Detecting operons from RNA-seq data using a convolutional and recurrent neural network architecture

by

A thesis submitted to the School of Graduate Studies

in partial fulfillment of the requirements for the degree of

Master of Science

Supervisor: Dr. Lourdes Peña-Castillo

Department of Computer Science

Memorial University of Newfoundland

September 2023

St. John's Newfoundland

Abstract

Operon is a characteristic of prokaryotic genomes that enables the co-regulation of adjacent genes. Identifying which genes belong to the same operon can help in understanding bacterial gene function and regulation, which can enhance, for instance, drug development and antibiotic resistance inhibition. There are numerous experimental and computational approaches for operon detection; however, many of the computational approaches have been developed for a specific target genome or require specific information only available for a restricted number of bacterial genomes. Here, we develop a novel general method that directly utilizes RNA-seq reads as a signal over nucleotide bases in the genome, extracting all the information from the RNA-seq data. This representation enabled us to employ deep learning techniques without limitations on species. The final model (OpDetect) demonstrates superior performance in terms of recall, f1-score and Area Under Receiver Operating Characteristic curve (AUROC) compared to previous approaches. Additionally, it showcases species-agnostic capabilities, successfully detecting operons even in *Caenorhabditis elegans* (*C. elegans*), the only eukaryotic organism known to have operons.

Acknowledgement

I am deeply grateful to the individuals and organizations who have supported and contributed to my research journey. First and foremost, I extend my sincere appreciation to my supervisor, Dr. Lourdes Peña-Castillo, for her unwavering support, valuable guidance, and exceptional mentorship throughout this endeavour. Her encouragement has been instrumental in shaping the direction of my work. I would also like to thank the School of Graduate Studies at Memorial University of Newfoundland and Labrador for providing me with the Master's fellowship, which has been essential in facilitating my academic pursuits. Additionally, I am grateful to the Digital Research Alliance of Canada for granting me access to their resources, significantly enhancing the efficiency of my research. Last but not least, I would like to acknowledge the collaborative efforts and assistance of Seyed Mohammad Amin Taheri Ghahfarokhi, a fellow master's student in the Bioinformatics Lab. His contributions, brainstorming sessions, and programming advice have been invaluable to the success of this thesis.

Contents

List of Figures

List of Tables

List of Abbreviations

List of Acronyms

Chapter 1

Introduction

In molecular biology, an operon refers to a group of neighbouring genes that are regulated together by one or more overlapping transcription units, all of which are transcribed in the same direction and contain at least one common gene. The regulation of operons is based on transcription units, which are regions of DNA that encompass the area from the promoter, where transcription is initiated, to the terminator, which marks the end of transcription. Genes within an operon are typically functionally related or involved in specific biological processes [1, 2, 3, 4].

Operon detection is the task of identifying genes belonging to the same operon in the genome, which results in mapping the organization of genes and regulatory networks. This will lead to a better understanding of gene functionality in prokaryotic genomes [3, 5]. Moreover, it enables the inference of unknown protein functionalities based on the known functions of co-transcribed genes [6, 7].

Bacteria, among prokaryotic organisms, are of particular interest due to their significant impact on the stability of multicellular organisms and ecosystems, with both beneficial and pathogenic effects. Therefore, comprehending their molecular functions is crucial. Operons play a critical role in bacteria gene regulation and are the functional basis of bacterial genome organization. Accordingly, operons have been identified as targets for drug development and the inhibition of antibiotic resistance [2, 3]. Detecting operons is particularly relevant for unannotated bacterial genomes, as it contributes to a better understanding of their genetic makeup and function [8, 9, 10]. This understanding has broad implications in various fields, including biological, medical, and environmental sciences [11].

Over the years, significant advancements have been made in experimental and computational techniques for operon detection in bacterial genomes. These computational techniques produce high-performance identification mechanisms for vastly studied and annotated bacterial genomes like *Escherichia coli* (*E. coli*) and *Bacillus subtilis* (*B. subtilis*) [7, 12]. However, these methods rely on the accurate annotation of the target genome sequence, which is not available for all bacteria. Therefore, providing a species-agnostic system for prokaryotic genome remains a challenge.

This thesis aims to overcome the challenge of identifying operon structures in diverse bacterial species by developing a generalizable tool for genomic analysis that enables researchers to uncover the functional organization of genes within bacterial genomes, with minimum requirements for processed data input.

The thesis is organized into the following main sections: The Related Work section (Chapter 2) delves into existing computational methods, both traditional and machine learning-based that have been utilized for operon prediction. It discusses previous approaches and identifies the research gap that motivates the need for the current study. The Methodology section (Chapter 3) is dedicated to explaining the process followed in this research. It details the data used for training and evaluation, the features extracted for model input, and the architecture of the machine learning model used. The Results section (Chapter 4) covers the benchmarking process to evaluate the model's performance across different datasets, and how it compares to existing methods. Finally, the Conclusion section (Chapter 5) summarizes our research outcomes and, reiterates the significance of our study's contributions, and suggests future research directions.

Chapter 2

Related Work

Initially, operon detection was performed empirically by human experts. Many operons were detected using wet lab experimental techniques such as RNA polymerase footprinting and primer extension or S1 mapping [7]. Another technique is the manual annotation of operons by biologists; they decide the operon regions and boundaries based on the sequence features and phylogenetic distances [3]. The results of experimental methods are highly accurate, and they have produced experimentally verified datasets of operon-annotated bacterial genomes such as DBTBS[13], Operon DataBase (ODB)[14], RegulonDB[15], and others [12, 16]. However, due to their time and resource-intensive nature and reliance on human judgment, the application of these techniques on a large scale is limited. As a result, computational approaches have been developed as an alternative means to address these constraints.

2.1 Computational Methods - Traditional

The first step in designing computational systems is choosing and extracting the proper features from the genome. There are three main categories of popular features in the literature: Primary Sequence features, External Data Source features, and Transcript Expression features [3, 5, 6, 7, 17, 18].

Primary sequence features include characteristics such as intergenic distance and the presence of transcription signals like promoters or terminators. Intergenic distance, for example, is often shorter within operons and is widely utilized in operon detection studies. External Data Source features, such as functional annotation of the protein products and the conservation of gene sequences across species, are commonly identified using comparative genomics analysis. External data source features mainly rely on the accurate annotation of genomes. This makes these features less suitable for poorly studied genomes. Transcript expression features are based on the idea that the co-expression of genes increases the likelihood of belonging to the same operon. Early studies used the microarray technique; however, the growth of RNA-seq has replaced microarray with RNA-seq data. RNA-seq is a high-throughput sequencing method that provides a comprehensive quantification of RNA molecules in a given sample, offering unprecedented insights into gene expression [19].

Conway et al. [4] demonstrated that the only necessary features for operon detection are transcription signals that bound the operon region (promoters and termina-

tors), and deep coverage[1] of transcript expression data, obtainable using the RNA-seq technique.

Ref	Year	Intergenic Distance	Promoter/ Terminator	Functional Annotation	Expression level by micro-array	Expression level by RNA-seq
[20]	2002	✓	-	-	✓	-
[21]	2002	✓	-	-	✓	-
[22]	2005	✓	-	✓	-	-
[23]	2010	-	✓	-	-	✓
[24]	2014	✓	✓	-	-	✓
[25]	2014	✓	-	✓	-	-
[26]	2018	-	✓	-	-	✓

Table 2.1: Features used in traditional computational methods for operon detection.

As demonstrated in Table 2.1, traditional computational studies have explored various features for operon detection. Sabatti et al. [20] and Tjaden et al. [21] were among the first to investigate the use of transcription information in operon detection. In [22], it is shown how intergenic distance can improve the accuracy of operon detection. Sharma et al. [23] and Slager et al. [26] specifically focused on detecting the operons in the genome of human pathogens, respectively *Helicobacter pylori* (*H. pylori*) and *Streptococcus pneumoniae* (*S. pneumoniae*). Fortino et al. [24] performed condition-dependent operon detection by combining the information from RNA-seq data and genomic sequence features. The algorithm of the Database of

[1]Deep coverage of transcript expression data refers to the process of generating a substantial amount of sequencing reads to obtain detailed information about gene expression within a sample.

Prokaryotic Operons [25] utilizes a combination of linear and non-linear classifiers, considering various features such as intergenic distance, the presence of a specific DNA motif in intergenic region, the ratio of gene lengths, the functional similarity between genes, and the conservation level of the genes' neighborhood.

2.2 Computational Methods - Machine Learning

2.2.1 Operon-mapper

Operon-mapper[27] is a publicly available web-based tool, accessible at `https://biocomputo.ibt.unam.mx/operon_mapper/`, that enables the prediction of operons in prokaryotic genomes. It utilizes computational algorithms derived from the Prokaryotic Operon Database (ProOpDB)[28, 29]. ProOpDB uses a two-layer neural network approach that incorporates intergenic distance and protein functional relationship scores of the genes, obtained from STRING[30]. It was trained and evaluated using *E. coli* and *B. subtilis*. The label sources and reported performances are available in Table 2.2.

Organism	Database	Accuracy	Sensitivity	Specificity
E. coli	RegulonDB (version 6.4)[16]	94.6%	95.2%	93.9%
B. subtilis	DBTBS[13]	93.6%	92.9%	94.9%

Table 2.2: Reported performance of ProOpDB obtained from [29].

2.2.2 Rockhopper

Rockhopper[7], available at `https://cs.wellesley.edu/~btjaden/Rockhopper/`, is a computational tool developed for analyzing RNA-seq data and predicting bacterial operons. It employs a Naïve Bayes model that leverages intergenic distances and gene expression levels extracted from RNA-seq data to make precise operon predictions. It has been trained and evaluated using *E. coli*, *B. subtilis* and *H. pylori*; the label sources and reported performances are in Table 2.3.

Organism	Database	Sensitivity	Specificity
E. coli	RegulonDB (version 6.4)[16]	90%	81%
B. subtilis	DBTBS[13]	88%	96%
H. pylori	[23]	95%	88%

Table 2.3: Reported performance of Rockhopper obtained from [7].

2.2.3 Operon Hunter

Operon Hunter[3] is a deep learning approach to predict operons based on a visual representation of genomes. The top six species with the most experimentally confirmed operons from the ODB[14] were obtained to construct the training data, see Table 2.4. Visual representations of the genomes were obtained using the Compare Region Viewer service provided by PATRIC[31], see Fig.2.1.

To address the limited size of the training data, operon hunter uses transfer learning to re-train the ResNet18 model, which is trained on ImageNet from FastAI[32].

The final model preserves the visual recognition strengths of the original ResNet18 and uses it to detect whether a pair of genes are in the same operon or not. The performance of Operon Hunter is reported in Table 2.5.

Organism	Operon pairs	Non-operon pairs
E. coli	1443	1322
Listeria monocytogenes (*L. monocytogenes*)	806	780
Legionella pneumophila (*L. pneumophila*)	611	791
Corynebacterium glutamicum (*C. glutamicum*)	525	396
Photobacterium profundum (*P. profundum*)	447	544
B. subtilis	474	457
Total	**4306**	**4290**

Table 2.4: Details of the data used to train Operon Hunter.

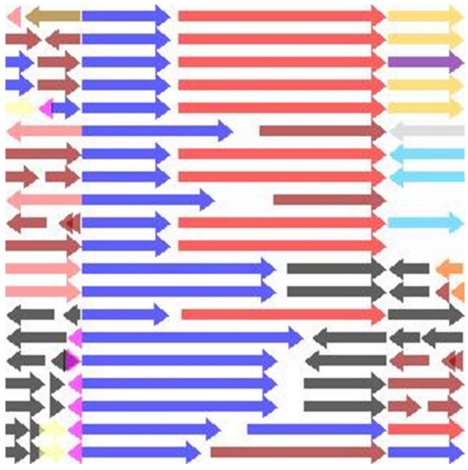

Figure 2.1: Visual representations by PATRIC. Arrows represent genes and their sizes indicate their actual relative length proportions. Red and Blue arrows represent the target paired genes, the first row is the query genome, and each other row is a region in an evolutionary close genome. This figure is from [3] (CC BY 4.0).

Organism	Database	Sensitivity	Specificity
E. coli	ODB[14]	88%	95%
B. subtilis	ODB[14]	97%	88%
Aggregate	ODB[14]	92%	92%

Table 2.5: Reported performance of Operon Hunter obtained from [3].

In contrast to previous studies, Operon Hunter excludes the genome they aimed to evaluate from the training data. This exclusion is crucial as it prevents data leakage and over-optimistic evaluation of the model. Data leakage refers to the unintended transfer of information from the training data to the evaluation phase of a model, which can misleadingly boost performance metrics [33]. Moreover, the performance of a model can be misleading when the model is biased on patterns specific to the training data and is assessed only with similar data, the same organism in this scenario. These unintentional biases can lead to an over-optimistic evaluation of the model. A clear separation of train and test data must be conducted in order to guarantee a robust assessment of machine learning models.

In addition to addressing reliable evaluation concerns, excluding the genome from the training data verifies the generalizability of models. The variation between the genome of different organisms can be significant; by evaluating the model on an unseen genome, the model's capacity to handle genomic variations is tested.

