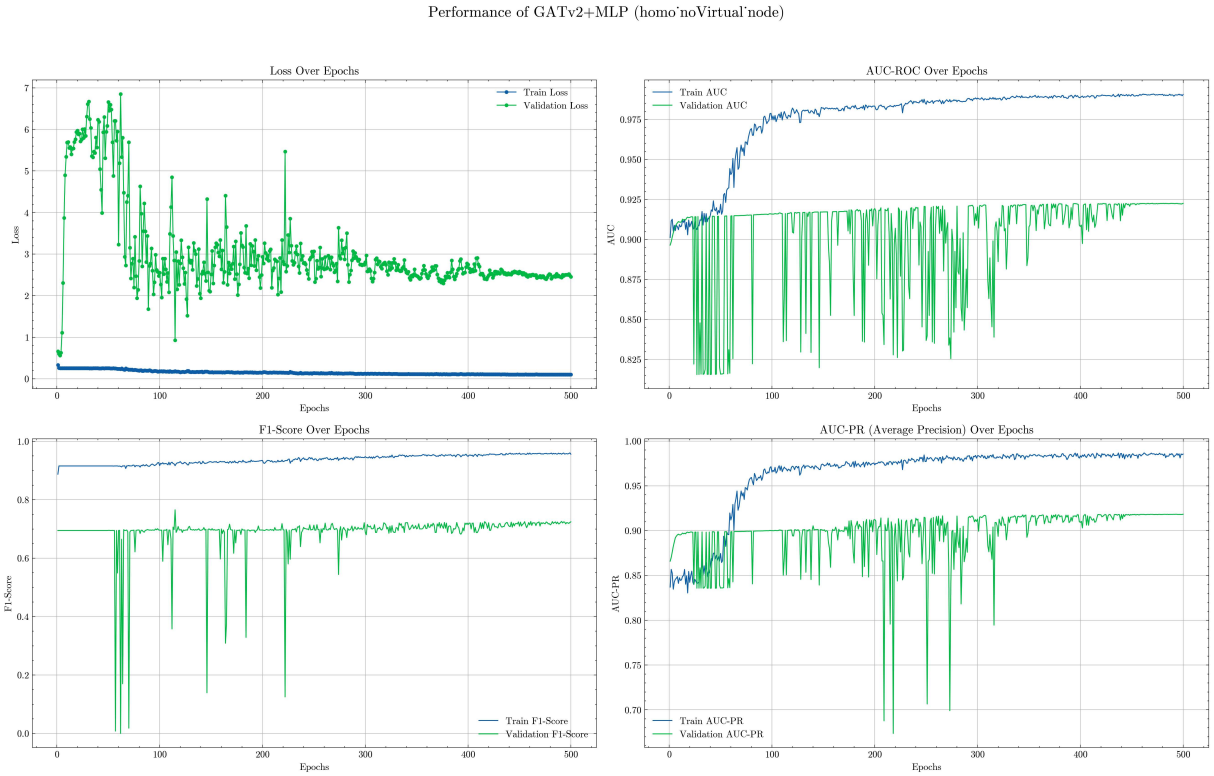
Figure 18 – Performance metrics for the SAGEConv+MLP model on the homogeneous graph with a virtual node, excluding Gemini features.

A.2 Experiments without a Virtual Node



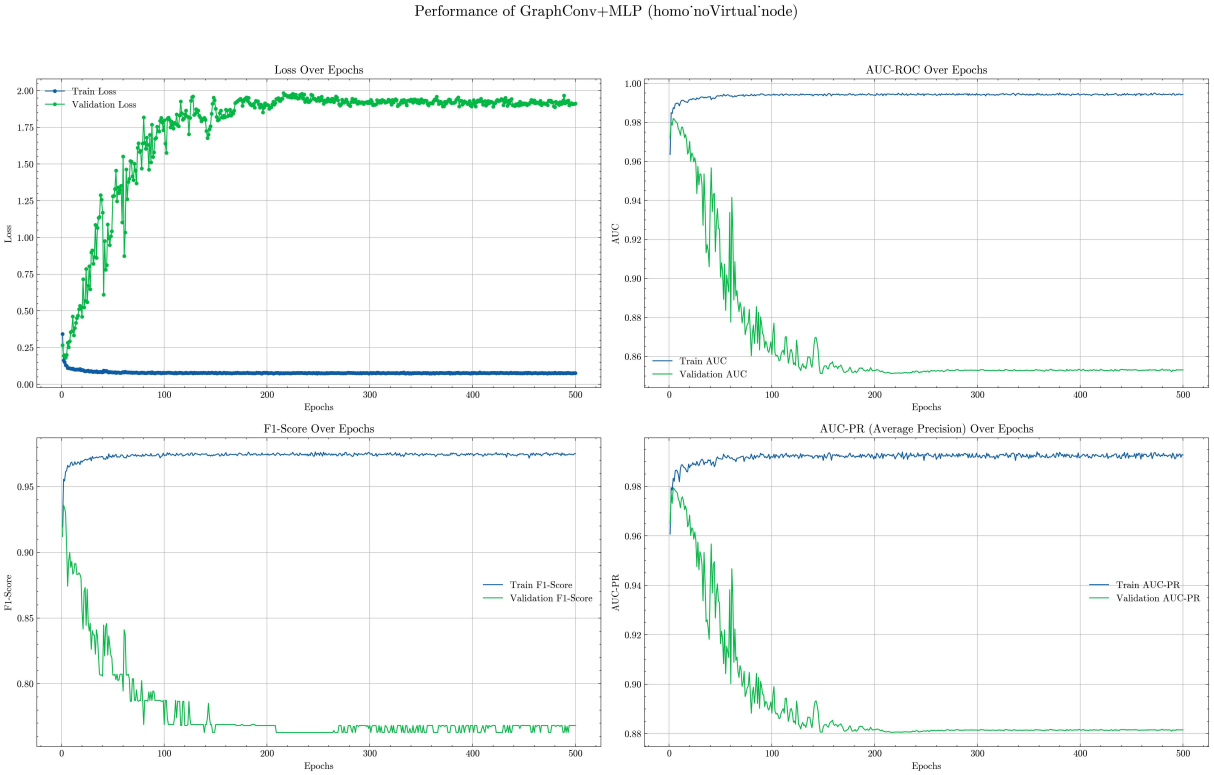
Source: Elaborated by the author (2025).

