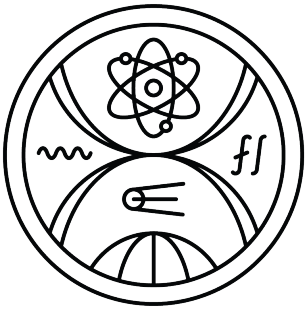
COMENIUS UNIVERSITY IN BRATISLAVA
FACULTY OF MATHEMATICS PHYSICS AND INFORMATICS

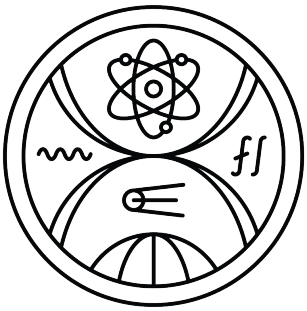


GRAPH NEURAL NETWORKS FOR PLASMID IDENTIFICATION

Master thesis

2025 Bc. Monika Buchalová

COMENIUS UNIVERSITY IN BRATISLAVA
FACULTY OF MATHEMATICS PHYSICS AND INFORMATICS



GRAPH NEURAL NETWORKS FOR PLASMID IDENTIFICATION

Master thesis

Study program:	Applied informatics
Branch of study:	Applied informatics
Department:	Department of Applied Informatics
Supervisor:	doc. Mgr. Tomáš Vinař, PhD.

Bratislava, 2025 Bc. Monika Buchalová

Comenius University Bratislava
Faculty of Mathematics, Physics and Informatics

THESIS ASSIGNMENT

Name and Surname:	Bc. Monika Buchalová
Study programme:	Applied Computer Science (Single degree study, master II. deg., full time form)
Field of Study:	Computer Science
Type of Thesis:	Diploma Thesis
Language of Thesis:	English
Secondary language:	Slovak

Title: Graph Neural Networks for Plasmid Identification

Annotation: Plasmid identification in short-read sequencing data sets is an important and challenging tasks.

Aim: The goal of the thesis is to explore application of various types of graph neural networks to this task.

Supervisor:	doc. Mgr. Tomáš Vinař, PhD.
Department:	FMFI.KAI - Department of Applied Informatics
Head of department:	doc. RNDr. Tatiana Jajcayová, PhD.
Assigned:	05.12.2024

Approved: 05.12.2024 prof. RNDr. Roman Ďurikovič, PhD.
Guarantor of Study Programme

.....
Student

.....
Supervisor

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Monika Buchalová
Študijný program: aplikovaná informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: anglický
Sekundárny jazyk: slovenský

Názov: Graph Neural Networks for Plasmid Identification
Grafové neurónové siete na identifikáciu plazmidov

Anotácia: Identifikácia plazmidov v sekvenačných dátach s krátkymi čítaniami je dôležitou úlohou.

Cieľ: Cieľom práce je preskúmať možnosti aplikácie rôznych typov grafových neurónových sietí na túto úlohu.

Vedúci: doc. Mgr. Tomáš Vinař, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: doc. RNDr. Tatiana Jajcayová, PhD.

Dátum zadania: 05.12.2024

Dátum schválenia: 05.12.2024 prof. RNDr. Roman Ďurikovič, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Prehlásenie

. . . . . .

Bratislava, 2025 Bc. Monika Buchalová

Acknowledgement

Abstract

The goal of this thesis is to apply attention mechanisms to the problem of contig classification in assembly graphs using the plASgraph2 architecture. A plasmid is a small, circular DNA segment that resides freely in the cytoplasm of bacteria. Plasmids are a major factor in the spread of antibiotic resistance between bacterial cells. However, identifying plasmids from short-read sequencing data is a challenging task. This work focuses on integrating Graph Attention Networks (GAT) into a graph neural network that classifies contigs as plasmid, chromosome, or ambiguous. The thesis includes an analysis of existing approaches, adaptation of the codebase for compatibility with updated software libraries, and preparation of an experimental framework for further development.

Keywords: neural networks, graph neural networks, plasmids, DNA

Abstrakt

Cieľom tejto diplomovej práce je aplikovať mechanizmus pozornosti (attention) na problém klasifikácie kontigov v assembly grafoch pomocou architektúry plASgraph2. Plazmid je malý kruhovito stočený úsek DNA, ktorý sa nachádza voľne v cytoplazme baktérií. Práve plazmidy sú častou príčinou šírenia antibiotickej rezistencie medzi baktériami. Identifikácia plazmidov z krátkych sekvenčných dát je však náročná. V práci sa preto zameriavame na využitie Graph Attention Networks (GAT) ako súčasti grafovej neurónovej siete, ktorá rozhoduje, či daný kontig pochádza z plazmidu, z chromozómu alebo má nejednoznačný pôvod. Súčasťou práce je analýza existujúcich riešení, aktualizácia softvérového kódu pre nové verzie knižníc a príprava experimentálneho prostredia pre ďalší výskum.

Kľúčové slová: neurónové siete, grafové neurónové siete, plazmidy, DNA

Contents

List of Figures

List of Tables

Introduction

Chapter 1

Background and problem statement

1.1 Bacterial DNA

DNA, or deoxyribonucleic acid, is the molecule that carries genetic information for all living organisms. Inside the cell, DNA is usually arranged in a double helix, where the two strands run in opposite directions because of their phosphodiester bonds. Even though the double helix is extremely thin, it can be incredibly long—much longer than the cell itself—so it stays tightly coiled and packed inside.

Prokaryotes, which are organisms without a nucleus, typically have just one DNA molecule that is essential for their survival. This DNA is found freely in the cytoplasm, and its ends are usually connected, forming a circular shape. Eukaryotes, on the other hand, are organisms that do have a nucleus. Most of their DNA is stored inside the nucleus as several separate molecules that usually have free ends.