Operon Hunter introduces a novel approach to represent a genome that preserves features such as gene conservation, strand direction, size, and intergenic distance,

which were broadly used for operon detection in the literature. By leveraging the power of deep learning, Operon Hunter achieves performance on par with previous works in the field. This innovative approach unlocks the potential of utilizing deep learning techniques for accurate operon prediction.

Operon Finder [34], available at `https://www.mefyi.com/operon`, is a user-friendly web service that builds upon Operon Hunter. Operon Finder utilizes a similar deep learning-based model to predict operons in prokaryotic genome. However, it improves upon the implementation of the prediction pipeline and model architecture to enhance computational efficiency and user experience.

2.2.4 OperonSEQer

OperonSEQer[11] is a voting system that combines six machine learning algorithms, including Logistic Regression with L2 ridge regularization, Support Vector Machine with an RBF kernel, Random Forest, XGBoost and Multi-Layer Perceptron. This system takes RNA-seq reads as input and extracts Kruskal-Wallis [35] statistics and p-values for gene pairs and their intergenic regions from the raw reads to determine whether the read counts of the genes and their intergenic region are statistically different. These features serve as input for each of the algorithms in the system. The operon labels used for training OperonSEQer were obtained from MicrobesOnline[36], a computational database known for its extensive collection of organisms. The training RNA-seq data used in this system encompassed eight different organisms in diverse

environments: *E. coli*, *B. subtilis*, *Clostridium difficile* (*C. difficile*), *Burkholderia pseudomallei* (*B. pseudomallei*), *Staphylococcus aureus* (*S. aureus*), *Synechococcus elongatus* (*S. elongatus*) PCC 7942, *Synechocystis* sp. PCC 6803, *Synechococcus* sp. PCC 7002.

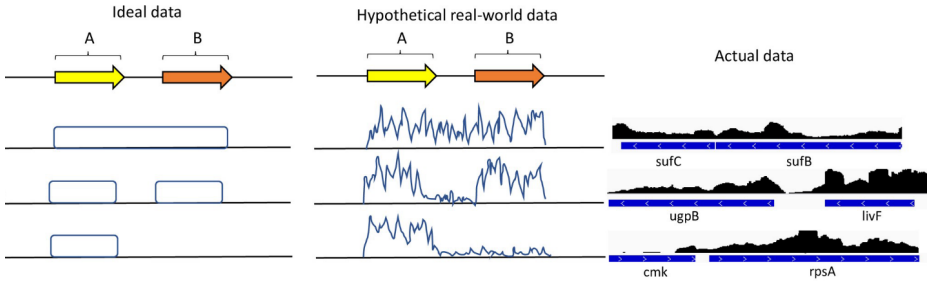

Figure 2.2: From top to bottom, examples of relative expression levels in operon pairs, non-operon pairs (both genes expressed), and non-operon pairs (only one gene expressed). Here, **A** and **B** indicate genes, and the arrows indicate the direction of their transcription. **Actual data** refers to read counts observed from RNA-seq data. This figure is from [11] (CCO 1.0).

The Kruskal-Wallis statistic was chosen as the feature extraction method based on observing RNA-seq signals between adjacent genes, visualized in Fig.2.2. Based on this observation, the expression levels of adjacent genes can imply whether they are in the same operon. OperonSEQer focuses on gene pairs where at least one of the genes exhibits sufficient expression, typically defined as an average of 10 reads.

The reported performance of OperonSEQer is recall and specificity of at least 80%, with respect to MicrobesOnline labels. The validation data used for assessing the

performance of OperonSEQer consists of RNA-seq reads from four of the organisms included in the training process. Additionally, the individual models' performances with respect to MicrobesOnline labels, are presented in Table 2.6.

Algorithm	Recall	Specificity
Support Vector Machine	91%	84%
Multilayer Perceptron	92%	81%
Logistic Regression with Ridge	93%	87%
Random Forest	95%	94%
Gaussian Naïve Bayes	95%	80%
XGBoost	99%	99%

Table 2.6: Reported validation performance of OperonSEQer by [11].

2.3 Research Gap and Objectives

Operon detection plays a crucial role in understanding the organization and regulation of bacterial genomes. Previous methods have demonstrated accurate results for well-studied species such as *E. coli* and *B. subtilis*. However, developing a system that detects operons in any bacterial genome remains a significant challenge. Therefore, this research aims to design a species-agnostic system that accurately detects operons in bacterial genomes.

Previous models for operon detection exhibit several limitations, which necessitate further research in this field:

- **Lack of Systematic Assessment of Models:** The assessment of operon detection models requires more consistency. Different models utilize various reference datasets and measurements, making comparing and selecting the most accurate approach difficult. Moreover, some studies evaluate models on the same organisms used for training, resulting in biased performance estimates and limited generalizability.

- **Training on Computationally Derived Labels:** Certain models, like OperonSEQer, utilize computationally derived labels for training. This approach can lead the model to imitate the decisions made by another computational algorithm, MicrobesOnline. Consequently, the reported performances may not accurately reflect real-world scenarios.

- **Constraining Feature Selection:** Some approaches, like Operon Hunter, incorporate features from external data sources, but these features are typically only available for well-studied organisms. As a result, the generalizability of these models to less-studied bacterial genomes is limited. On the other hand, Primary Sequence features and Transcript Expression features are readily available for most bacteria.

This research aims to address the limitations of previous models for operon detection in bacterial genomes. To achieve this, the following objectives will be pursued:

- **Utilization of Accessible Input Data:** Use genome sequence and RNA-seq

data only, which are commonly accessible for a wide range of bacteria.

- **Better Assessment of Models:** Develop a standard evaluation framework for operon detection models. This involves standardizing reference datasets and evaluation measures. Establishing a systematic evaluation process can enhance the reliability of model assessments and facilitate model comparisons.

- **Utilization of Experimentally Verified Labels:** Incorporate existing experimentally verified operon labels to enhance the real-world performance and reliability of the operon detection system. The model can learn from accurate operon structures by integrating reliable experimental data, leading to improved prediction accuracy and increased confidence in the system's output.

- **Employment of Deep Learning Techniques:** Employ deep learning techniques while ensuring high usability. Operon Hunter demonstrated the possibility of using deep learning techniques for operon detection using Visual representations from PATRIC. However, this study will avoid the constraints imposed by specific data sources that limit usability.

- **Development of a Generalizable Model:** Strive to develop a genome-agnostic operon detection model to detect operons in bacterial genomes across different species effectively. Focus on creating a model that can adapt and perform well even in the presence of dynamic changes in operon structures under varying environmental conditions.

Chapter 3

Methodology

The feature representation pipeline and machine learning model in this study are developed using Python (version 3.10.2), and the list of used packages with their versions is available in Table 3.1.

Package	**Version**	**Package**	**Version**	**Package**	**Version**
numpy	1.23.2	pandas	1.4.0	matplotlib	3.7.0
scipy	1.9.3	tensorflow	2.11.0	scikit_learn	1.2.1
pydot	1.4.2	Orange3	3.30.0		

Table 3.1: List of Python packages with their versions.

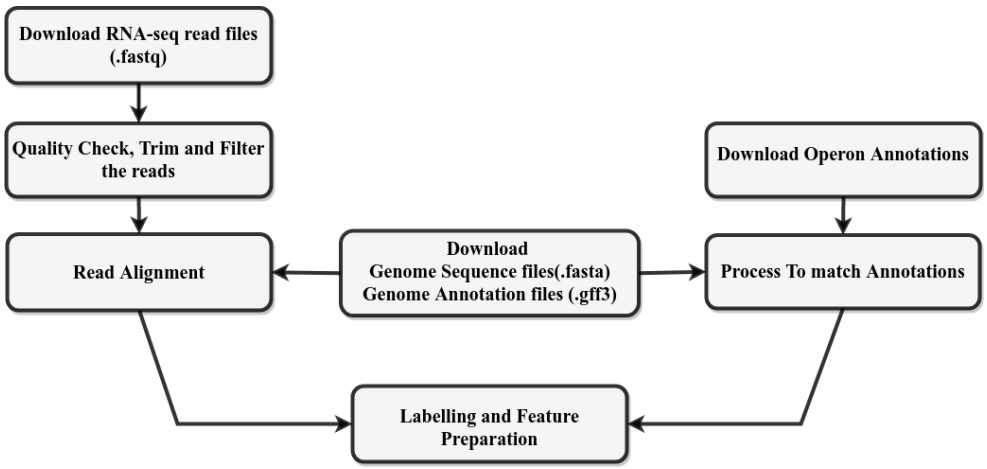

Figure 3.1: Data Gathering workflow

3.1 Data

This study utilizes genome sequences, RNA-seq data, and operon annotations as the primary data sources (Fig.3.1). The genome sequence and annotation files are obtained from the RefSeq[37] database, with a selection of seven bacterial organisms used for training purposes and another seven organisms for the testing process. Considering the dynamic nature of transcription and the potential variations in operon structures under different environmental conditions [7], RNA-seq data for up to six samples of each organism are used. The Sequence Read Archive (SRA)[38] and the European Nucleotide Archive (ENA)[39], the sources of this data, are well-known for storing, sharing, and providing access to high-throughput sequencing data. Detailed information on organisms and their accession codes for genome sequences and RNA-seq data are available in Table 3.2. Operon annotations are obtained from ODB, the fourth version of OperonDB introduced by [14]. This curated database, available at

```
A) fastp -i in.R1.fq [-I in.R2.fq] -o out.R1.fq [-O out.R2.fq]
--cut_front_window_size 1 --cut_front_mean_quality 3
-r --cut_right_window_size 4 --cut_right_mean_quality 15
B) samtools view -b -o BAM_file.bam SAM_file.sam
    samtools sort BAM_file.bam -o sorted_BAM_file.bam
C) bedtools genomecov -d -ibam sorted_BAM_file.bam > read_counts.bed
```

Figure 3.2: A) Fastp command. The -I and -O option are used for paired-end data. B) Commands to convert SAM file to BAM and sort it. C) Command to extract read counts from BAM file.

the website `https://operondb.jp/`, contains experimentally known operons. The number of experimentally verified operon pairs in ODB is in Table 3.3.

The initial steps in processing the data involve trimming and filtering the raw sequencing data in FASTQ format. This process is performed using fastp (version 0.23.1)[40], a reliable tool for quality control and preprocessing RNA-seq reads by eliminating low-quality reads and adapter contamination. The trimmed and filtered FastQ files are then aligned to the reference genomes using HISAT2 (version 2.2.1)[41] with default arguments. HISAT2 employs a mapping algorithm that ensures the precise assignment of sequencing reads to their corresponding genomic locations. RNA-seq read coverage for each genome base was extracted from aligned reads using SAMtools (version 1.17)[42] and BEDtools (version 2.30.0)[43]. The resulting base coverages, along with the operon labels obtained from ODB, form the basis for our

feature representation. The specific employed commands are detailed in Fig.3.2.

	Organism		RNA-seq				
	Name	**Genome**	**Study**	**Samples**			**Ref**
Training	*B. subtilis* subsp. subtilis str. 168	NC_000964.3	GSE179533	SRR15049591	SRR15049592	SRR15049593	[44]
			E-MTAB-10658	ERR6156944	ERR6156945	ERR6156946	[45]
	C. glutamicum ATCC 13032	NC_006958.1	GSE120924	SRR7977557	SRR7977561	SRR7977565	[46]
			E-MTAB-8070	ERR3380462	ERR3380465	ERR3380468	[47]
	E. coli K-12 substr. MG1655	NC_000913.2	GSE65642	SRR1787590	SRR1787592	SRR1787594	[48]
			GSE114917	SRR7217927	SRR7217928	SRR7217929	[49]
	H. pylori 26695	GCA_000008525	GSE94268	SRR5217496			[50]
	L. pneumophila str. Paris	NC_006368.1	E-MTAB-4095	ERR1157043	ERR1157044	ERR1157045	[51]
	L. monocytogenes EDG-e	NC_003210.1	GSE152295	SRR11998208	SRR11998211	SRR11998214	[52]
				SRR11998217	SRR11998220	SRR11998223	
	Mycoplasma pneumoniae (*M. pneumoniae*) M129	NC_000912.1	E-MTAB-8537	ERR3672190	ERR3672191	ERR3672192	[53]
				ERR3672193			
Validation	*P. profundum* SS9	NC_006370.1, NC_006371.1	GSE38259	SRR500950	SRR500951		[54]
	Agrobacterium fabrum (*A. fabrum*) str. C58	NC_003062, NC_003063	GSE173921	SRR14432343	SRR14432344	SRR14432345	NA
	Borrelia burgdorferi (*B. burgdorferi*) B31	NC_001318.1	GSE152295	SRR11997800	SRR11997801	SRR11997802	[52]
	Bradyrhizobium diazoefficiens (*B. diazoefficiens*) USDA 110	NC_004463.1	GSE163004	SRR13238987	SRR13238988	SRR13238989	[55]
	Pseudomonas aeruginosa (*P. aeruginosa*) PAO1	NC_002516.2	GSE152295	SRR11998427	SRR11998428	SRR11998429	[52]
	Yersinia pestis (*Y. pestis*) CO92	NC_003143.1	PRJNA384395	SRR5489122	SRR5489125		NA
	C. elegans	GCF_000002985.6	GSE149300	SRR11605370	SRR11605378	SRR11605385	NA

Table 3.2: Data used in this study.