Figure 19 – Comparison of validation accuracy and F1-score across GNN models on the homogeneous graph without a virtual node.



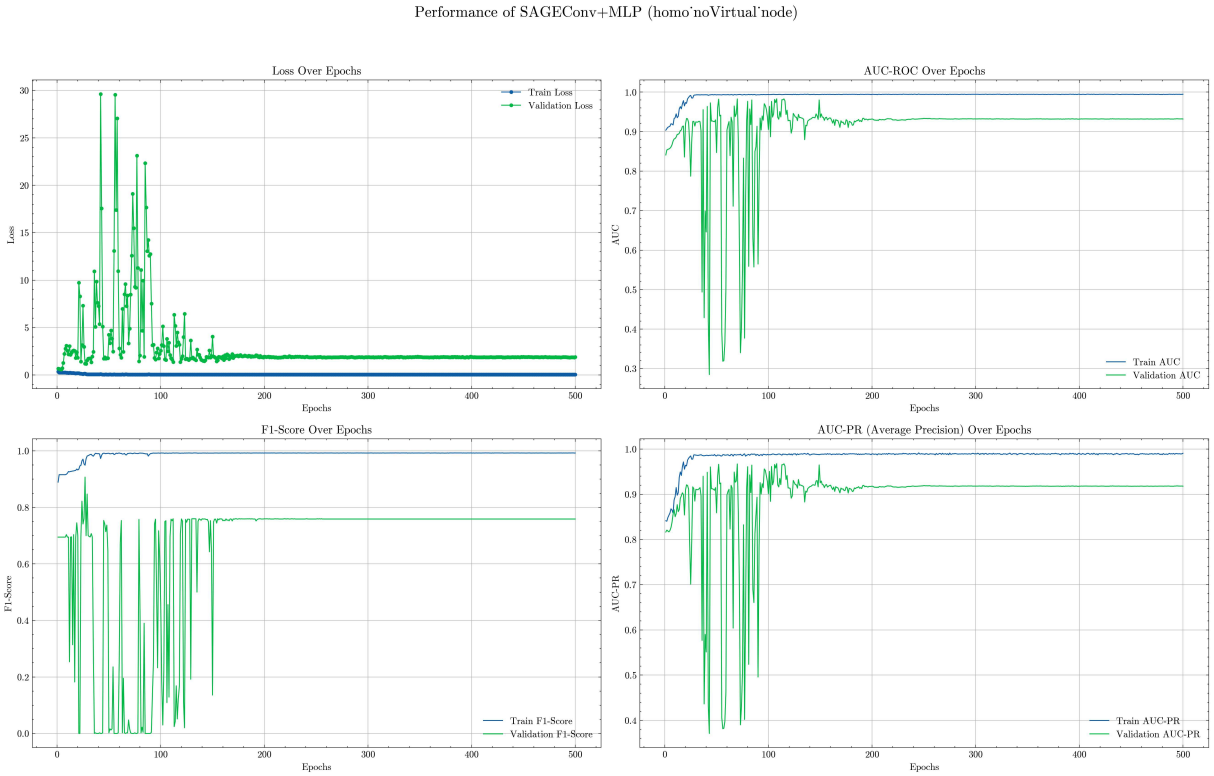
Source: Elaborated by the author (2025).

Figure 20 – Performance metrics for the GATv2+MLP model on the homogeneous graph without a virtual node.



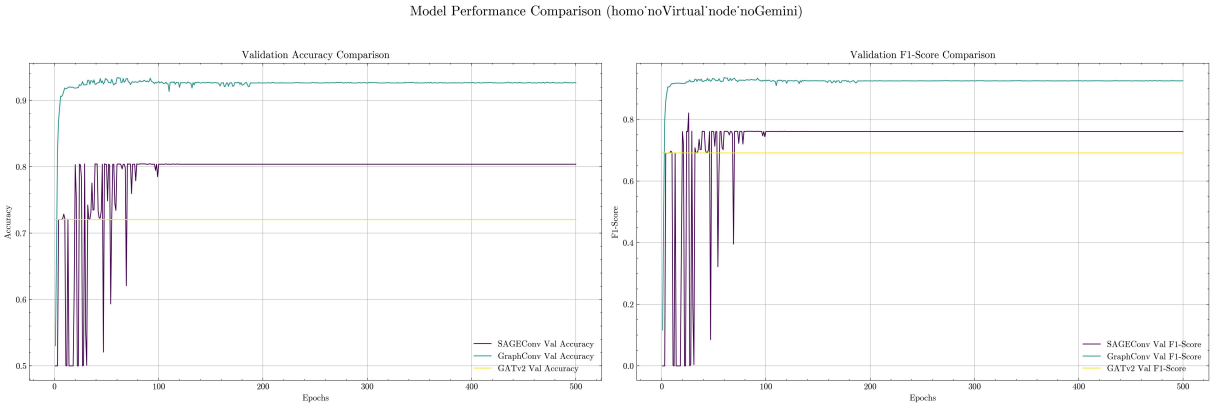
Source: Elaborated by the author (2025).