Besides their main chromosomal DNA, many organisms also have smaller DNA molecules. These can be found in certain cell organelles or as short circular pieces of DNA in the cytoplasm called plasmids.

1.1.1 Plasmids

Plasmids are double-stranded circular molecules of DNA. They are able to replicate in the cells of the host independently from chromosomal DNA. Simple example of the plasmid can be seen on the figure Fig. 1.1 They can be found in various types of bacteria and other lower organisms such as yeast. They come with various lengths. Some shorter plasmids can only contain few thousand bases. The longer types can be hundreds of thousands of bases long. Plasmids do not contain genes that are necessary for the proper function of the cell; however, they can provide various advantages that can be utilized in different types of environments. Most genes carry resistance against various types of antibiotics and heavy metals. There are also genes, that are used for encoding bacteriocin production, genes for the breakdown of complex organic

substances, restriction-modification systems, or virulence genes.

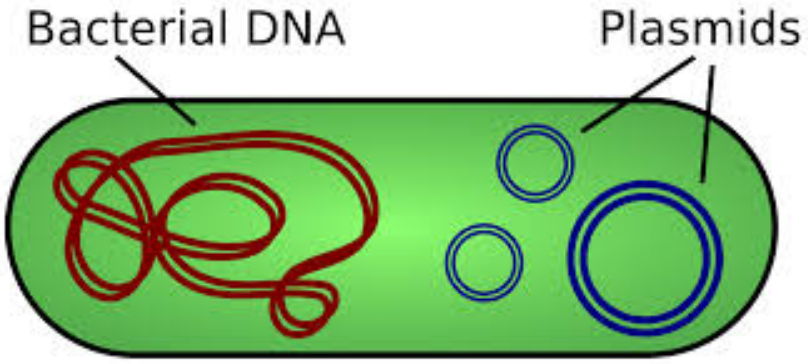


Figure 1.1: This picture shows a bacterial cell with its main bacterial DNA and several plasmids inside it. The large, tangled red structure represents the bacterial chromosome, which contains most of the genetic information. The smaller blue circular structures represent plasmids, which are extra DNA molecules that can replicate independently from the main chromosome.

Plasmids replicate independently from bacterial chromosomes, that means, every plasmid is separate replicant. Gene sequences that are used for enzymes production can be usually found on bacterial chromosomes. These enzymes are necessary for plasmid replication. These enzymes coding sequences can be also sometimes found on plasmid itself. The start of replication also determines the range of hosts for a given plasmid. Some plasmids are able to replicate only in specific types of bacteria; others can replicate in majority of known types of bacteria.

Not all types of plasmids can coexist in the same bacteria. It is called plasmid incompatibility and is defined as the inability of two different plasmids to exist in the same cell without selection pressure. The replication origin determines this property, i.e., two plasmids with the same origin cannot coexist in one cell.

By their ability to transfer between host cells, plasmids can be conjugative or non-conjugative. Conjugative plasmids are capable of independent transfer from the donor to the recipient cell. Transfer is ensured by transfer genes that are encoded on the plasmid. Non-conjugative plasmids are not capable of transfer between cells. Some plasmids can also be mobilizable, that means, that they do not contain their own transfer genes but only a mob region that ensures that they are capable of transfer from donor to recipient cells if conjugative plasmids are present in the cells.

Stability is an important feature of the plasmids. They are able to stably transfer to daughter cells. Lot of low-copy plasmids, which are plasmids that have 1-2 copies to one bacterial chromosome, contain genes and specific sites in their DNA that are responsible for the distribution of copies to both daughter cells. These specific parts responsible for the transfer are called partitioning loci. On the other hand, high-copy

plasmids, which are plasmids that have 20 or more copies, do not need to contain par sites. The reason is, that it is unlikely that a newly formed cell receives no copies of the plasmid during division. However, for distribution to daughter cells, it is necessary for the plasmids to segregate after replication, because they need to be divided into individual molecules.

1.2 Bioinformatical methods for genome processing

In recent years, the development of bioinformatical methods has accelerated significantly. As a result, the way we approach the analysis of genetic information has also changed. From the first DNA sequencing techniques to today's high-capacity sequencing platforms, the methods of obtaining and processing genomic data have undergone a significant evolution. Modern methods allow us to generate huge amounts of sequence data, which require effective computational tools for interpretation. In this chapter, we focus on the historical and technological development of sequencing techniques. In this chapter we also write about two main approaches of genome reconstruction from obtained fragments. First, we will write about read mapping, which is a method that is based on aligning sequences to a reference genome. The second method is de novo assembly, which allows the genome to be assembled without prior knowledge of the reference.

1.2.1 DNA sequencing

DNA sequencing is a laboratory method used to precisely determine the order of nucleotides within a DNA molecule. Technology has evolved through several stages, and today we recognize three generations of sequencing.

Scientists first managed to describe the structure of DNA and capture its image in 1953, but it was not until 1978 that the genome was successfully decoded using Sanger sequencing, a method based on radionuclides. Later, radionuclides were replaced by dideoxynucleosides. The first commercial automated sequencer was based on Sanger sequencing, which was able to obtain around 1000 base pairs. This marks the first generation of sequencing. In the mid-1990s, the second generation of sequencing, otherwise known as next generation sequencing (NGS) or massive parallel sequencing (MPS), began. During this period, scientists were able to sequence whole human genome using massively parallelized sequencing. Third generation sequencing began mid-to-end of 2000s. Unlike in the first generation, in the third generation we can obtain up to 10,000 base pairs from a single read. Each generation has its advantages and disadvantages. Right now, all three generations are used in genome sequencing, based on the specifics of the sequencing experiment.

Currently, no sequencer can sequence the whole genome. Instead, it yields smaller parts, which are called reads. These reads have different lengths and overlap with each other. Because we have large number of reads, several reads can cover the same position in genome. Today, we have two methods that are used to obtain back the original genome: mapping and assembly.