Train		Validation	
Organism	**Operon pairs**	**Organism**	**Operon pairs**
E. coli K-12 substr. MG1655	1726	*C. elegans*	1184
C. glutamicum ATCC 13032	1077	*P. profundum* SS9	676
L. monocytogenes EDG-e	1031	*P. aeruginosa* PAO1	67
L. pneumophila str. Paris	877	*B. burgdorferi* B31	20
H. pylori 26695	744	*B. diazoefficiens* USDA 110	15
B. subtilis subsp. subtilis str. 168	644	*A. fabrum* str. C58	14
M. pneumoniae M129	246	*Y. pestis* CO92	6
Total	**6345**	**Total**	**1982**

Table 3.3: Number of operon pairs in each organism, from ODB. As we use the same database for operon annotations as Operon Hunter, this table can be compared with Table 2.4. This comparison shows we have increased the number of species and operon pairs by using fewer constraints toward organisms. Moreover, *C. elegans* is the only known eukaryote whose genome is organized in operons. We further evaluated our model's generalizability by testing its performance on this non-bacterium organism.

3.2 Feature generation

The feature representation used in this study draws inspiration from signal processing techniques, particularly the work of [56] in classifying human activities using signals from wearable sensors. However, instead of analyzing signals over time, we focus on the RNA-seq read counts across nucleotide bases in a genome. In our case, the different sensors correspond to multiple samples of the same organism. This perspective

allows us to leverage signal processing advancements for our task, operon detection.

By adopting this representation, we aim to maximize the utilization of information from RNA-seq data. Previous approaches often relied on statistical analyses of RNA-seq data, which by summarizing the data with statistics, removed potential informative patterns from the input data. Our feature representation enables us to not only use this information but also to combine them with primary sequence features, such as gene length, gene borders, and intergenic distances. Another advantage of our feature representation is the compatibility of the final dataset's shape with convolutional neural network architectures which allows us to take advantage of advancements in this field.

The features used in our model are derived from read counts per base of the genome, and the operon labels obtained from ODB. To construct the feature representation, we follow a step-by-step process:

1. **Grouping Reads:** The read counts for each gene are grouped using genome annotations, which specify each gene's start and end bases. We extract the corresponding portion of the read count data from the starting base to the end base of each gene and put the read counts in a vector.

2. **Pairing Genes:** The vector of read counts of consecutive genes are paired up, and their intergenic region's read counts are extracted by assembling a vector for each gene pair that includes the read counts from the first gene's end base to the second gene's start base. This process is done separately for each strand

(forward and reverse) to ensure that genes from different strands are not paired together. The read counts vector for each pair consists of [first gene, intergenic region, second gene].

3. **Handling Samples:** The Pairing Genes process is repeated for up to six samples for each organism. If an organism has fewer than six samples, the available samples are duplicated to fill the remaining slots, which ensures consistency across all organisms, with each gene pair having six samples.

4. **Resampling:** All vectors are resampled so that the total size of genes and their intergenic distance is a fixed size of 150. The relative size of each part is calculated, and the vectors of genes and intergenic regions are resampled to these sizes. This step ensures uniform array sizes across all gene pairs. Also, as the relative length of genes and their intergenic region is reflected, these primary sequence features are built-in into the input.

5. **Scaling:** After resampling, we perform scaling to transform the gene pairs' read counts to a range of 0 to 255. This scaling is carried out because we ultimately treat the gene pairs as visualizations akin to images. Scaling the read counts to the image range allows us to interpret and analyze the data effectively.

6. **Constructing Channels:** Next, we construct separate channels for each part of the gene pairs by padding the vectors with zeros to replace the other parts. By doing so, we create arrays where the gene of interest retains its original

values while the locations of the other parts are filled with zeros. For example, the vector of the first gene in each pair starts with the read counts of that gene, followed by zeros for the length of the intergenic region and the second gene. This process makes each part's final length to be the fixed size we determined earlier, 150. This construction of separate channels allows us to treat each vector as an independent entity within our feature representation. It parallels the concept of RGB channels in images, where each channel emphasizes different aspects of visual information. In our case, the separate channels highlight the first gene, intergenic regions, and second gene of the gene pair, respectively.

7. **Final Shape:** The resulting vector for each gene pair has a shape of (resample size, number of samples, number of vectors). Hence, the shape of each gene pair's vector in the data is (150, 6, 3); As we use the vectors for two neighbouring genes and their intergenic region, the number of vectors is three.

8. **Labelling:** Labels are assigned to each gene pair based on the operon annotation from ODB. Gene pairs labelled as 1 are in the same operon, while pairs labelled as 0 are not. Pairs labelled as 2 indicate that at least one gene is not mentioned in the annotation file, meaning there is insufficient experimental evidence to determine their operon relationship. Gene pairs with label 2 are excluded from the training data. Note that label 2 is only used for filtering gene pairs. The size of the training data and each label is in Table 3.4.

Label	Operon(**1**)	Non-operon(**0**)	Unknown(**2**)
Number of gene pairs	6345	4030	9828

Table 3.4: The size of each label class in our training data. Labels 0 and 1 are used to train the model, making the final size of train data **10375** gene pairs.

Figs. 3.3, 3.4, and 3.5, illustrate examples of the data visualization obtained from our processing pipeline, showcasing different labels. These visualizations utilize a colour scheme where red, green and blue represent the first gene, the intergenic region, and the second gene. Each figure consists of a series of subfigures, with six subfigures displaying the read count signals over nucleotide bases for each sample corresponding to the gene pair. As observed in Fig.2.2, the transcription levels of genes and their corresponding intergenic regions align with their presence in the same operon.

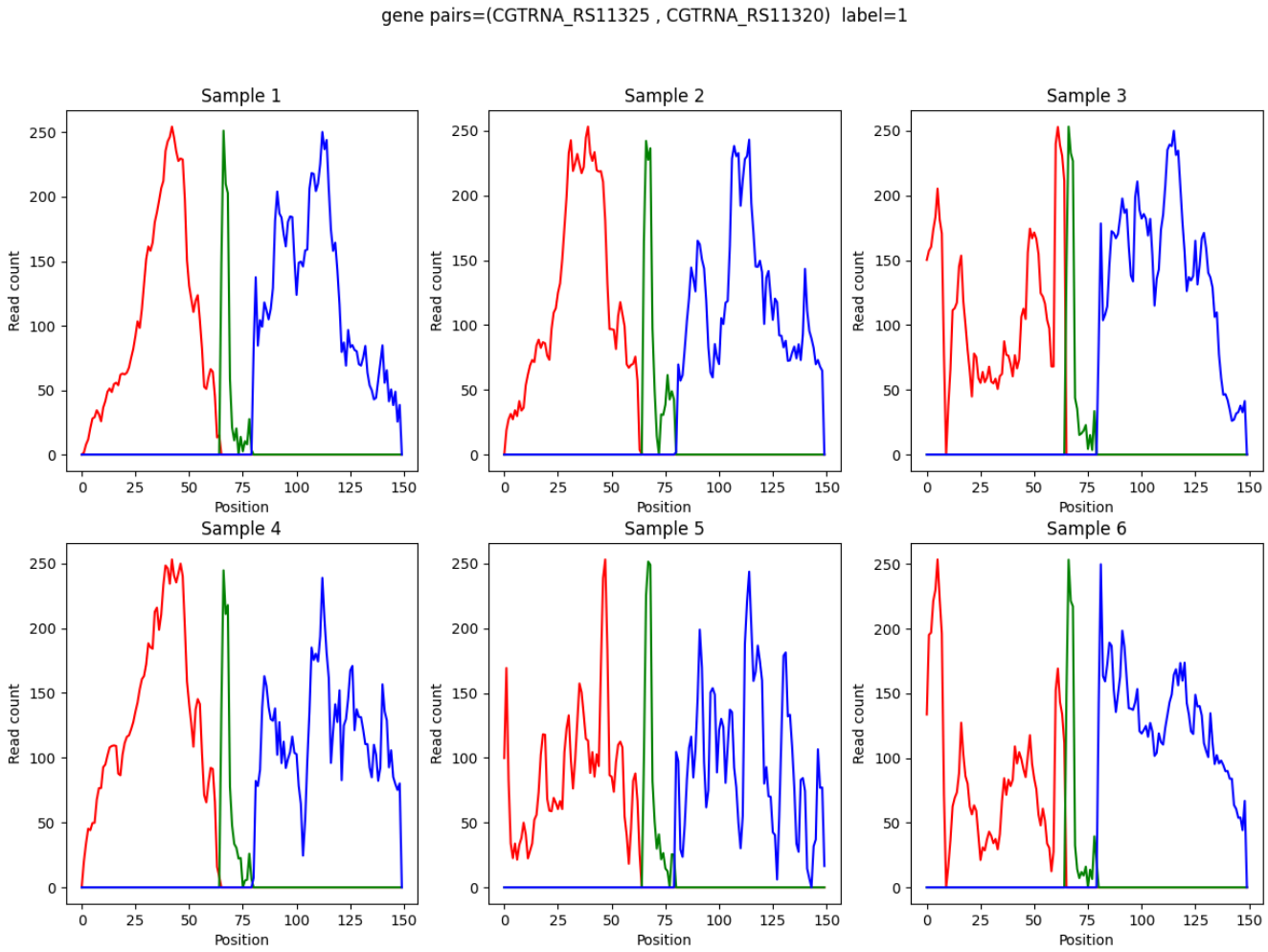

Figure 3.3: Example of data visualization with label Operon(**1**) in *C. glutamicum* ATCC 13032. The top six plots show the resampled read counts for the first gene (red), intergenic region (green) and second gene (blue). The pattern in this sample resembles the first example in Fig. 2.2

.

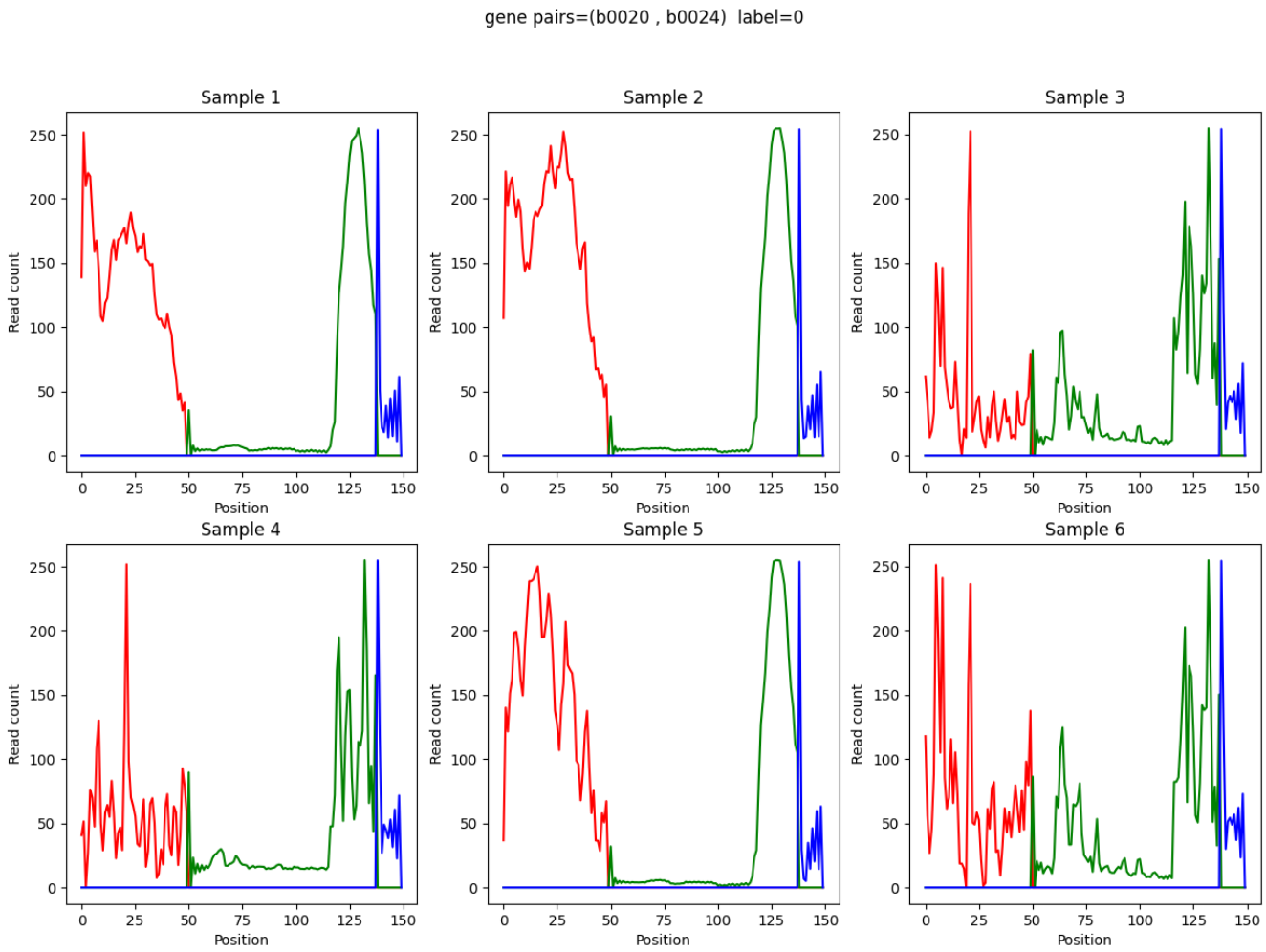

Figure 3.4: Example of data visualization with label Non-operon(**0**) in *E. coli* K-12 substr. MG1655. The top six plots show the resampled read counts for the first gene (red), intergenic region (green) and second gene (blue). The pattern in this sample resembles the second example in Fig. 2.2

.