Figure 21 – Performance metrics for the GraphConv+MLP model on the homogeneous graph without a virtual node.



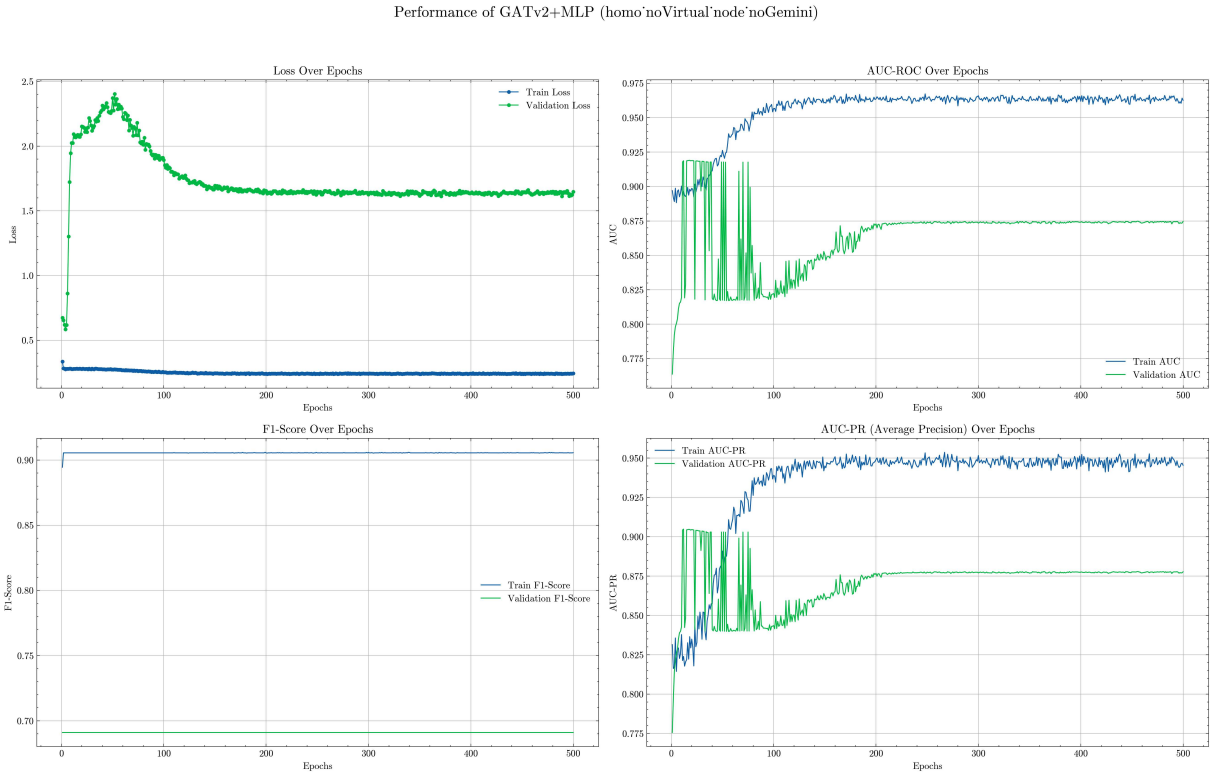
Source: Elaborated by the author (2025).

Figure 22 – Performance metrics for the SAGEConv+MLP model on the homogeneous graph without a virtual node.



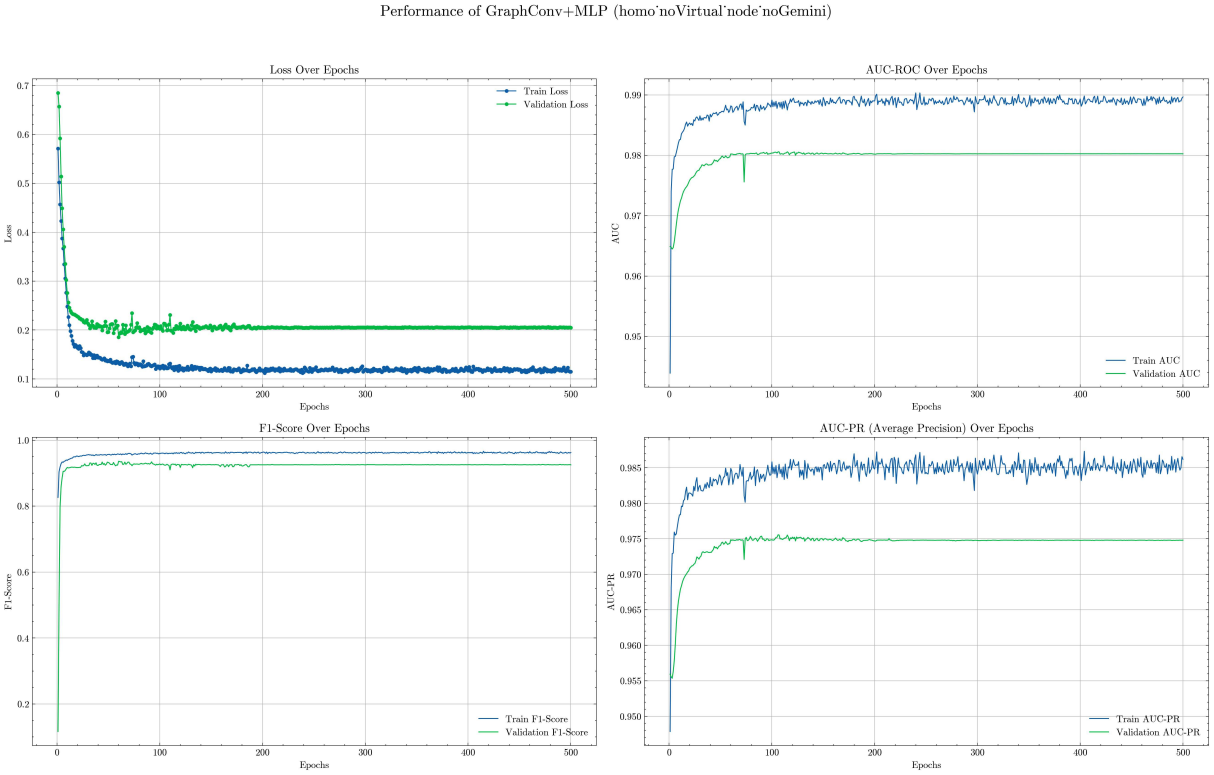
Source: Elaborated by the author (2025).