First, we will talk about assembly. After sequencing we have multiple reads with different lengths. During assembly, we put together reads according to overlapping sequences. For example in Fig. 1.2 we have 3 DNA fragments CGAGATTAGCTGCA, TGCATAGCGATATCA and TATCATGATTA. The first fragment and the second fragment share the sequence TGCA and the second fragment and the third fragment share the sequence TATCA. Joining these DNA fragments gives us the resulting local DNA sequence CGAGATTAGCTGCATAGCGATATCATGATTA. Overlaps are written only once in the resulting sequence. Using this method, we are often unable to assemble the entire genome. Obtained local genome sequences are stored in a FASTA file. Each sequence starts with a one-line description followed by lines of sequence data.

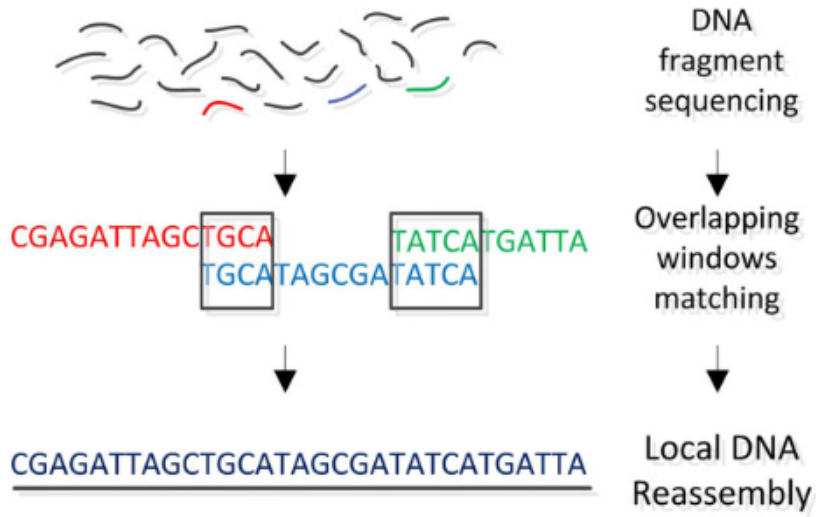


Figure 1.2: At the start of the assembly process is set of reads. During the assembly process we put together reads according to to overlapping sequences. At the end of the process, we have obtained multiple local genomes, that are stored in FASTA files.

Another method to obtain the human genome is mapping. Unlike assembly, mapping uses a reference genome sequence to obtain the genome of interest from reads. As we can see in Fig 1.3 at the start of this process, we have various reads with different lengths, that we obtained from sequencing and the reference genome of an organism whose genome we are trying to obtain. During the process, each read is compared to a different part of a reference genome to find the best fit. As we can see on the bottom of Fig 1.3, Read1 matches in 9 out of 11 bases, Read2 matches in 7 out of 14 bases and Read3 matches in 10 out of 12 bases. These mismatches are caused by various types of

mutations. At the end of the process, we have a BAM file that contains all the locally find genomes. These BAM files are later used in various bioinformatic analyzes.

The most common mutations are substitutions, insertions, and deletions. We are talking about deletion, when one or more nucleotides are missing from a sequence. For example, if we know that at the exact position in the reference genome is sequence AATG, but our sequence from our genome is AAG. We talk about substitution when two nucleotides are switched. As we can see in Fig 1.3, where in Read1, for example, nucleotide C in the reference genome was switched to nucleotide A. We talk about an insertion if a nucleotide or a short sequence of nucleotides is added to the sequence. Again, in Fig 1.3, we can see that the short sequence of nucleotides CC was inserted into Read3. Thanks to these mutations, organisms can adapt to various environmental changes, but they can also be a cause of various dangerous genetic diseases.

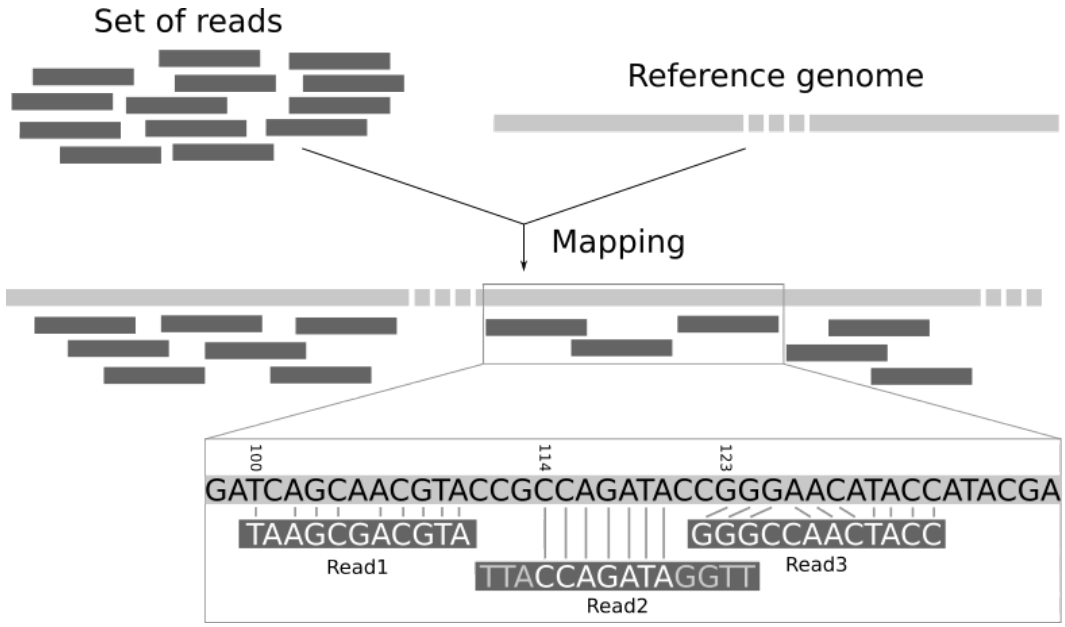


Figure 1.3: At the start of the mapping process is set of reads and reference genome. During the mapping process, each read is compared to each sequence of the reference genome. At the end of the process, each read is placed into position with the highest amount of matching bases.