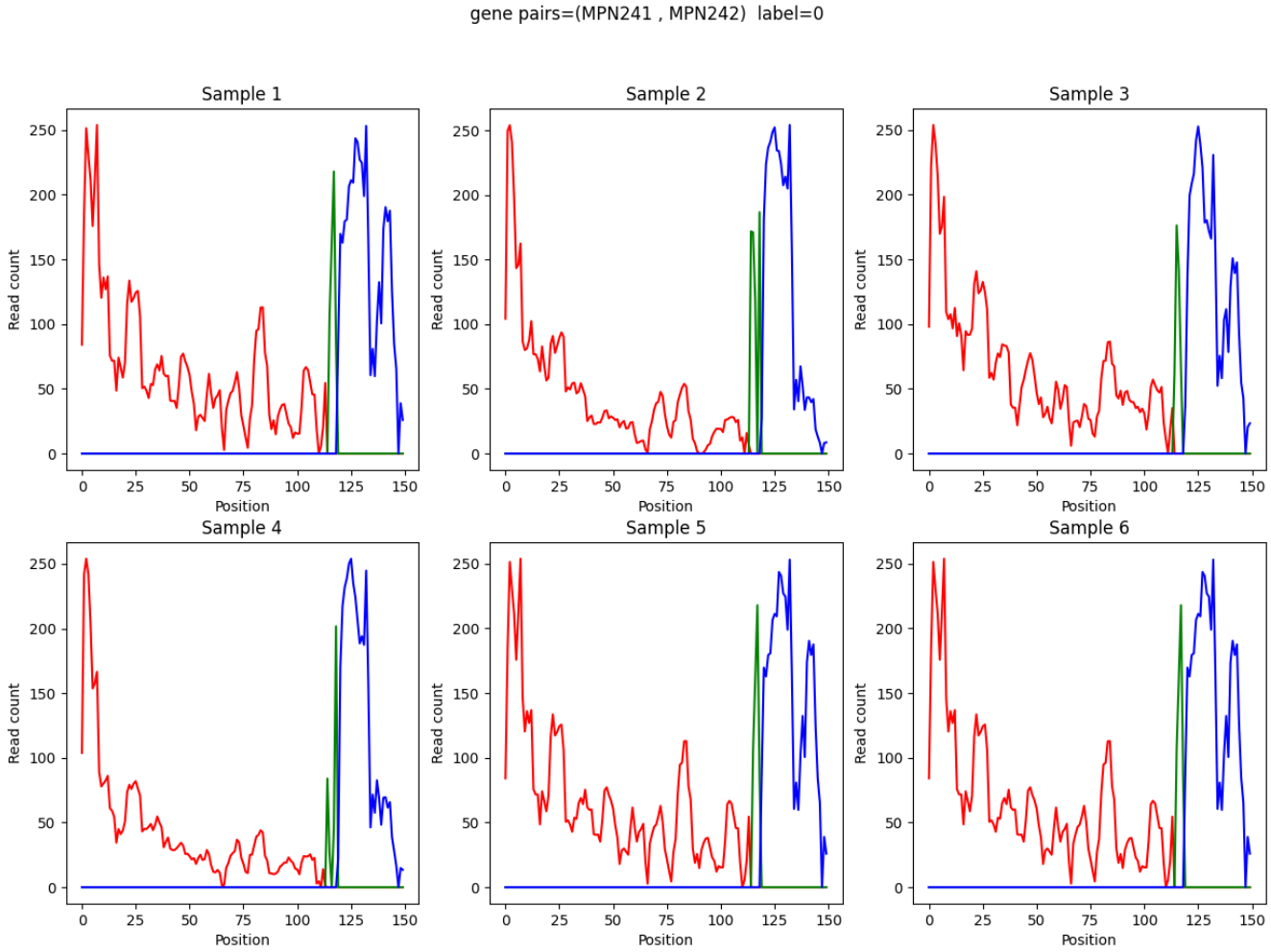

Figure 3.5: Example of data visualization with label Non-operon(**0**) in *M. pneumoniae* M129. The top six plots show the resampled read counts for the first gene (red), intergenic region (green) and second gene (blue). The pattern in this sample resembles the third example in Fig. 2.2

.

3.3 Model

Convolutional Neural Network-Long Short-Term Memory (CNN-LSTM), is the deep learning architecture employed by [56]. It combines the strengths of two popular neural network models, Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) [57], to capture both spatial and temporal dependencies in data. In the context of this study, this fusion of architectures proves beneficial for operon detection, allowing for efficient and accurate analysis of the underlying sequential patterns in RNA-seq data.

The CNN component of the architecture is responsible for extracting spatial features from the input data. CNNs are widely used in image recognition tasks, as they are adept at detecting patterns and structures within the data. By applying convolutional filters to the input data, CNNs can effectively learn hierarchical representations, enabling them to identify meaningful features.

The LSTM component, on the other hand, is a specialized type of recurrent neural network (RNN) designed to model and capture temporal dependencies in sequential data. LSTMs are particularly effective in processing and understanding time series data, as they can remember information from previous time steps and utilize it to make predictions. This memory capability allows LSTMs to handle long-range dependencies, making them suitable for tasks involving sequential patterns.

Self-attention mechanism by [56] is a notable extension to CNN-LSTM architecture. This mechanism enhances the model's ability to focus on informative features

within the data by assigning weights to different elements based on their relevance.

3.3.1 Architecture

The CNN-LSTM architecture that is utilized in this study includes a CNN, a Lambda, an LSTM, an Attention, and a Dense layer, respectively.

The CNN layer functions as a feature detector, extracting spatial patterns from the input data. It scans the data and identifies relevant features that are crucial for understanding the underlying structure. A Lambda layer is incorporated to ensure a seamless transition between the CNN and LSTM layers. Its purpose is to reshape the output from the CNN layer, making it compatible with input into the LSTM layer. The LSTM layer plays a critical role in capturing the sequential dependencies present in the data. Its ability to remember information from previous steps allows it to predict future steps. To further enhance the model's performance, an Attention layer is included. This layer assigns weights to different elements of the sequence based on their relevance, enabling the model to focus on important regions within the data. The dropout regularization technique is employed to prevent overfitting, which is a common problem in deep learning models. Dropout randomly sets a fraction of the input units to 0 at each training step, forcing the model to learn more robust and generalized representations. This regularization technique helps prevent the model from relying too heavily on specific features or patterns in the training data, thereby improving its ability to generalize to new, unseen data. Finally, the Dense layer takes

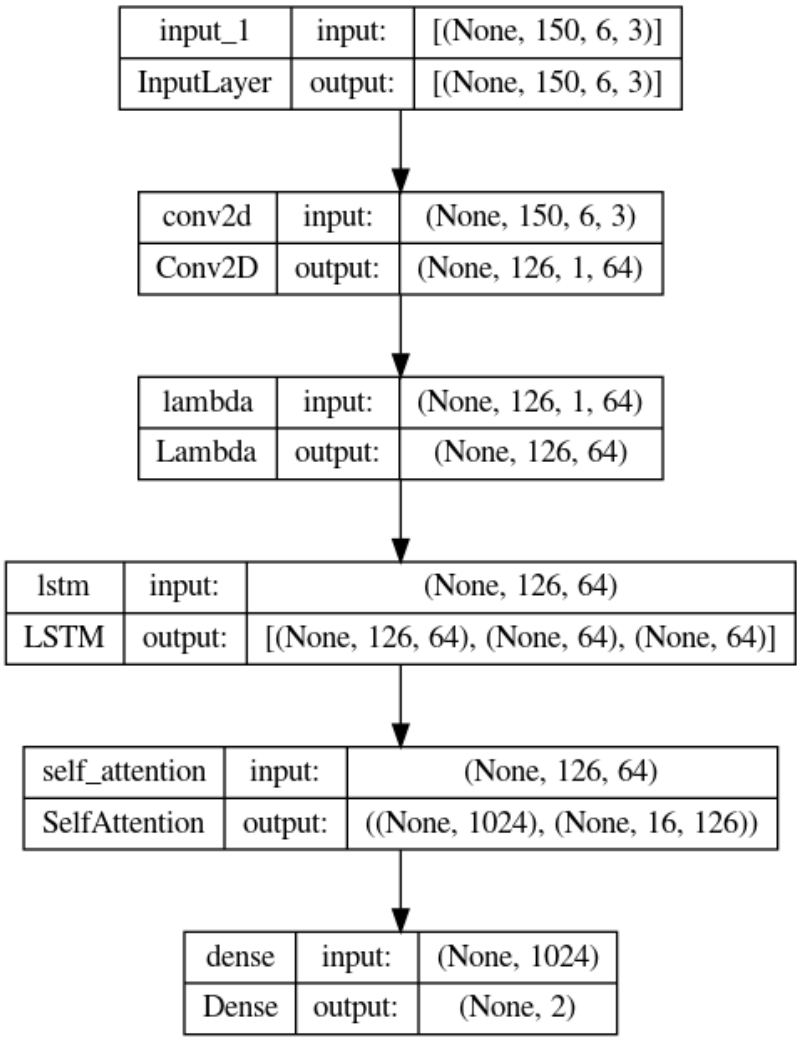

Figure 3.6: Architecture. (None, 150, 6, 3) specifies that the CNN model can accept input data with variable batch sizes (None), where each sample has a length of 150 elements (resample size), a height of 6 rows (number of samples), and a width of 3 columns (number of vectors). The Lambda layer is used to adjust the shape of the output from the CNN layer to input for the LSTM layer.

the processed information from the previous layers and generates the final binary output and the classification result. The architecture is illustrated in Fig.3.6.

3.3.2 Training

During the training process, the dataset consisted of 6,289 gene pairs labelled as 1, indicating that they belonged to the same operon, and 3,998 gene pairs labelled as 0, indicating that they did not belong to the same operon (Table 3.4). The remaining 9,765 gene pairs labelled as 2 were excluded from the training data to prevent the injection of false information into the model, as these gene pairs had incomplete information, making it unclear whether or not they belonged to the same operon. To prevent overfitting, the model underwent 10-fold cross-validation during training, and the final decision is based on the average of the results across the folds. This technique divides the dataset into ten equal parts, called folds, and performs training and evaluation ten times, each time using a different fold as the test set and the remaining folds as the training set. This approach allows for a robust assessment of the model's performance across different data subsets.

The hyperparameters listed in Table 3.5 were selected to achieve a high AUROC. The specifications for the attention layer in this model are exactly as defined in [56]. Since the attention layer hyperparameters have been proven effective in the referenced study, we decided to adopt them without modification in our implementation.

Moreover, several techniques and metrics were utilized to improve and evaluate

Hyperparameter	Value	Hyperparameter	Value	Hyperparameter	Value
Batch size	32	Epoch	100	Dropout rate	0.3
Kernel size	(5, 6)	CNN filters	64	LSTM units	64
Optimizer	Adam	Learning rate	0.001	Early stopping patience	10
Metric	AUROC	Loss	Categorical cross entropy		

Table 3.5: Hyperparameters used in the model.

the performance of the model. These include early stopping, the Adam optimizer, and the AUROC as the evaluation metric. Early stopping prevents overfitting by monitoring the model's performance on a validation set during training. If the performance does not improve for a specified number of epochs (patience), training is stopped. This method helps prevent the model from continuing to train on data that it has already learned, thus improving its generalization ability. Adam is an adaptive optimization algorithm widely used for training deep learning models. It adjusts the learning rate based on parameter gradients, leading to faster convergence and improved optimization performance. AUROC is a metric used to evaluate the performance of binary classification models, particularly when dealing with imbalanced datasets. It measures the model's ability to classify positive instances while minimizing false positives. Maximizing the AUROC indicates a better model performance in operon detection.

3.3.3 Evaluation

In this section, we discuss the metrics used to evaluate our final model's performance compared to existing models in the literature. For this evaluation, we consider only samples with 0 and 1 labels in Table 3.6,

Organism	Operon(**1**)	Non-operon(**0**)	Unknown(**2**)
C. elegans	1184	56	43520
P. profundum SS9	676	87	4806
P. aeruginosa PAO1	67	3	63264
B. burgdorferi B31	20	0	817
B. diazoefficiens USDA 110	15	2	7119
A. fabrum str. C58	14	0	4560
Y. pestis CO92	6	0	4028

Table 3.6: The size of each data set per label class in our validation data. Labels 0 and 1 are used to compare the performance of models.