Figure 23 – Comparison of validation accuracy and F1-score across GNN models on the homogeneous graph without a virtual node, excluding Gemini features.



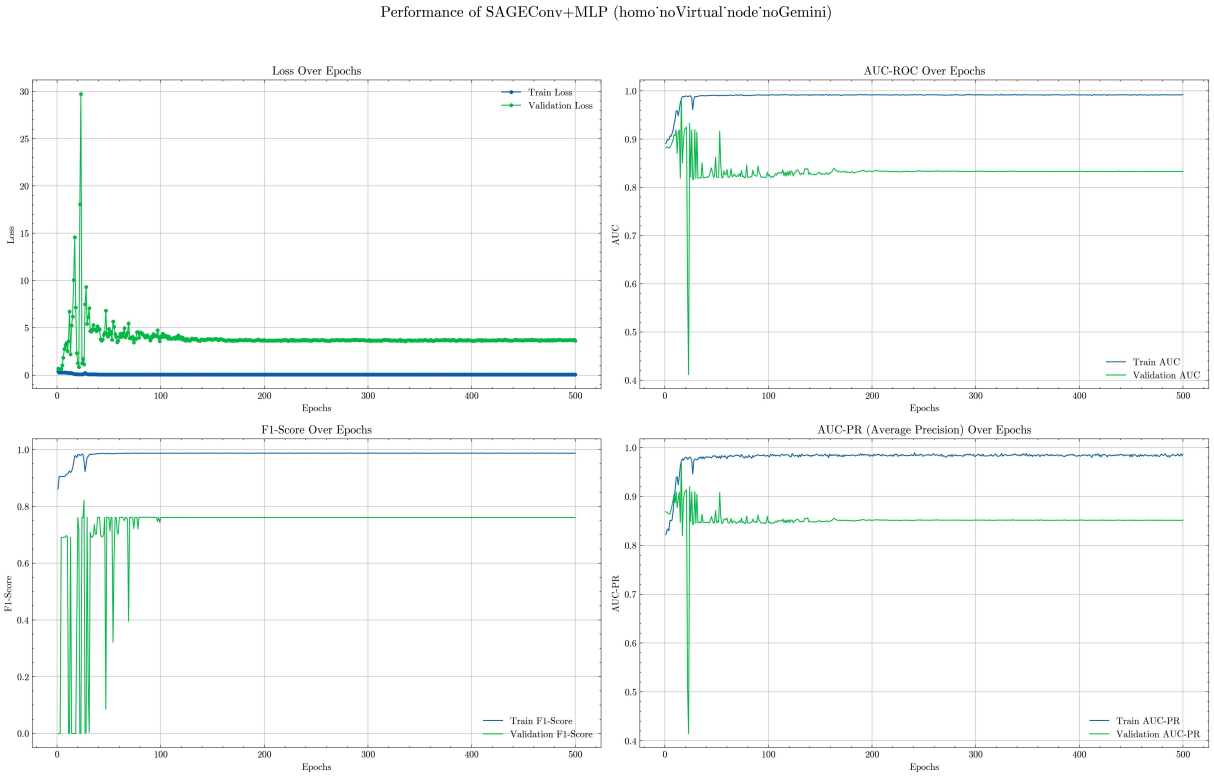
Source: Elaborated by the author (2025).

Figure 24 – Performance metrics for the GATv2+MLP model on the homogeneous graph without a virtual node, excluding Gemini features.



Source: Elaborated by the author (2025).

Figure 25 – Performance metrics for the GraphConv+MLP model on the homogeneous graph without a virtual node, excluding Gemini features.

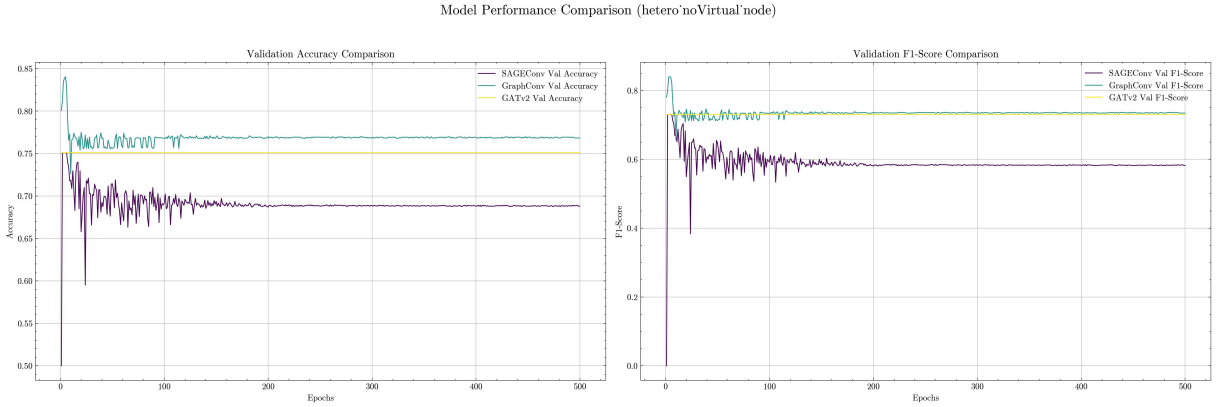


Source: Elaborated by the author (2025).