1.3 State of the art

1.3.1 RFPlasmid

1.3.2 Platon

1.3.3 PlasForest

1.3.4 PlasClass

1.3.5 PlasMAAG

1.3.6 Limitations of the current approaches

1.4 Problem statement

Chapter 2

plASgraph2 and its limitations

Plasmids, as described in Chapter 1.1.1, are small double-stranded circular DNA molecules that can be found in the cytoplasm of a wide variety of microorganisms. They are independent of chromosomal DNA and are able to transfer between bacterial cells. Because of this mobility, plasmids allow microorganisms such as bacteria to adapt to many different environments. They are also the main reason why antibiotic resistance can spread so effectively among bacteria. Detecting plasmid sequences in short-read assemblies is therefore important for public health. As explained in Chapter 1.2.1, we are still not able to sequence an entire genome directly. During sequencing, the genome of an organism is split into many short fragments, which are later combined into contigs using assembly or mapping methods. However, these contigs do not contain enough information on their own to reliably determine whether they originate from chromosomal or plasmid DNA.

For this reason, the tool plASgraph2 was developed. In contrast to the tools described in Chapter 1.3, plASgraph2 does not analyze each contig independently and does not require comparison with a database of known plasmids. Instead, it uses the structure of the assembly graph to determine whether a contig belongs to a chromosome or a plasmid. In this graph, contigs are connected to each other based on assembly information. plASgraph2 characterizes each contig using several features that are known to distinguish plasmids from chromosomes: read coverage (used as a proxy for copy number), GC content, and contig length. In addition to these established features, it introduces two new ones: the node degree in the assembly graph and the similarity between the contig's k-mer profile and the k-mer profile of the entire assembly. Although these features provide useful information, they are not sufficient by themselves to classify a contig correctly. The connections between contigs offer additional context that helps during classification. [2]

Since traditional neural networks are not well suited for working with graph-structured data, plASgraph2 incorporates Graph Neural Networks (GNNs) into its architecture.

GNNs are able to pass information between connected contigs through a process called message passing. Thanks to this ability, plASgraph2 can take advantage of the structure of the assembly graph and make accurate decisions even in cases where the features of a single contig are not sufficient for correct classification.

2.1 Graph Neural Networks

GNNs, or Graph Neural Networks, are a type of network used when we have data structured as graphs. A graph is a structure made of nodes and edges, which represent objects and the relationships between them. Simple representation of GNNs can be seen on Fig. 2.1. A lot of systems in the real world can be represented as graphs; for example, social networks, telecommunication networks, 3D meshes, biological networks, brain connectomes, and so on. [3] Usually, we use neural networks on data that has a regular format, such as text or images. However, graphs do not have a fixed shape or order, so traditional neural networks struggle with these kinds of structures. This is the reason why Graph Neural Networks were created.

The main idea of GNNs is that each node in the neural network should learn based on the information contained in the nodes that are connected to it. At the start of the training, each node contains its initial information, which we call features. These features change during training. In every GNN layer, a node collects information from its neighbors. This process is called "message passing". This means that nodes exchange information with the nodes to which they are linked. After that, they update their own representation based on the information they have received. This process is repeated in multiple layers. Between each layer, some type of activation function, such as ReLU, is used to learn more complex relationships. Without using any type of activation function, the entire neural network would become a large linear transformation, and no complex relationships would be learned. The simple example of this process can be seen in Figure 1.4. At the end of the process, each node does not capture only its own properties. It also captures the context provided by the surrounding parts of the graph. [3]

The most popular neural network in GNN is the Graph Convolutional Network. This network works almost the same way as convolutional neural networks used for image processing. Convolutional neural networks used for image processing look at nearby pixels. However, graph convolution looks at nearby nodes in a graph. This property allows the model to learn patterns that are based not only on the features of the nodes. They are also based on how those nodes are connected. Thanks to this property, GNNs can detect structures and relationships that traditional models would miss because these structures are too complex to capture.