Recall, also known as sensitivity or true positive rate, quantifies the proportion of correctly predicted positive instances out of all actual positive instances. In the context of operon detection, recall is of paramount importance as it indicates the model's ability to identify operons, which are positive instances. Precision, on the other hand, quantifies the proportion of correctly predicted positive instances out of all instances predicted as positive. It provides insights into the accuracy of positive predictions made by the model. F1-score is a metric that combines precision and recall into a single value by taking their harmonic mean. This metric is particularly

useful when dealing with imbalanced datasets.

To comprehensively evaluate our model's effectiveness in operon detection, we primarily consider the f1-score. However, due to limited 0-label samples in five of the seven test organisms, f1-score calculation is unreliable. To overcome this, we assess our model's performance by comparing the number of identified operons with those reported in the literature for each organism.

In addition to these metrics, we also utilize the AUROC. The Receiver Operating Characteristic (ROC) curve visually represents the trade-off between the True Positive Rate (TPR) and the False Positive Rate (FPR) at various classification thresholds. The ROC curve is created by plotting TPR on the y-axis against FPR on the x-axis. It allows us to assess the model's discriminative ability across different thresholds. An ideal model with high TPR and low FPR values would exhibit a curve approaching the plot's top-left corner, representing perfect prediction. This curve also includes a reference line known as the "random chance" line. The random chance line represents the expected performance of a classifier that makes predictions by random chance. Any model that performs better than random chance will have a ROC curve above the random chance line. Additionally, (AUROC) is calculated as a summary measure of the model's discriminative performance. A higher AUROC indicates a better overall performance.

3.4 Comparative assessment

In order to compare the performance of our model with existing approaches, we evaluated four models from the literature using all training and validation organisms using the genome files listed in Table 3.2. Here, we provide a brief overview of our evaluation process for each model:

- **Operon Mapper:** In the Operon Mapper's website, `https://biocomputo.ibt.unam.mx/operon_mapper/`, we uploaded the FASTA file with genome sequence and GFF file with the coordinates of the ORFs[1] in the genome sequence as input files for each organism and chose "Predicted operonic gene pairs" as output option. As Operon Mapper does not use transcript expression features, the RNA-seq data was not used. The output file contains gene pairs with the predicted "Opero" or "NoOperon" class. We mapped these classes to 1 and 0 to match the labels we prepared for evaluation.

- **Rockhopper:** To perform the prediction for each organism, we obtained the Rockhopper application from `https://cs.wellesley.edu/~btjaden/Rockhopper/`. Following the provided instructions, we downloaded the Rockhopper.jar file and utilized the command "java -Xmx1200m -cp Rockhopper.jar Rockhopper" followed by the respective genome sequence files and RNA-seq data files in fastq format to execute the prediction process. The output consists of lines with comma-separated gene names, where each line represented an operon. The

[1]Open Reading Frames

downside of this output format is that it focuses on displaying predicted operons, meaning predicted non-operon pairs were not included. To address this limitation, we labelled gene pairs mentioned within the line as operons(1), while labelling other gene pairs as non-operons(0).

- **Operon Finder:** To perform the prediction for each organism, Operon Finder only required to select the organism's name from the provided list of available organisms in `https://www.iitg.ac.in/spkanaujia/operonfinder.html`. It's worth noting that Operon Finder relies on external data sources, which limits its predictions to the organisms available in those specific sources. Similar to Rockhopper, the output of Operon Finder specifically identifies genes belonging to the same operon. To categorize the predictions into operons(1) and non-operons(0), we employed a similar method as described earlier.

- **OperonSEQer:** We obtained the source code of operonSEQer from `https://github.com/sandialabs/OperonSEQer` and utilized it to generate predictions for our dataset. Each sample of organisms from Table 3.2 was separately predicted by operonSEQer, and the final label for each gene pair was determined based on the average prediction from these samples. We used threshold 3 in operonSEQer, as it was the preferred threshold according to [11]. The output file contains gene pairs with the predicted operons(1) and non-operons(0) classes.

We compared the performances of multiple methods on various species using sta-

tistical analyses. We used the Friedman test to assess AUROC differences between the models, using stats.friedmanchisquare() from scipy. For pairwise comparisons, we employed the Nemenyi post-hoc test. We obtained ranks using stats.rankdata() from scipy and calculated the Critical Difference (alpha=0.05) using evaluation.compute_CD() from Orange3 module. The graphical representation of ranks was created using evaluation.graph_ranks() from the same library.

Chapter 4

Results and Discussion

In this chapter, we provide the results of evaluating our final model and comparing its performance with that of existing methods.

4.1 Cross-validation results

For the aggregated training data of the seven organisms, the performance of our model over 10-fold cross-validation is as follows:

- Mean Recall: 89.21%, with a 90% confidence interval[1] of [88.46%, 89.96%].
- Mean F1-score: 89.49%, with a 90% confidence interval of [88.78%, 90.21%].
- Mean AUROC: 89.21%, with a 90% confidence interval of [88.46%, 89.96%].
- Mean Accuracy: 90.09%, with a 90% confidence interval of [89.43%, 90.76%].

[1]The confidence interval suggests that the model's recall is 90% probable to be within this range.

While accuracy is reported, it may not be the most suitable metric when dealing with unbalanced labels. In the case of operon detection, where the goal is to correctly identify positive instances (operons), recall and f1-score are the most informative measures.

4.2 Comparative assessment

4.2.1 Limitations

In the following sections, we conducted a performance comparison between our model and four methods previously described in the literature. While comparing we encountered additional constraints or drawbacks related to the previous method:

- Due to the large size of the genome files for *C. elegans*, we faced difficulties in uploading them to the Operon Mapper website for prediction. Despite our attempts to find alternative solutions, we did not receive any response from the authors. As a result, we currently do not possess predictions of *C. elegans* operons from the Operon Mapper.

- Due to the reliance of Operon Finder on external data sources, the predictions generated by this method are restricted to the organisms that are available within those specific sources. Consequently, were unable to obtain predictions for *Y. pestis* CO92, *B. diazoefficiens* USDA 110, and *C. elegans* as these organisms were not included in the available data sources used by Operon Finder.

- While evaluating operonSEQer, we identified certain constraints in the predictions it generated. Notably, operonSEQer attempted to predict gene pairs from different strands, whereas it is known that genes within the same operon must be located on the same strand [58]. This discrepancy could potentially affect the accuracy of operonSEQer's predictions. Moreover, we observed that this model was unable to predict labels for certain gene pairs due to limitations related to the average read counts of genes.

4.2.2 Validation Organisms

This section presents the evaluation results for each method on the validation organisms. Tables 4.1, 4.2, 4.3, 4.4, 4.5, 4.6, and 4.7 display the number of predicted and true operons and non-operons for each method per organism.

The highlighted numbers in the tables represent the highest count of correctly identified operons and non-operons among the five evaluated methods. For instance, Table 4.1 shows that for *C. elegans*, operonSEQer correctly identified 225 operons (1) and 17 non-operons (0). Similarly, our method detected 1042 operons (1), which is the highest count among all methods for identified operon pairs, and 33 non-operons (0) in this organism. This table also shows that out of 56 non-operon pairs, only 18, and out of 1184 operon pairs, only 1041 were predicted with a label by operonSEQer. This suggests that operonSEQer struggled to make predictions for a significant portion of the gene pairs in the dataset. This limitation should be taken into account when

assessing OperonSEQer's overall performance compared to the other methods.

To further evaluate the models' discriminative abilities, we have included ROC curves in Figures 4.1 and 4.2, and f1-scores and recalls in Table 4.8. F1-score and ROC curves are only available for two organisms, *C. elegans* and *P. profundum* SS9, since they provide sufficient samples with both labels (operons and non-operons). Upon examining the ROC curves, it becomes evident that our model exhibits the highest AUROC for both organisms. Visually, the lines representing our model, blue lines, are closer to the reference line representing a perfect prediction.

It is important to note that in the case of *P. profundum* SS9 in Table 4.2, operonSEQer and Rockhopper have detected the largest number of operon and non-operon pairs, respectively. However, this higher detection rate comes at the cost of mislabeling a greater number of pairs from the other class. Our model has the best balance between the identified operon and non-operon pairs which is evident in Fig.4.2, where our model has the highest AUROC among the five methods for *P. profundum* SS9.

The evaluation results demonstrate that our model has higher recall rates than the other evaluated models. On *P. profundum* SS9, our model has a slightly lower f1-score than OperonSEQer (70% vs 74%); however, it is worth noting that our model successfully labelled all the gene pairs in the input data while OperonSEQer missed to give a prediction for 2.5% of the gene pairs, which might have contributed to achieving a higher f1-score.

Operon Mapper	Rockhopper			
	True/Predicted	0	1	All
NA	0	56	0	56
	1	1117	67	1184

Operon Finder	OperonSEQer			
	True/Predicted	0	1	All
NA	0	17	1	18
	1	816	225	1041

Our model			
True/Predicted	0	1	All
0	33	23	56
1	142	1042	1184

Table 4.1: Comparative performance on *C. elegans*. Highlighted numbers indicate the highest number of True Positives (1,1) or True Negatives (0,0) achieved.

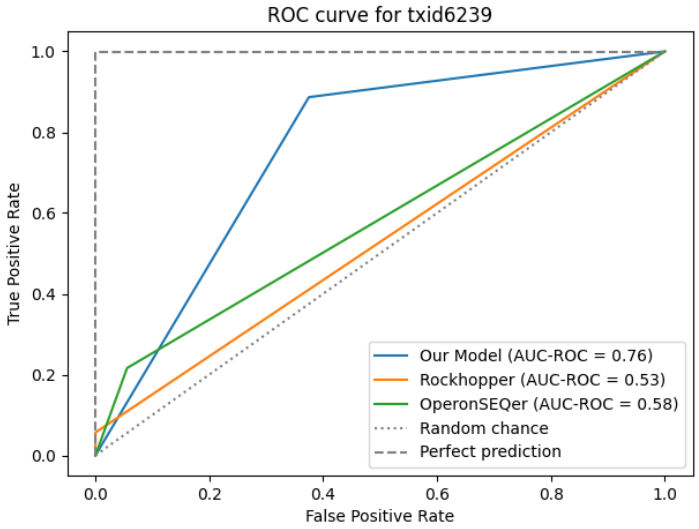

Figure 4.1: ROC for *C. elegans*.

Operon Mapper				**Rockhopper**			
True/Predicted	0	1	All	True/Predicted	0	1	All
0	65	22	87	0	74	13	87
1	391	285	676	1	504	172	676

Operon Finder				**OperonSEQer**			
True/Predicted	0	1	All	True/Predicted	0	1	All
0	43	44	87	0	31	37	68
1	71	605	676	1	20	656	676

Our model			
True/Predicted	0	1	All
0	60	27	87
1	95	581	676

Table 4.2: Comparative performance on *P. profundum* SS9. Highlighted numbers indicate the highest number of True Positives (1,1) or True Negatives (0,0) achieved.

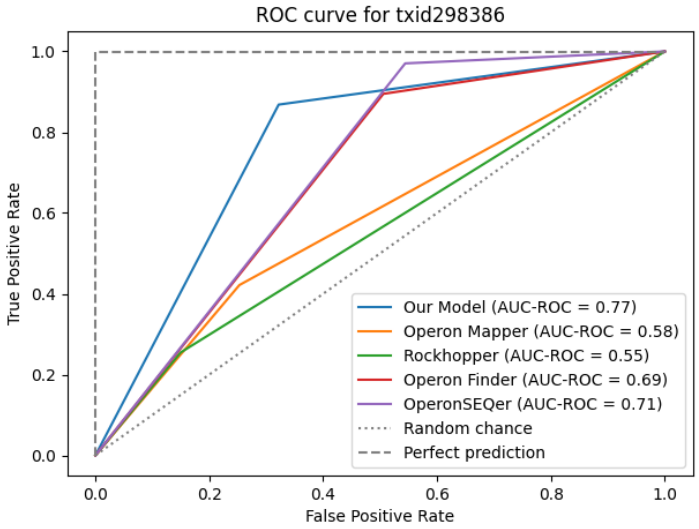

Figure 4.2: ROC for *P. profundum* SS9.

Operon Mapper

True/Predicted	0	1	All
0	3	0	3
1	27	40	67

Rockhopper

True/Predicted	0	1	All
0	3	0	3
1	32	35	67

Our model

True/Predicted	0	1	All
0	2	1	3
1	0	67	67

Operon Finder

True/Predicted	0	1	All
0	3	0	3
1	4	63	67

OperonSEQer

True/Predicted	0	1	All
0	1	1	2
1	6	61	67

Table 4.3: Comparative performance on *P. aeruginosa* PAO1. Highlighted numbers indicate the highest number of True Positives (1,1) or True Negatives (0,0) achieved.