Figure 26 – Performance metrics for the SAGEConv+MLP model on the homogeneous graph without a virtual node, excluding Gemini features.

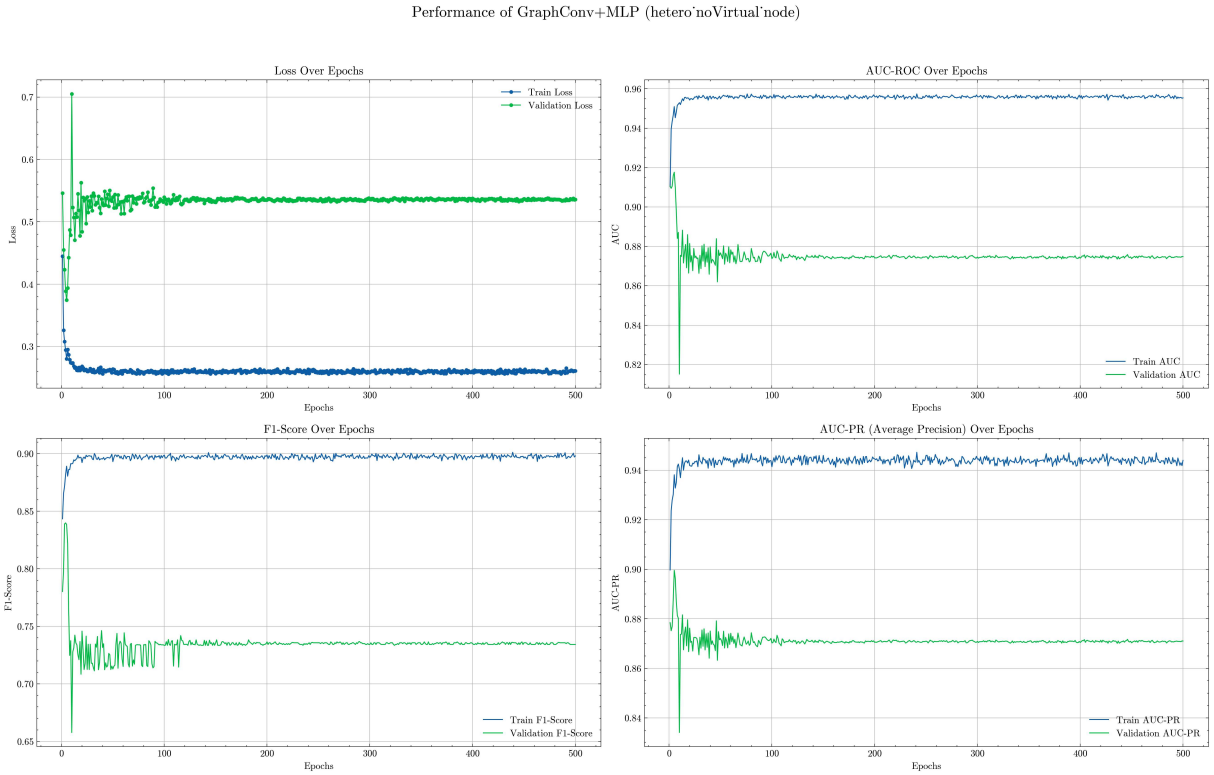
APPENDIX B – TRAINING DETAILS FOR HETEROGENEOUS GRAPH MODELS

This appendix provides supplementary material on the training performance of the Graph Neural Network models evaluated on the heterogeneous graph. The following figures illustrate the learning dynamics by plotting validation accuracy, F1-score, and loss over the training epochs. These visualizations correspond to the two main experimental configurations discussed in the main text: one using the full feature set and another without the Gemini-derived features, allowing for a detailed comparison of each model's behavior.