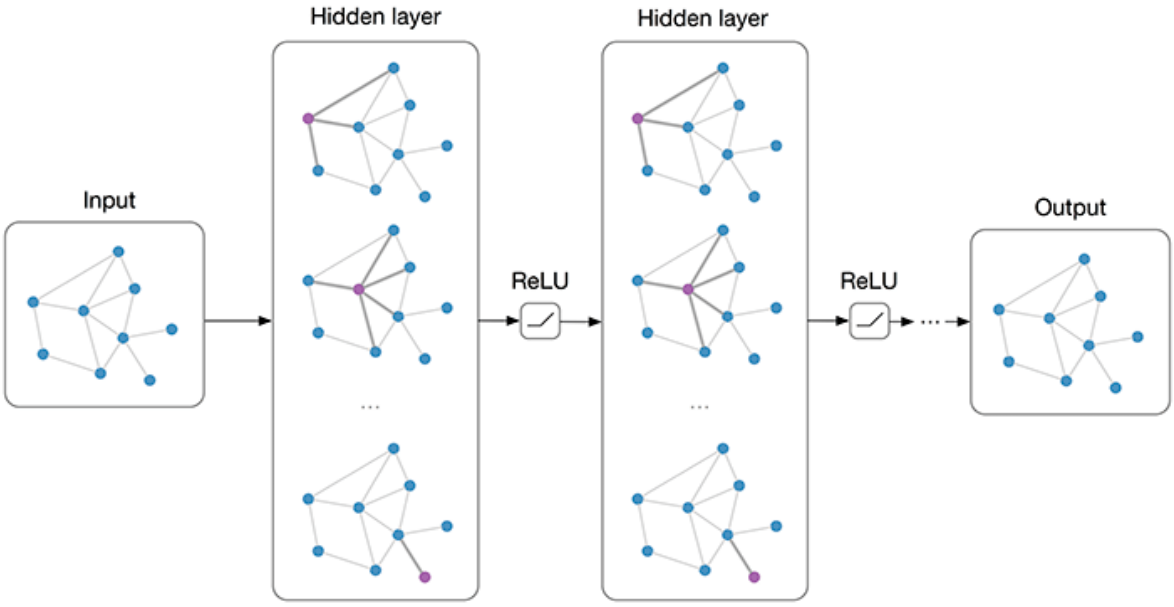


Figure 2.1: Graphical representation of training in Graph Neural Networks using ReLu activation function to learn complex relationships between each layer.

We can use GNNs for many tasks. The big advantage is, that they can classify individual nodes. For example, they can predict whether a sequence of bases belongs to a plasmid or to a bacterial chromosome. Also, they can classify entire graphs. They are able to decide whether a molecule is toxic or non-toxic. The most important property of GNNs is that they can work on incomplete types of graphs. For example, during training, we can hide some parts of the graph. The result of this is that the neural network was not able to gain all the knowledge about the data. Besides this disadvantage, during testing, GNNs are able to predict the correct answers with high accuracy. In general, we are using GNNs to solve problems that involve data where the relationships between each item are as important as the individual items themselves.

Graph neural networks are a powerful tool that allows us to solve more complex problems that traditional neural networks would not be able to handle. However, they also have their limitations. These neural networks can be very computationally demanding, especially in problems that are highly complex and require a deep network to capture all the important relationships between nodes. The accuracy and computational cost of these networks also depend on how well the graph is constructed. Despite these challenges, graph neural networks are increasingly being used to address various biological problems.

2.2 Input features of plASgraph2

At the start of the whole process is the assembly graph of bacterial isolates, where nodes represent contigs and edges represent adjacencies supported by sequencing data. Each contig in this graph contains six features.

The first feature is node degree. This feature counts how many neighbors a contig has in the assembly graph. Plasmids often have a different graph structure compared to chromosomes, so this information is useful. The second feature is normalized contig length. The contig length is divided by a constant (two million). This prevents very large values from dominating the model. The third feature is the logarithm of contig length. This feature helps the model distinguish between long and short contigs. The fourth feature that describes each contig is Relative GC content. The GC content of the contig is compared to the average GC content of the entire sample. This makes the feature less dependent on which bacterial species is being analyzed. The Second-to-last feature is relative coverage. The coverage of a contig is divided by the weighted median coverage of the sample. Plasmids often have higher coverage than chromosomes because they are present in multiple copies. The last feature is relative pentamer composition. The relative pentamer content compares the 5-mer (pentamer) composition of a contig with the 5-mer composition of the entire assembly. For each contig or set of contigs, the number of occurrences of every pentamer and its reverse complement is used to form a normalized pentamer profile, where a small pseudocount $\varepsilon = 0.01$ is added to avoid zero values. The similarity between the pentamer profile of the contig and that of the whole assembly is then expressed as the dot product of the two profile vectors. This produces a single value that represents how similar the sequence composition of the contig is to the sequence composition of the entire assembly. This process again provides a species-independent feature. [2]

Using relative features allows the model to generalize across different bacterial species. It also helps to avoid dependence on species-specific values. For example, if the raw GC content were used as a feature, the model could learn that chromosomal sequences tend to have one characteristic GC content, while plasmid sequences tend to have another. However, this information is not a universal property found in different types of organisms because GC content varies widely across taxa. On the other hand, using relative GC content allows the model to learn that the GC content of chromosomal contigs is usually closer to the overall GC content of the sample because chromosomal DNA is longer than the DNA of the plasmid. Using relative features also allows the model to realize that the GC content of the plasmids usually deviates from this value. [2]

Using relative features allows the model to be trained on multiple species, not only on one specific species.

2.3 Model architecture

The architecture of plASgraph2 is designed that way that model is able to label each contig in an assembly graph as plasmid, chromosome, or ambiguous. As we can see in Fig 2.2 the model consists of input encoding at the start, several graph convolutional layers used for training and final prediction module.

2.3.1 Input Encoding

Each contig is described by six input features, that are described in more details in Chapter 2.2. Before entering the graph neural network, these features are transformed by a dense layer with a sigmoid activation function.

The sigmoid activation function is a mathematical function that takes any real-valued number and transforms it into a value between 0 and 1. Values close to 1 indicate high probability, while values close to 0 indicate low probability.

At the end of this first step we end up with a $10 \times n$ representation, where n is the number of contigs in the assembly graph. The output of this encoding stage is then processed by two dense layers with ReLU activation.

The ReLU (Rectified Linear Unit) activation function outputs the input value when it is positive and outputs zero when the input is negative. This simple form introduces non-linearity into the model, which allows the network to learn more complex relationships between features. At the same time, ReLU is computationally efficient and helps reduce issues such as the vanishing gradient during training.

The first dense layer produces a $32 \times n$ representation. This representation is then used as the main input to the graph convolutional layers. The second layer produces a $32 \times n$ node identity vector. This vector preserves the original contig identity throughout the network. Both the main input and the node identity representation are reused in each of the six graph convolutional layers. [2]

2.3.2 Graph Convolutional Layers

The core of the model consists of six identical graph convolutional (GCN) layers. All GCN layers share the same parameters, which means that the same transformation is applied repeatedly as information propagates through the graph. Each layer contains several steps. First, dropout (10%) is applied to the $32 \times n$ main input. A GCN layer with a ReLU activation function is then applied. This layer takes the current $32 \times n$ representation together with the $n \times n$ adjacency matrix and performs message passing between neighboring contigs. Afterwards, the updated node features are concatenated with the preserved node identity vector, resulting in a $64 \times n$ representation. Finally,

this $64 \times n$ representation is passed through a dense layer that transforms it back to $32 \times n$. The output of this dense layer becomes the input to the next GCN layer. [2]

After the first round of this process, each node has incorporated information from its direct neighbors, and its feature vector is updated according to this information. After the second round, each node also receives information from the neighbors of its neighbors, and so on. This design allows each contig to accumulate information from increasingly distant regions of the assembly graph while preserving its original identity.