Operon Mapper

True/Predicted	0	1	All
0	-	-	-
1	15	5	20

Rockhopper

True/Predicted	0	1	All
0	-	-	-
1	14	6	20

Our model

True/Predicted	0	1	All
0	-	-	-
1	0	20	20

Operon Finder

True/Predicted	0	1	All
0	-	-	-
1	3	17	20

OperonSEQer

True/Predicted	0	1	All
0	-	-	-
1	5	15	20

Table 4.4: Comparative performance on *B. burgdorferi* B31. No non-operon labels are available in ODB4. Highlighted numbers indicate the highest number of True Positives (1,1) or True Negatives (0,0) achieved.

Operon Mapper

True/Predicted	0	1	All
0	-	-	-
1	12	2	14

Rockhopper

True/Predicted	0	1	All
0	-	-	-
1	9	5	14

Our model

True/Predicted	0	1	All
0	-	-	-
1	0	14	14

Operon Finder

True/Predicted	0	1	All
0	-	-	-
1	1	13	14

OperonSEQer

True/Predicted	0	1	All
0	-	-	-
1	2	12	14

Table 4.5: Comparative performance on *A. fabrum* str. C58. No non-operon labels are available in ODB4. Highlighted numbers indicate the highest number of True Positives (1,1) or True Negatives (0,0) achieved.

Operon Mapper

True/Predicted	0	1	All
0	1	1	2
1	5	10	15

Rockhopper

True/Predicted	0	1	All
0	2	0	2
1	12	3	15

Our model

True/Predicted	0	1	All
0	1	1	2
1	2	13	15

Operon Finder

NA

OperonSEQer

True/Predicted	0	1	All
0	1	1	2
1	3	11	14

Table 4.6: Comparative performance on *B. diazoefficiens* USDA 110. Highlighted numbers indicate the highest number of True Positives (1,1) or True Negatives (0,0) achieved.

Operon Mapper

True/Predicted	0	1	All
0	-	-	-
1	0	6	6

Rockhopper

True/Predicted	0	1	All
0	-	-	-
1	0	6	6

Our model

True/Predicted	0	1	All
0	-	-	-
1	0	6	6

Operon Finder

NA

OperonSEQer

True/Predicted	0	1	All
0	-	-	-
1	0	6	6

Table 4.7: Comparative performance on *Y. pestis* CO92. No non-operon labels are available in ODB4. Highlighted numbers indicate the highest number of True Positives (1,1) or True Negatives (0,0) achieved.

F1-scores

	Operon Mapper	Rockhopper	Operon Finder	OperonSEQer	Our model
C. elegans	NA	10%	NA	20%	61%
P. profundum SS9	41%	31%	67%*	74%	70%
Average	**NA**	**20.5%**	**NA**	**47%**	**65.5%**

Recalls

	Operon Mapper	Rockhopper	Operon Finder	OperonSEQer	Our model
C. elegans	NA	53%	NA	58%	73%
P. profundum SS9	58%	55%	69%*	71%	77%
P. aeruginosa PAO1	80%	76%	97%	71%	83%
B. burgdorferi B31	12%	15%	42%	38%	100%
A. fabrum str. C58.	7%	18%	46%	43%	100%
B. diazoefficiens USDA 110	58%	60%	NA	64%	68%
Y. pestis CO92	100%	100%	NA	100%	100%
Average	**NA**	**53.9%**	**NA**	**63.6%**	**85.9%**

Table 4.8: F1 scores for all validation organisms with non-operon labels, *C. elegans* and *P. profundum* SS9, and recalls for all validation organisms. The highlighted numbers in the tables represent the best performance per organism among different methods. * indicates that the target organism is included in the training data of the corresponding method.

4.2.3 Training Organisms

The evaluation process was extended to the training organisms to assess the performance of all methods on more organisms. The evaluation employed metrics similar to the validation phase, including AUROC, recall and f1-score. To avoid any data leaking and over-optimistic evaluation, we excluded the organism we specifically wanted to evaluate from the training process of our model. However, excluding the target organism from the training of the other methods was not possible since we used their pre-trained models. To show the effect of including the organism to be evaluated in the training data, we also calculated the performance of our model without excluding the target organism from our training data. The ROC curves for the training organisms can be found in figures 4.3, 4.4, 4.5, 4.6, 4.7, 4.8, and 4.9, illustrating the models' detection abilities across different organisms. In these figures, "Our Model NE" refers to our model without excluding the examined organism from the training process. Additionally, Table 4.9 presents the f1-scores and recalls for each method on the training organisms. In every case, our model outperformed or achieved similar performance compared to the other methods, demonstrating its effectiveness.

F1-scores

Organism	Operon Mapper	Rockhopper	Operon Finder	OperonSEQer	Our Model	Our Model NE
E. coli K-12 substr. MG1655	63%*	79%*	87%*	85%*	85%	**90%***
C. glutamicum ATCC 13032	62%	41%	81%*	71%	87%	89%*
L. monocytogenes EDG-e	46%	24%	80%*	34%	85%	89%*
L. pneumophila str. Paris	42%	27%	67%*	62%	79%	81%*
H. pylori 26695	46%	20%*	75%	70%	89%	90%*
B. subtilis subsp. subtilis str. 168	54%*	68%*	82%*	73%*	84%	89%*
M. pneumoniae M129	62%	50%	53%	71%	78%	79%*

Recalls

Organism	Operon Mapper	Rockhopper	Operon Finder	OperonSEQer	Our Model	Our Model NE
E. coli K-12 substr. MG1655	65%*	78%*	87%*	86%*	86%	**90%***
C. glutamicum ATCC 13032	64%	53%	81%*	73%	87%	89%*
L. monocytogenes EDG-e	68%	56%	88%*	63%	82%	**88%***
L. pneumophila str. Paris	63%	57%	77%*	72%	78%	78%*
H. pylori 26695	69%	54%*	87%	85%	87%	88%*
B. subtilis subsp. subtilis str. 168	68%*	79%*	87%*	86%*	84%	**88%***
M. pneumoniae M129	69%	62%	64%	74%	76%	77%*

Table 4.9: F1-scores and recalls achieved on the training organisms. Our model showed a slightly lower f1-score for *E. coli*, and recall for *E. coli*, *L. monocytogenes* and *B. subtilis*. However, Operon Finder includes these organisms in its training data, which contributes to its relatively better performance on this specific organism. As it can be seen from the column "Our Model NE", when the organism to be evaluated upon is left in the training data, our model achieves better performance, outperforming Operon Finder. Note that column "Our Model" more accurately reflects the generalization capabilities of our model. The highlighted numbers in the tables represent the best performance per organism among different methods. * indicates that the target organism is included in the training data of the corresponding method.

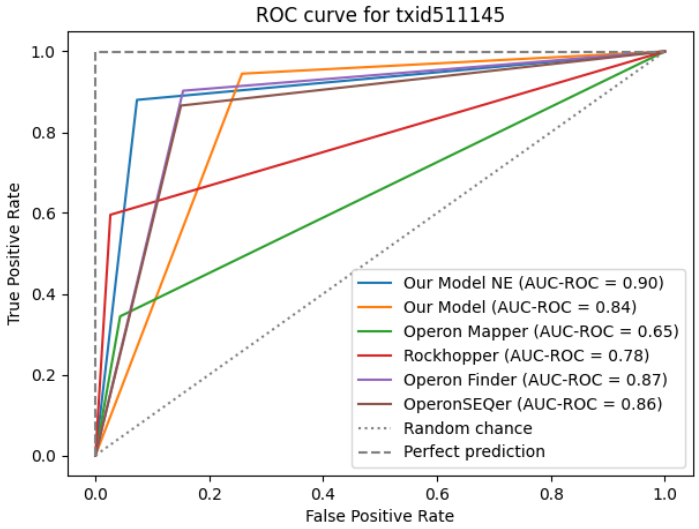

Figure 4.3: ROC for *E. coli* K-12 substr. MG1655.

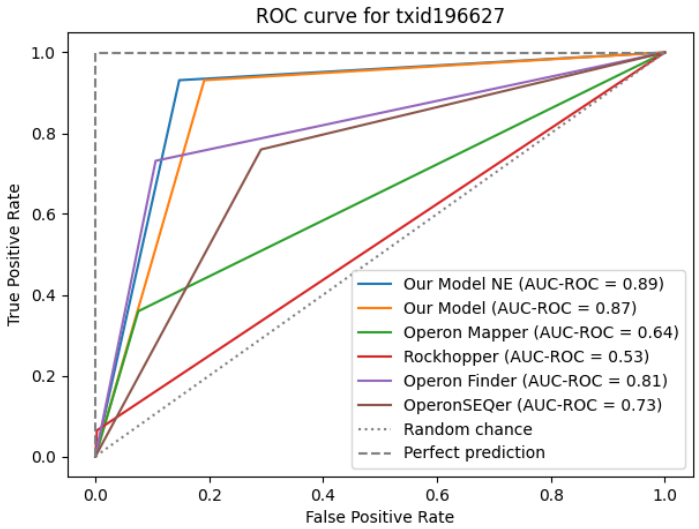

Figure 4.4: ROC for *C. glutamicum* ATCC 13032.

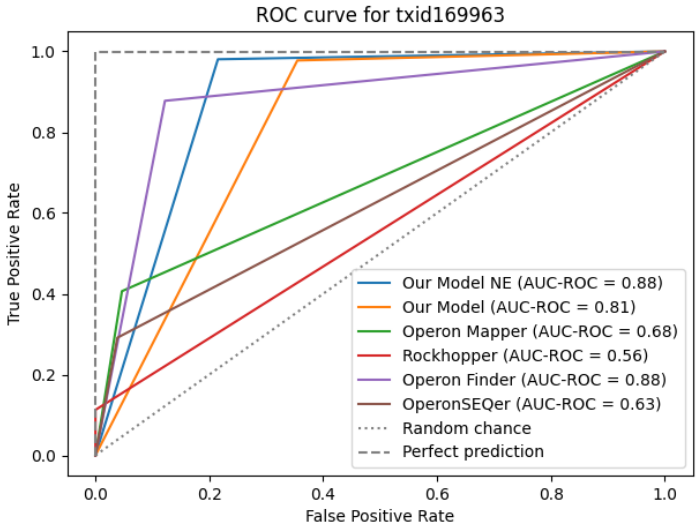

Figure 4.5: ROC for *L. monocytogenes* EDG-e.

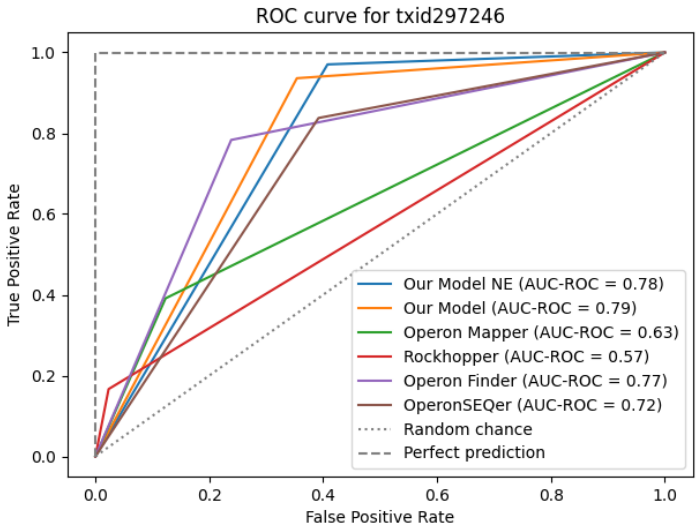

Figure 4.6: ROC for *L. pneumophila* str. Paris.

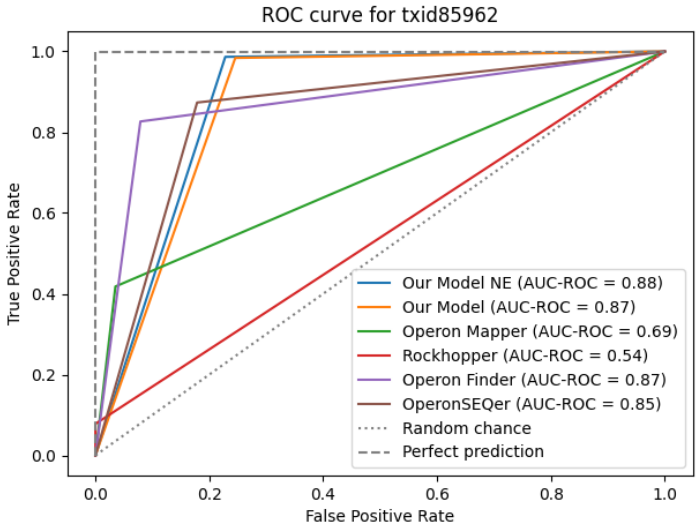

Figure 4.7: ROC for *H. pylori* 26695.