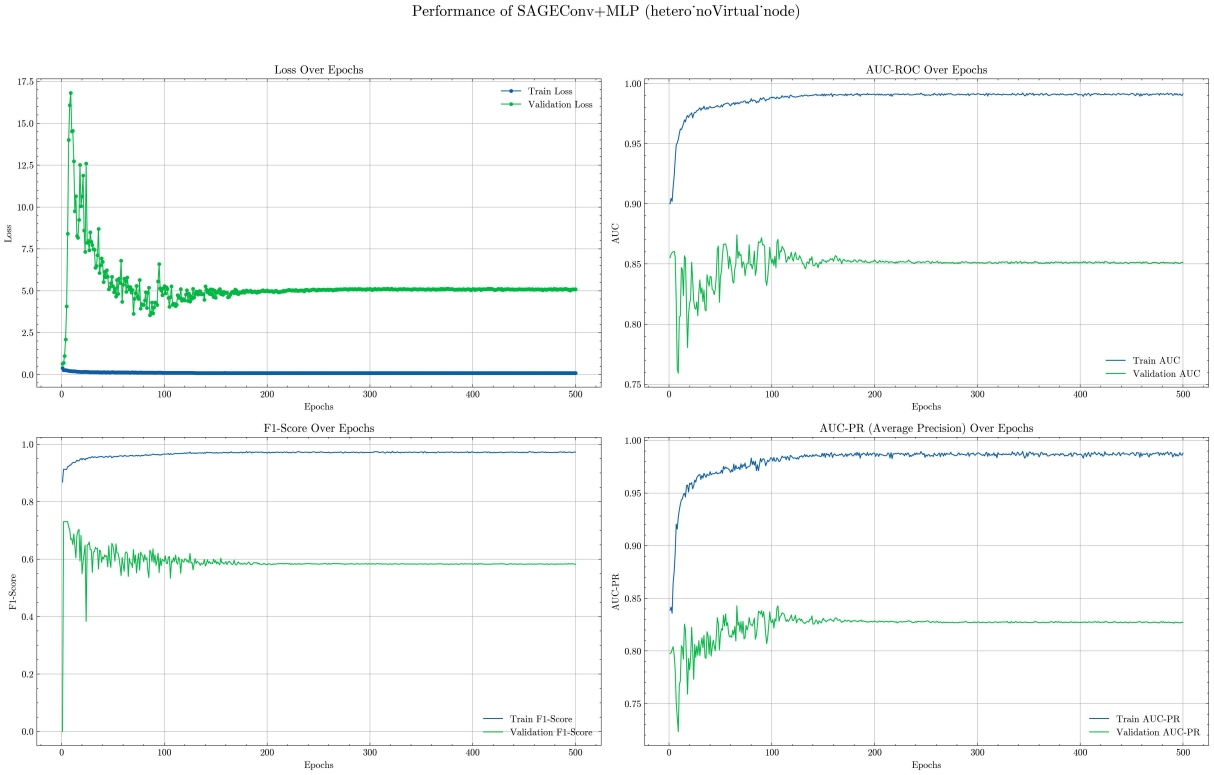
Source: Elaborated by the author (2025).

Figure 27 – Comparison of validation accuracy and F1-score across GNN models on the heterogeneous graph without a virtual node.



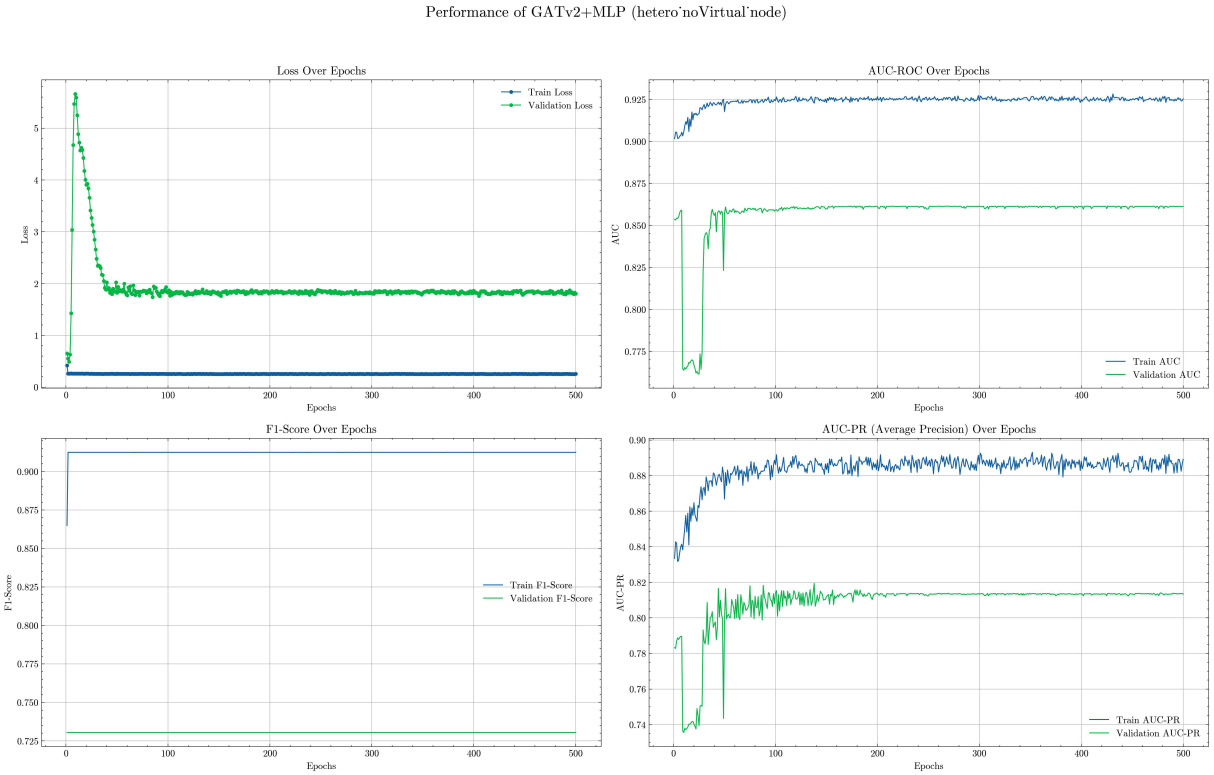
Source: Elaborated by the author (2025).

Figure 28 – Performance metrics (accuracy, F1-score, loss) over training epochs for the GraphConv+MLP model on the heterogeneous graph without a virtual node.