2.3.3 Output Layer

After the last round of computation, the final dense layer uses a sigmoid activation function to produce two independent scores for each contig, using the information obtained during the training process. First, it produces plasmid score and then it produces chromosome score.

These scores do not need to sum to one. A contig may therefore receive a high score for both classes (ambiguous), low scores for both classes (unlabeled), or a high score for only one class (plasmid or chromosome). This behavior reflects the biological reality that some contigs share characteristics of both plasmids and chromosomes, or may not contain enough information to be classified with high confidence. [2]

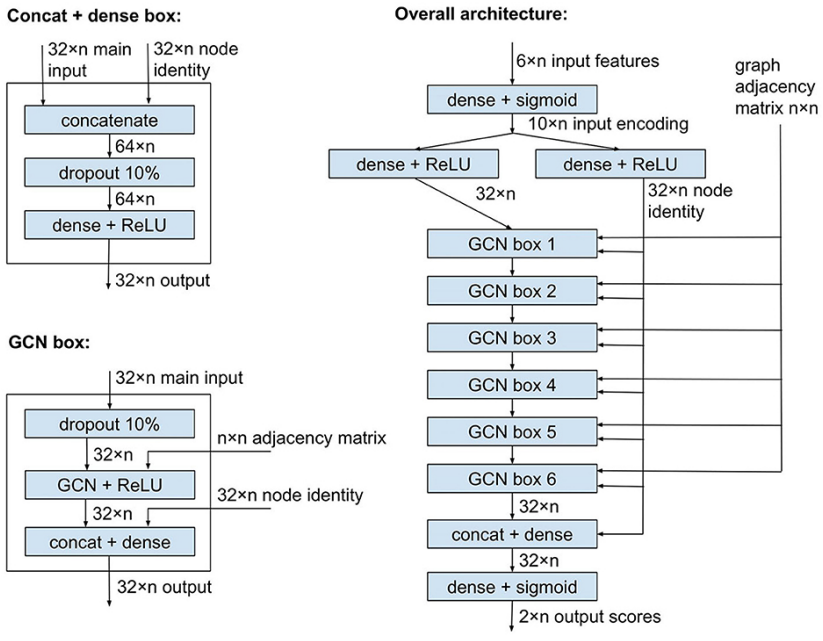


Figure 2.2: Model architecture of plASgraph2. The model takes as input the assembly graph structure and six features per node (contig). The core of the network is composed of six graph convolutional layers. The model generates two outputs per node, which facilitate the classification of plasmids and chromosomes as two separate classification tasks.

2.4 Training

After preparing the training and testing data sets, the model was trained using the assembly graphs and the contig features described above. Each assembly graph was treated as an independent training sample, meaning that a single batch contained the full graph of one isolate. For optimization, the model used the Adam optimizer with a learning rate of 0.005. Training was run for a maximum of 1000 epochs with a patience value of 100 for early stopping, so the training stopped automatically once the validation loss no longer improved. During training, the data set was split into 80% training samples and 20% validation samples. This allowed the model to be monitored for overfitting while ensuring that the validation graphs remained unseen during training. All six graph convolutional layers shared the same parameters. This design helped reduce the total number of parameters and made the model more robust. At the end of training, the output of the model consisted of two independent sigmoid scores for each contig: a plasmid score and a chromosome score. Before applying the model to new assemblies, the authors performed an additional threshold optimization step. The aim of this step was to choose score thresholds that maximized the F1 score on the validation set. These thresholds were then used when classifying contigs in the testing data and in external samples. [2]

2.5 Results

The performance of plASgraph2 was evaluated on several data sets and compared with a set of existing plasmid-detection tools, which are described in Chapter 1.3, namely PlasClass, Platon, 3CAC, and PlasForest. The main evaluation metrics were accuracy, AUROC, and F1 score. The performance of the model was evaluated separately for chromosomal and plasmid contigs.

As we can see in Fig 2.3, for the ESKAPEE test samples, plASgraph2 achieved the highest median F1 score for plasmid identification among all tested tools. It also reached the best median AUROC and accuracy for plasmid classification. For chromosomal contigs, the performance of plASgraph2 was close to that of Platon, which uses homology-based information and, therefore, tends to perform very well on long chromosome-derived contigs.

An important observation was that plASgraph2 performed especially well on short contigs. This is a challenging region where many homology-based tools struggle because short sequences often lack meaningful alignment information. In contrast, plASgraph2 benefits from the structure of the assembly graph, which provides additional context.

To investigate whether the model can generalize beyond the species represented in the training set, the authors also tested plASgraph2 on isolates that were not part

of the ESKAPEE group but were still closely related. The model retained strong performance, showing that the use of relative features and graph structure helps the model work well even in species that were not directly used for training.

When evaluating very distantly related species, the performance decreased, which is expected because plasmids and chromosomes may share similar sequence characteristics outside the ESKAPEE clade. Nevertheless, plASgraph2 remained competitive with the other methods in these tests.

The authors also examined the predictions directly on the assembly graphs. In comparison with tools such as PlasForest, which sometimes produce isolated classification errors scattered across the graph, the predictions of plASgraph2 tended to appear in coherent regions. Errors were usually found in small clusters, suggesting that information passed through the graph helped maintain consistent predictions across neighboring contigs.