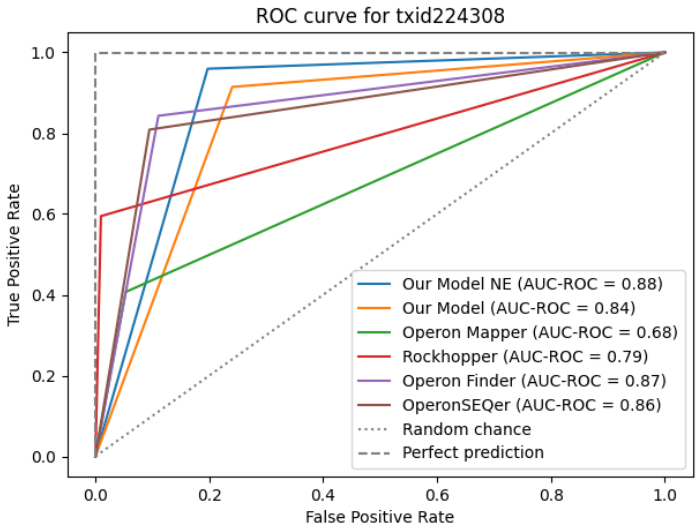

Figure 4.8: ROC for *B. subtilis* subsp. subtilis str. 168.

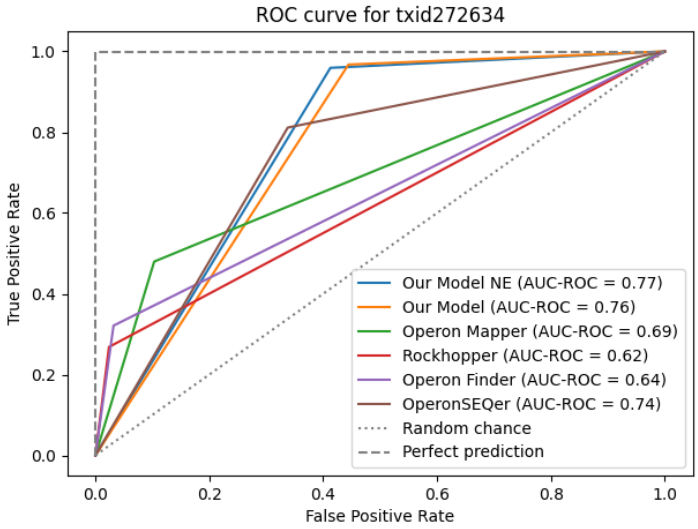

Figure 4.9: ROC for *M. pneumoniae* M129.

4.3 Statistical analysis of the results

To highlight the performance differences among the models, we generated a plot displaying the mean AUROC obtained on all training organisms and two validation organisms, *C. elegans* and *P. profundum* SS9. Fig.4.10 clearly illustrates that our model achieved the highest mean AUROC compared to all other methods. Moreover, our model's standard deviation (± 4.14) is notably smaller than the closest competitors, OperonSEQer(± 7.85) and Operon Finder(± 8.48). This observation indicates that our model exhibits greater robustness across different organisms.

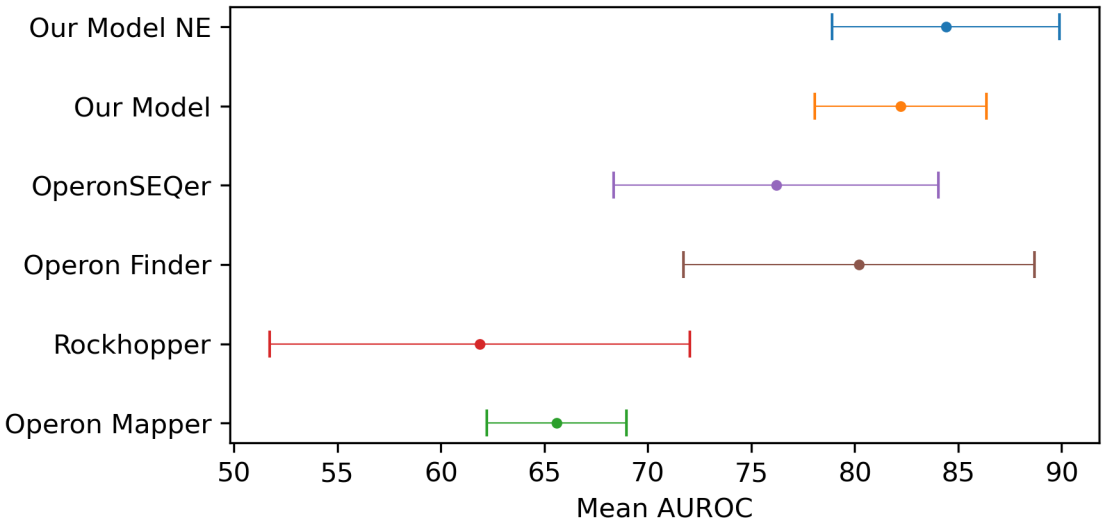

Figure 4.10: Mean AUROC. The dot indicates the mean AUROC and the lines are the standard deviation of the evaluated methods over seven training organisms and two validation organisms for which predictions were obtained from all methods.

Moreover, we employed the Friedman non-parametric statistical test that is suitable for comparing multiple classifiers over multiple datasets to assess the statistical significance of performance differences between the models [59]. The Friedman test yielded a p-value of 0.0001, indicating that the mean AUROC obtained by certain classifiers significantly deviates from the others. We conducted pairwise comparisons using the Nemenyi post-hoc test [59] to further investigate this difference.

Since the Nemenyi test operates on ranks, we ranked each model based on the mean AUROC obtained on all training organisms and two validation organisms, *C. elegans* and *P. profundum* SS9. The model with the highest AUROC was assigned rank one, while ties received the same rank (Table 4.10).

Organism	Operon Mapper	Rockhopper	Operon Finder	OperonSEQer	Our Model
C. elegans	5	3	5	2	1
P. profundum SS9	4	5	3*	2	1
E. coli K-12 substr. MG1655	5*	4*	1*	3*	2
C. glutamicum ATCC 13032	4	5	2*	3	1
L. monocytogenes EDG-e	3	5	1*	4	2
L. pneumophila str. Paris	4	5	2	3	1
H. pylori 26695	4	5*	1	3	2
B. subtilis subsp. subtilis str. 168	4*	5*	1*	2*	3
M. pneumoniae M129	3	5	4	2	1

Table 4.10: Rank of AUROC of the evaluated methods over seven training organisms and two validation organisms. The highlighted numbers in the tables represent the best rank per organism among different methods. * indicates that the target organism is included in the training data of the corresponding method.

Upon applying the Nemenyi test, we observed that three groups of classifiers with similar ranks emerged, as illustrated in Fig.4.11. According to the Nemenyi test results, both Operon Finder and Our Model demonstrated statistically significantly lower ranks (p-value = 0.0001) than the other three models' ranks. The critical difference boundary is at a mean rank of 2.156.

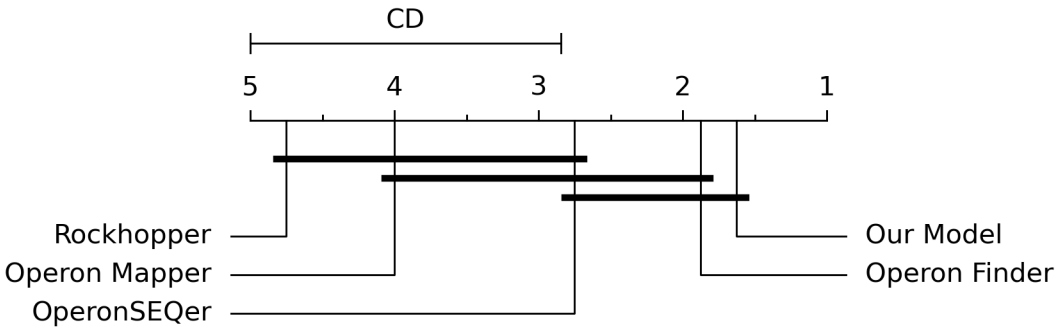

Figure 4.11: Critical Difference plot, over seven training organisms and two validation organisms. Classifiers that do not show significant differences according to the Nemenyi test at a significance level of 0.05 are connected with a horizontal line. The methods with the best performance are to the right.

Based on these results, we have found significant differences in ranks between our model and the three other classifiers, Operon Mapper, Rockhopper and OperonSEQer. While Operon Finder showed comparable performance, it comes with the limitation of predicting only specific types of species due to its reliance on external data source features. Additionally, our model exhibited the highest mean AUROC with a relatively low standard deviation, indicating more consistent performance across different

datasets. As a result, we can confidently conclude that our model outperforms the other classifiers in terms of AUROC in the task of operon detection.

4.4 Code and data availability

The code used in this research and the datasets utilized for training and evaluation are openly available on GitHub at `https://github.com/BioinformaticsLabAtMUN/OpDetect`. By making the code and data publicly accessible, we aim to promote transparency and reproducibility in scientific research. Users can freely explore the implementation details of our model, replicate the experiments, and further contribute to the advancement of operon detection and deep learning methodologies in genomics. This open availability of resources will foster collaboration and facilitate the development of more robust and accurate models for gene-related tasks.

Chapter 5

Conclusion

The contributions made by this thesis to the field of operon detection are:

- **Utilization of Accessible Input Data:** Our approach exclusively relies on genome annotations and RNA-seq reads, avoiding the need for external data source features. This ensures a more accessible process for operon detection.

- **Systematic Assessment of Models:** We conducted a thorough comparison with recent best-performing models available for operon detection, highlighting the strengths and advantages of our proposed approach.

- **Effective Feature Representation:** We have proposed a new feature representation for RNA-seq data, which draws inspiration from signal processing techniques. This approach allows for broader and more effective application of deep learning techniques in the context of operon detection.

- **CNN-LSTM Architecture:** We employed a CNN-LSTM architecture, which has never been applied to operon detection before. This unique combination harnesses the respective strengths of both networks, enabling us to effectively capture both spatial and temporal dependencies in the RNA-seq data.

- **Superior Performance:** Our model outperforms previous methods for operon detection in terms of recall, f1-score and AUROC, demonstrating its efficacy in identifying co-transcribed gene pairs.

- **Species Agnostic Model:** Our model's versatility extends even beyond bacterial genomes by detecting operons in *C. elegans*.

However, one significant limitation is the relatively limited number of experimentally verified operon pairs and non-operon pairs available in the current dataset. Increasing the data available could increase our model's robustness and reliability.

For future work, we propose exploring the inclusion of promoter and terminator data, as they have been reported to have significant influence [4]. Additionally, we recommend exploring the application of transfer learning techniques in operon detection. Investigating alternative architectures by replacing the CNN and LSTM layers with pre-trained models designed explicitly for sequence analysis tasks could also yield valuable improvements..

In conclusion, this thesis opens up exciting possibilities for more accurate and widely applicable operon detection methods, paving the way for advancements in understanding gene regulation and interactions in various biological systems.

Bibliography

[1] Citlalli Mejía-Almonte, Stephen J.W. Busby, Joseph T. Wade, Jacques van Helden, Adam P. Arkin, Gary D. Stormo, Karen Eilbeck, Bernhard O. Palsson, James E. Galagan, and Julio Collado-Vides. Redefining fundamental concepts of transcription initiation in bacteria. *Nature Reviews Genetics*, 21(11):699–714, 2020.

[2] Anne E. Osbourn and Ben Field. Operons. *Cellular and Molecular Life Sciences*, 66(23):3755–3775, 2009.

[3] Rida Assaf, Fangfang Xia, and Rick Stevens. Detecting operons in bacterial genomes via visual representation learning. *Scientific Reports*, 11(1):1–10, 2021.

[4] Tyrrell Conway, James P. Creecy, Scott M. Maddox, Joe E. Grissom, Trevor L. Conkle, Tyler M. Shadid, Jun Teramoto, Phillip San Miguel, Tomohiro Shimada, Akira Ishihama, Hirotada Mori, and Barry L. Wanner. Unprecedented high-resolution view of bacterial operon architecture revealed by RNA sequencing. *mBio*, 5(4), 2014.

[5] James P. Creecy and Tyrrell Conway. Quantitative bacterial transcriptomics with RNA-seq. *Current Opinion in Microbiology*, 23:133–140, 2015.

[6] Rutger W.W. Brouwer, Oscar P. Kuipers, and Sacha A.F.T. Van Hijum. The relative value of operon predictions. *Briefings in Bioinformatics*, 9(5):367–375, 2008.

[7] Brian Tjaden. A computational system for identifying operons based on RNA-seq data. *Methods*, 176(April 2019):62–70, 2020.

[8] Andrew S. Lang, Olga Zhaxybayeva, and J. Thomas Beatty. Gene transfer agents: Phage-like elements of genetic exchange. *Nature Reviews Microbiology*, 10(7):472–482, 2012.

[9] Brian J Arnold, I Huang, William P Hanage, et al. Horizontal gene transfer and adaptive evolution in bacteria. *Nature Reviews Microbiology*, pages 1–13, 2021.

[10] Andrew S. Lang, Alexander B. Westbye, and J. Thomas Beatty. The Distribution, Evolution, and Roles of Gene Transfer Agents in Prokaryotic Genetic Exchange. *Annual Review of Virology*, 4:87–104, 2017.

[11] Raga Krishnakumar and Anne M. Ruffing. OperonSEQer: A set of machine-learning algorithms with threshold voting for detection of operon pairs using short-read RNA-sequencing data. *PLoS Computational Biology*, 18(1):1–18, 2022.