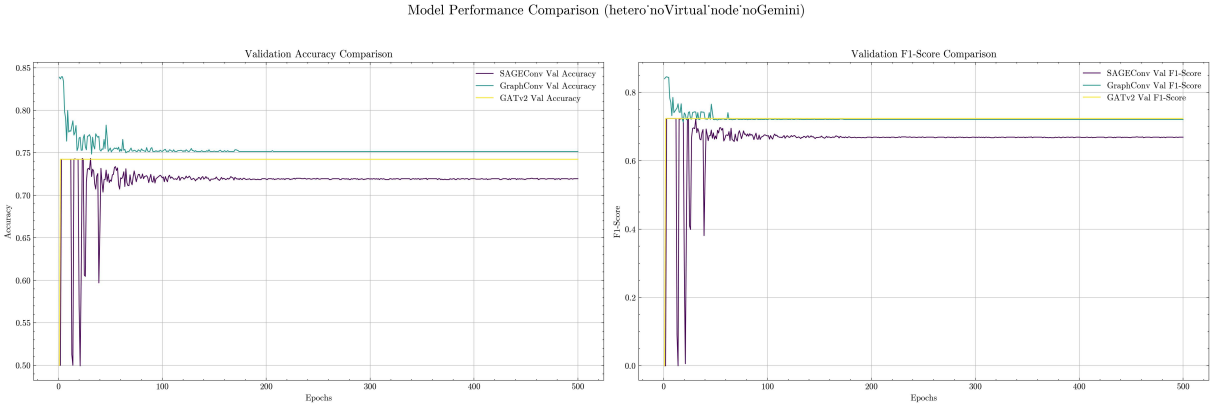
Source: Elaborated by the author (2025).

Figure 29 – Performance metrics (accuracy, F1-score, loss) over training epochs for the SAGEConv+MLP model on the heterogeneous graph without a virtual node.



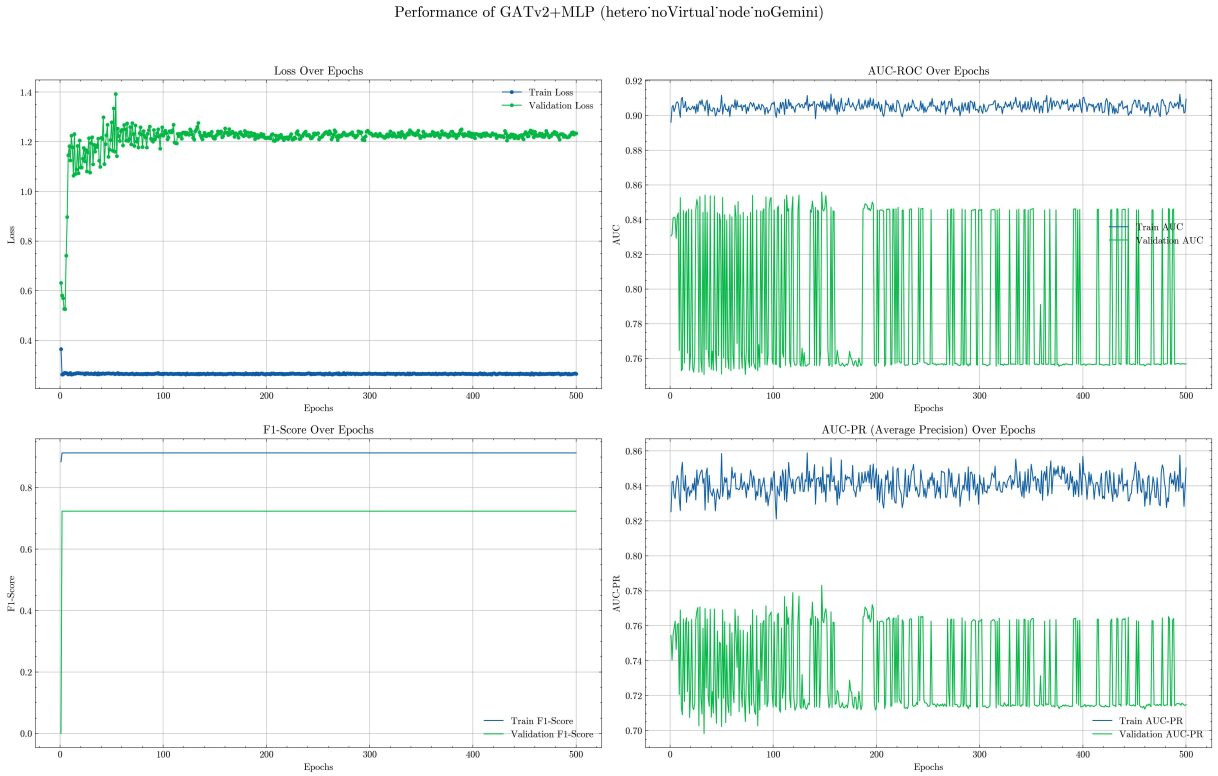
Source: Elaborated by the author (2025).

Figure 30 – Performance metrics (accuracy, F1-score, loss) over training epochs for the GATv2+MLP model on the heterogeneous graph without a virtual node.