Method	SS	DB	AUROC	Precision	Recall	F1	Accuracy
A: Plasmid classification, contigs >100 bp, n =38,110							
plASgraph2	–	–	**0.991**	0.906	0.908	**0.808**	**0.935**
mlplasmids	X	–	0.896	0.273	**0.957**	0.480	0.641
PlasClass	–	–	0.892	0.381	0.939	0.617	0.794
PlasForest	–	X	n/a	0.486	0.939	0.711	0.852
Platon	–	X	n/a	**1**	0.5	0.667	0.924
Deeplasmid	–	X	n/a	n/a	n/a	n/a	n/a
RFPlasmid	X	X	0.973	0.854	0.789	0.667	0.885
B: Chromosome classification, contigs >100 bp, n =38,110							
plASgraph2	–	–	**0.991**	0.975	**1**	0.968	0.943
mlplasmids	X	–	0.908	**1**	0.540	0.697	0.609
PlasClass	–	–	0.878	**1**	0.738	0.840	0.766
PlasForest	–	X	n/a	0.992	0.771	0.855	0.795
Platon	–	X	n/a	0.957	**1**	**0.973**	**0.952**
Deeplasmid	–	X	n/a	n/a	n/a	n/a	n/a
RFPlasmid	X	X	0.959	0.982	0.936	0.933	0.893
C: Plasmid classification, contigs >1,000 bp, n =15,687							
plASgraph2	–	–	0.997	0.960	0.933	0.852	0.946
mlplasmids	X	–	0.974	0.526	**1**	0.783	0.864
PlasClass	–	–	0.986	0.75	**1**	0.857	0.929
PlasForest	–	X	n/a	0.824	0.944	0.835	0.927
Platon	–	X	n/a	**1**	0.836	**0.897**	**0.961**
Deeplasmid	–	X	0.929	**1**	0.333	0.5	0.892
RFPlasmid	X	X	**0.998**	0.914	0.926	0.862	0.942
D: Chromosome classification, contigs >1,000 bp, n =15,687							
plASgraph2	–	–	**0.996**	0.976	**1**	0.969	0.951
mlplasmids	X	–	0.966	**1**	0.845	0.906	0.860
PlasClass	–	–	0.972	**1**	0.897	0.936	0.904
PlasForest	–	X	n/a	**1**	0.919	0.936	0.902
Platon	–	X	n/a	0.989	**1**	**0.983**	**0.973**
Deeplasmid	–	X	0.911	0.903	**1**	0.935	0.893
RFPlasmid	X	X	0.987	**1**	0.954	0.954	0.931

The table shows the median values for each metric from results on all 224 samples included in the testing set. The highest value in each category is shown in bold. SS, method uses Species-Specific models; DB, method uses a DataBase of plasmids and/or chromosomes or other features derived from homology search. Note that, Deeplasmid only allows classification of contigs longer than 1,000 bp. Further, PlasForest and Platon do not provide confidence scores for each prediction. Therefore, calculation of AUROC is not applicable (n/a).

Figure 2.3: In this table we can see the accuracy of plASgraph2 againstother tools

2.6 Limitations

Although plASgraph2 represents a significant improvement in plasmid detection from short-read assemblies, the method has several limitations that affect its performance and generalizability.

A first limitation is the strong dependence on the quality of the assembly graph. The model makes use of the structure of the graph to exchange information across contigs, and this is one of its main strengths. However, in highly fragmented assemblies or in assemblies with misassembled regions, the graph structure may not provide enough reliable information. In such cases, the advantage of using a graph neural network becomes smaller, and the model may behave more similarly to sequence-based tools.

A second limitation concerns very short contigs. plASgraph2 does not classify contigs shorter than 100 bp because these sequences do not contain enough signal to produce meaningful feature values. Although the graph is adjusted by connecting the neighbors of these short contigs, the excluded sequences still represent regions where no prediction is made. In samples that contain many short contigs, this may reduce the overall usefulness of the method.

Another limitation is related to species diversity. The model was trained primarily on ESKAPEE species, and its use of relative features allows it to generalize to closely related species reasonably well. However, when applied to bacterial groups that are phylogenetically distant from the training set, the performance decreases. In such cases, plasmid and chromosome sequences may share similar sequence characteristics, and the relationships learned during training may not transfer effectively.

The model also depends on accurate ground-truth labels during training. Although hybrid assemblies offer much better reliability than short-read assemblies alone, they are still subject to misassemblies or incomplete labeling, especially in complex samples. Any mistakes in the labeling process can propagate into the model and affect its predictions.

Despite these limitations, plASgraph2 demonstrates that incorporating the assembly graph into plasmid detection can substantially increase classification accuracy, particularly for short and ambiguous contigs. The approach shows that graph neural networks can provide meaningful improvements in tasks where both sequence features and structural information are important. However, further work is needed to extend this approach to broader taxonomic groups, handle very short contigs more effectively, and improve robustness to variation in assembly quality.

Chapter 3

Introduction of attention mechanism to plASgraph2

Although plASgraph2 achieves strong performance by combining sequence-derived features with information from the assembly graph, its original architecture treats all neighboring contigs as equally important during message passing. In the graph convolutional layers used by plASgraph2, information from all adjacent nodes is aggregated using fixed, uniform weighting. This assumption may be limiting in biological assembly graphs, where not all connections are equally informative. Some edges represent strong biological relationships, while others may arise from repeats, assembly ambiguities, or sequencing artifacts.

To address this limitation, an attention mechanism was introduced into the plASgraph2 architecture. Attention-based graph neural networks allow the model to assign different importance weights to neighboring nodes during message passing. Instead of averaging information from all neighbors, the model learns which connections are more relevant for the classification task. This is particularly useful in fragmented assemblies, where misleading or weak connections can negatively influence predictions.