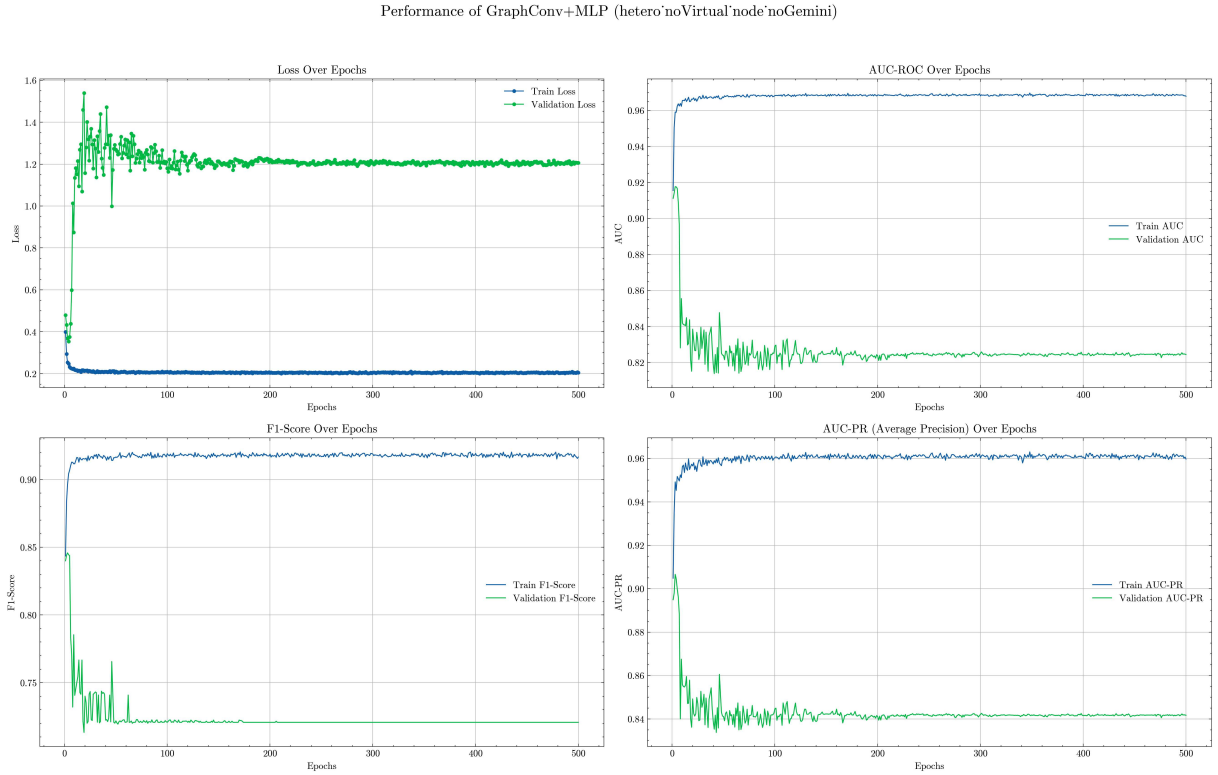
Source: Elaborated by the author (2025).

Figure 31 – Comparison of validation accuracy and F1-score across GNN models on the heterogeneous graph without a virtual node, excluding Gemini features.



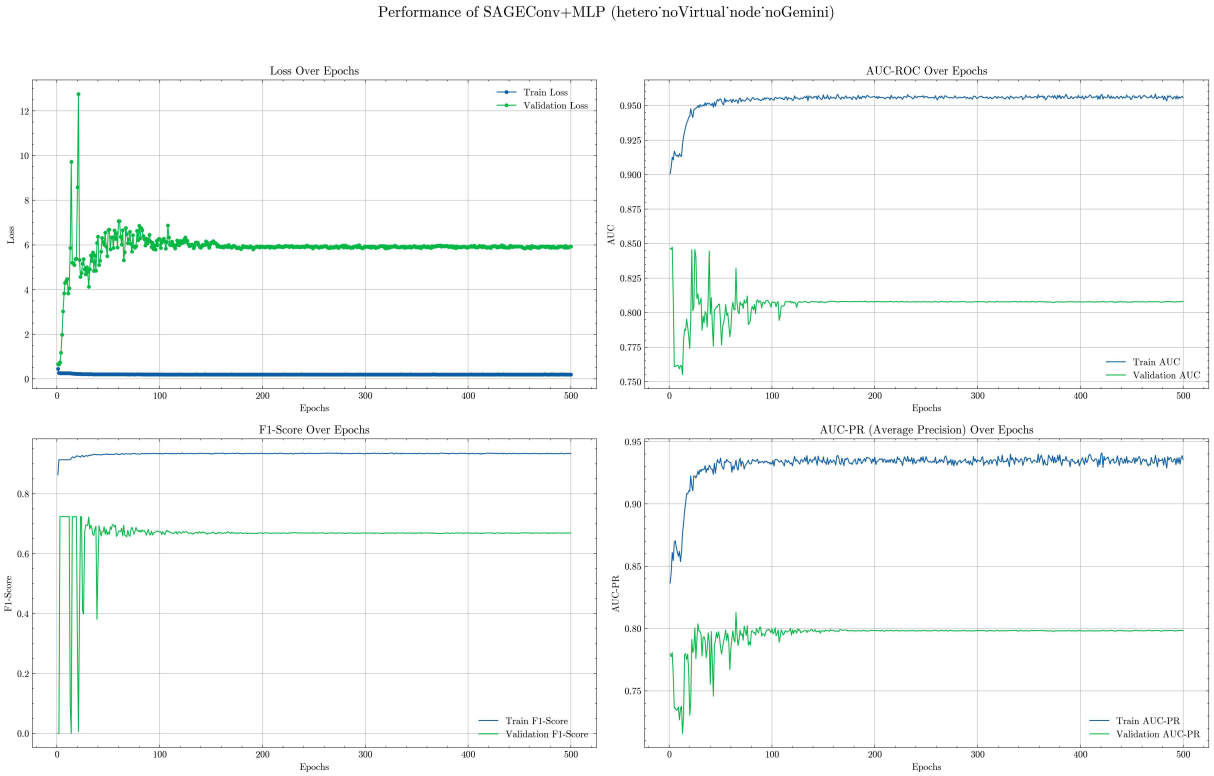
Source: Elaborated by the author (2025).

Figure 32 – Performance metrics for the GATv2+MLP model on the heterogeneous graph without a virtual node, excluding Gemini features.



Source: Elaborated by the author (2025).

Figure 33 – Performance metrics for the GraphConv+MLP model on the heterogeneous graph without a virtual node, excluding Gemini features.



Source: Elaborated by the author (2025).

Figure 34 – Performance metrics for the SAGEConv+MLP model on the heterogeneous graph without a virtual node, excluding Gemini features.