In this work, the original graph convolutional layers were replaced with graph attention layers, while keeping the rest of the architecture unchanged. This design choice ensures that the model continues to benefit from the structural information encoded in the assembly graph, while gaining the ability to focus on the most informative neighbors. The goal of this modification is to improve classification performance, especially for ambiguous and short contigs, without increasing reliance on species-specific sequence features or external databases.

3.1 Attention mechanism in Neural Network

Graph neural networks are already very strong models, but their performance can be further improved by adding an attention mechanism. The main idea behind attention is that a node should not treat all of its neighbors as equally important. Instead, the model learns which neighbors contain more relevant information and should therefore contribute more to the node's updated representation. This idea was introduced in Graph Attention Networks [3], where the network learns attention coefficients that determine how much weight each neighbor receives.

To understand how this mechanism works, we start with the input features of each node, denoted as $\mathbf{h}_i$. These features are first transformed through a learnable linear layer using a weight matrix W, which produces new feature vectors $W\mathbf{h}_i$. This step is necessary because it allows the model to map the original features into a space where meaningful interactions between nodes can be learned. During training, the matrix W is adjusted so that the transformed features become more informative.

After the linear transformation, the model computes attention coefficients between every node i and each of its neighbors j. This means that, at the end of this process, each node knows how important its neighbors are. This information is then used during the update of the node's features. The score is computed using a small feed-forward neural network implemented as

$$e_{ij} = \text{LeakyReLU}\left(\mathbf{a}^T \left[W\mathbf{h}_i \,\|\, W\mathbf{h}_j\right]\right),$$

where $\mathbf{a}$ is a learnable weight vector, and $\|$ represents the concatenation of the feature vectors. The LeakyReLU activation is used to introduce non-linearity into the attention computation, and the vector $\mathbf{a}$ is tuned during training allowing the network to discover only meaningful patterns in how the features are related to one another. The graphical representation of this process can be seen on Figure 3.1. [3]

Because the raw attention scores cannot be used as they are, they are normalized with the softmax function. This makes the neighbors' contributions add up to one and turn them into proper importance weights. The normalized coefficients are computed as

$$\alpha_{ij} = softmax_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in N(i)} \exp(e_{ik})}.$$

These coefficients describe how much attention node i gives to each neighbor j. As training progresses, the model modifies the parameters so that neighbors that help the model make correct predictions receive larger α_{ij} values, while less useful neighbors receive smaller ones. In this way, the node learns to "focus" on the most informative parts of the graph.

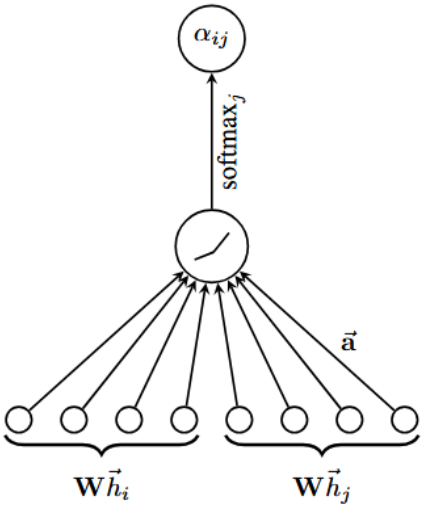


Figure 3.1: This picture shows how a Graph Attention Network computes the attention coefficient α_{ij} between two connected nodes i and j. First, the transformed feature vectors $W\vec{h}_i$ and $W\vec{h}_j$ are concatenated and passed into a small neural network represented by the weight vector $\vec{a}$. This network produces a single score, which then goes through a LeakyReLU activation. After this, the softmax function is applied over all neighbors of node i, producing the final attention coefficient α_{ij}.

Using the attention coefficients, the node features are finally updated by combining the transformed features of the neighbors:

$$\mathbf{h}_i^{\text{new}} = \sigma\left(\sum_{j \in N(i)} \alpha_{ij}\, W\mathbf{h}_j\right),$$

where σ is a non-linear activation function.

To improve stability and expressiveness, GATs use a technique called multi-head attention. Instead of computing the attention mechanism once, the model computes it independently K times. Each head learns its own set of attention coefficients and its own weight matrix W_k. The outputs of these heads are then concatenated in the hidden layers:

$$\mathbf{h}_i^{\text{new}} = \Big\|_{k=1}^{K} \sigma\left(\sum_{j \in N(i)} \alpha_{ij}^k W_k \mathbf{h}_j\right).$$

This allows the model to capture different types of relationships between nodes simultaneously. Each head can focus on different structural or semantic aspects of the neighborhood. In the final layer, the outputs of the attention heads are averaged instead of concatenated, which helps produce stable predictions:

$$\mathbf{h}_i^{\text{new}} = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in N(i)}\alpha_{ij}^k W_k \mathbf{h}_j\right).$$

The multi-head approach, therefore, increases the robustness of the model and provides larger feature representations. Because of all these mechanisms and properties, the flexibility of the attention mechanism combined with the multi-head approach usually provides us with a model that achieves higher accuracy than simpler aggregation methods in graph neural networks. [3]

Chapter 4

Experimental evaluation

Conclusion and discussion

Bibliography

[1] Pau Piera Líndez, Lasse Schnell Danielsen, Iva Kovačić, Marc Pielies Avellí, Joseph Nesme, Lars Juhl Jensen, Jakob Nybo Nissen, Søren Johannes Sørensen, and Simon Rasmussen. Accurate plasmid reconstruction from metagenomics data using assembly-alignment graphs and contrastive learning. *bioRxiv*, pages 2025–02, 2025.

[2] Janik Sielemann, Katharina Sielemann, Broňa Brejová, Tomáš Vinař, and Cedric Chauve. plasgraph2: using graph neural networks to detect plasmid contigs from an assembly graph. *Frontiers in Microbiology*, 14:1267695, 2023.

[3] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.