[12] Rachel Haller, Meghann Kennedy, Nick Arnold, and Robert Rutherford. The transcriptome of Mycobacterium tuberculosis. *Applied Microbiology and Biotechnology*, 86(1):1–9, 2010.

[13] Nicolas Sierro, Yuko Makita, Michiel De hoon, and Kenta Nakai. DBTBS: A database of transcriptional regulation in *Bacillus subtilis* containing upstream intergenic conservation information. *Nucleic Acids Research*, 36(SUPPL. 1):93–96, 2008.

[14] Shujiro Okuda and Akiyasu C. Yoshizawa. ODB: A database for operon organizations, 2011 update. *Nucleic Acids Research*, 39(SUPPL. 1):552–555, 2011.

[15] Alberto Santos-Zavaleta, Mishael Sánchez-Pérez, Heladia Salgado, David A. Velázquez-Ramírez, Socorro Gama-Castro, Víctor H. Tierrafría, Stephen J.W. Busby, Patricia Aquino, Xin Fang, Bernhard O. Palsson, James E. Galagan, and Julio Collado-Vides. A unified resource for transcriptional regulation in *Escherichia coli* K-12 incorporating high-throughput-generated binding data into RegulonDB version 10.0. *BMC Biology*, 16(1):1–12, 2018.

[16] Socorro Gama-Castro, Verónica Jiménez-Jacinto, Martín Peralta-Gil, Alberto Santos-Zavaleta, Mónica I. Peñaloza-Spinola, Bruno Contreras-Moreira, Juan Segura-Salazar, Luis Muñiz-Rascado, Irma Martínez-Flores, Heladia Salgado, César Bonavides-Martínez, Cei Abreu-Goodger, Carlos Rodríguez-Penagos, Juan Miranda-Ríos, Enrique Morett, Enrique Merino, Araceli M. Huerta, Luis

Treviño-Quintanilla, and Julio Collado-Vides. RegulonDB (version 6.0): Gene regulation model of *Escherichia coli* K-12 beyond transcription, active (experimental) annotated promoters and Textpresso navigation. *Nucleic Acids Research*, 36(SUPPL. 1):120–124, 2008.

[17] Li Yeh Chuang, Hsueh Wei Chang, Jui Hung Tsai, and Cheng Hong Yang. Features for computational operon prediction in prokaryotes. *Briefings in Functional Genomics*, 11(4):291–299, 2012.

[18] Brian T. Wilhelm, Samuel Marguerat, Ian Goodhead, and Jürg Bähler. Defining transcribed regions using RNA-seq. *Nature Protocols*, 5(2):255–266, 2010.

[19] Zhong Wang, Mark Gerstein, and Michael Snyder. Rna-seq: a revolutionary tool for transcriptomics. *Nature reviews genetics*, 10(1):57–63, 2009.

[20] Chiara Sabatti, Lars Rohlin, Min Kyu Oh, and James C. Liao. Co-expression pattern from DNA microarray experiments as a tool for operon prediction. *Nucleic Acids Research*, 30(13):2886–2893, 2002.

[21] Brian Tjaden, David R. Haynor, Sergey Stolyar, Carsten Rosenow, and Eugene Kolker. Identifying operons and untranslated regions of transcripts using Escherichia coli RNA expression analysis. *Bioinformatics (Oxford, England)*, 18:S337–344, 2002.

[22] Morgan N. Price, Katherine H. Huang, Eric J. Alm, and Adam P. Arkin. A novel

method for accurate operon predictions in all sequenced prokaryotes. *Nucleic Acids Research*, 33(3):880–892, 2005.

[23] Cynthia M. Sharma, Steve Hoffmann, Fabien Darfeuille, Jérémy Reignier, Sven Findeiß, Alexandra Sittka, Sandrine Chabas, Kristin Reiche, Jörg Hackermüller, Richard Reinhardt, Peter F. Stadler, and Jörg Vogel. The primary transcriptome of the major human pathogen *Helicobacter pylori*. *Nature*, 464(7286):250–255, 2010.

[24] Vittorio Fortino, Olli Pekka Smolander, Petri Auvinen, Roberto Tagliaferri, and Dario Greco. Transcriptome dynamics-based operon prediction in prokaryotes. *BMC Bioinformatics*, 15(1), 2014.

[25] Xizeng Mao, Qin Ma, Chuan Zhou, Xin Chen, Hanyuan Zhang, Jincai Yang, Fenglou Mao, Wei Lai, and Ying Xu. DOOR 2.0: Presenting operons and their functions through dynamic and integrated views. *Nucleic Acids Research*, 42(D1):654–659, 2014.

[26] Jelle Slager, Rieza Aprianto, and Jan Willem Veening. Deep genome annotation of the opportunistic human pathogen *Streptococcus pneumoniae* D39. *Nucleic Acids Research*, 46(19):9971–9989, 2018.

[27] Blanca Taboada, Karel Estrada, Ricardo Ciria, and Enrique Merino. Operon-mapper: A web server for precise operon identification in bacterial and archaeal genomes. *Bioinformatics*, 34(23):4118–4120, 2018.

[28] Blanca Taboada, Ricardo Ciria, Cristian E. Martinez-Guerrero, and Enrique Merino. ProOpDB: Prokaryotic operon database. *Nucleic Acids Research*, 40(D1):627–631, 2012.

[29] Blanca Taboada, Cristina Verde, and Enrique Merino. High accuracy operon prediction method based on STRING database scores. *Nucleic Acids Research*, 38(12), 2010.

[30] Lars J. Jensen, Michael Kuhn, Manuel Stark, Samuel Chaffron, Chris Creevey, Jean Muller, Tobias Doerks, Philippe Julien, Alexander Roth, Milan Simonovic, Peer Bork, and Christian von Mering. STRING 8 - A global view on proteins and their functional interactions in 630 organisms. *Nucleic Acids Research*, 37(SUPPL. 1):412–416, 2009.

[31] Alice R. Wattam, David Abraham, Oral Dalay, Terry L. Disz, Timothy Driscoll, Joseph L. Gabbard, Joseph J. Gillespie, Roger Gough, Deborah Hix, Ronald Kenyon, Dustin MacHi, Chunhong Mao, Eric K. Nordberg, Robert Olson, Ross Overbeek, Gordon D. Pusch, Maulik Shukla, Julie Schulman, Rick L. Stevens, Daniel E. Sullivan, Veronika Vonstein, Andrew Warren, Rebecca Will, Meredith J.C. Wilson, Hyun Seung Yoo, Chengdong Zhang, Yan Zhang, and Bruno W. Sobral. PATRIC, the bacterial bioinformatics database and analysis resource. *Nucleic Acids Research*, 42(D1):581–591, 2014.

[32] Jeremy Howard and Sylvain Gugger. fastai documentation. `https://docs.fast.ai/`, 2022.

[33] Sayash Kapoor and Arvind Narayanan. Leakage and the reproducibility crisis in ML-based science. *arXiv preprint arXiv:2207.07048*, 2022.

[34] Tejasvi Singh Tomar, Pratik Dasgupta, and Shankar Prasad Kanaujia. Operon finder: a deep learning-based web server for accurate prediction of prokaryotic operons. *Journal of Molecular Biology*, page 167921, 2022.

[35] William H Kruskal and W Allen Wallis. Use of ranks in one-criterion variance analysis. *Journal of the American statistical Association*, 47(260):583–621, 1952.

[36] Paramvir S Dehal, Marcin P Joachimiak, Morgan N Price, John T Bates, Jason K Baumohl, Dylan Chivian, Greg D Friedland, Katherine H Huang, Keith Keller, Pavel S Novichkov, et al. Microbesonline: an integrated portal for comparative and functional genomics. *Nucleic acids research*, 38(suppl_1):D396–D400, 2010.

[37] Daniel H. Haft, Michael DiCuccio, Azat Badretdin, Vyacheslav Brover, Vyacheslav Chetvernin, Kathleen O'Neill, Wenjun Li, Farideh Chitsaz, Myra K. Derbyshire, Noreen R. Gonzales, Marc Gwadz, Fu Lu, Gabriele H. Marchler, James S. Song, Narmada Thanki, Roxanne A. Yamashita, Chanjuan Zheng, Françoise Thibaud-Nissen, Lewis Y. Geer, Aron Marchler-Bauer, and Kim D. Pruitt. Refseq: An update on prokaryotic genome annotation and curation. *Nucleic Acids Research*, 46(D1):D851–D860, 2018.

[38] Rasko Leinonen, Hideaki Sugawara, and on behalf of the International Nucleotide Sequence Database Collaboration Shumway, Martin. The sequence read archive. *Nucleic Acids Research*, 39(suppl-1):D19–D21, 11 2010.

[39] Rasko Leinonen, Ruth Akhtar, Ewan Birney, Lawrence Bower, Ana Cerdeno-Tárraga, Ying Cheng, Iain Cleland, Nadeem Faruque, Neil Goodgame, Richard Gibson, et al. The european nucleotide archive. *Nucleic acids research*, 39(suppl_1):D28–D31, 2010.

[40] Shifu Chen, Yanqing Zhou, Yaru Chen, and Jia Gu. Fastp: An ultra-fast all-in-one FASTQ preprocessor. *Bioinformatics*, 34(17):i884–i890, 2018.

[41] Daehwan Kim, Joseph M. Paggi, Chanhee Park, Christopher Bennett, and Steven L. Salzberg. Graph-based genome alignment and genotyping with HISAT2 and HISAT-genotype. *Nature Biotechnology*, 37(8):907–915, 2019.

[42] Petr Danecek, James K Bonfield, Jennifer Liddle, John Marshall, Valeriu Ohan, Martin O Pollard, Andrew Whitwham, Thomas Keane, Shane A McCarthy, Robert M Davies, and Heng Li. Twelve years of SAMtools and BCFtools. *GigaScience*, 10(2), 02 2021. giab008.

[43] Aaron R. Quinlan and Ira M. Hall. BEDTools: a flexible suite of utilities for comparing genomic features. *Bioinformatics*, 26(6):841–842, 01 2010.

[44] Zhuoran Jiang, Chao Wang, Zixin Wu, Kun Chen, Wei Yang, Hexiang Deng, Heng Song, and Xiang Zhou. Enzymatic deamination of the epigenetic nucle-

oside N6-methyladenosine regulates gene expression. *Nucleic Acids Research*, 49(21):12048–12068, 2021.

[45] Joerg Stuelke Patrick Faßhauer and Tobias Busche. RNAseq of *Bacillus subtilis* wildtype and cspB-cspD deletion mutant reveals that the lack of the cold shock proteins CspB and CspD affects the expression of about 20% of all genes. `https://www.ebi.ac.uk/biostudies/arrayexpress/studies/E-MTAB-10658`, 2021.

[46] Marc Keppel, Max Hünnefeld, Andrei Filipchyk, Ulrike Viets, Cedric Farhad Davoudi, Aileen Krüger, Christina Mack, Eugen Pfeifer, Tino Polen, Meike Baumgart, Michael Bott, and Julia Frunzke. HrrSA orchestrates a systemic response to heme and determines prioritization of terminal cytochrome oxidase expression. *Nucleic Acids Research*, 48(12):6547–6562, 2020.

[47] Tobias Busche Matthias Ruwe and Jörn Kalinowski. Transcriptional analysis of regulatory effects in *Corynebacterium glutamicum* and a derived (p)ppGpp0 mutant strain. `https://www.ebi.ac.uk/biostudies/arrayexpress/studies/E-MTAB-8070`, 2019.

[48] Donghyuk Kim, Sang Woo Seo, Ye Gao, Hojung Nam, Gabriela I. Guzman, Byung Kwan Cho, and Bernhard O. Palsson. Systems assessment of transcriptional regulation on central carbon metabolism by Cra and CRP. *Nucleic Acids Research*, 46(6):2901–2917, 2018.

[49] Aurelie Guyet, Martyn Dade-Robertson, Anil Wipat, John Casement, Wendy

Smith, Helen Mitrani, and Meng Zhang. Mild hydrostatic pressure triggers oxidative responses in Escherichia coli. *PLoS ONE*, 13(7):1–19, 2018.

[50] Sumith Kumar, Bipul C. Karmakar, Deepesh Nagarajan, Asish K. Mukhopadhyay, Richard D. Morgan, and Desirazu N. Rao. N4-cytosine DNA methylation regulates transcription and pathogenesis in Helicobacter pylori. *Nucleic Acids Research*, 46(7):3429–3445, 2018.

[51] Xavier Charpentier. RNAseq profiling of a lpp1663 mutant compared to its isogenic parent *Legionella pneumophila* strain Paris. `https://www.ebi.ac.uk/biostudies/arrayexpress/studies/E-MTAB-4095`, 2020.

[52] Kemal Avican, Jehad Aldahdooh, Matteo Togninalli, A. K.M.Firoj Mahmud, Jing Tang, Karsten M. Borgwardt, Mikael Rhen, and Maria Fällman. RNA atlas of human bacterial pathogens uncovers stress dynamics linked to infection. *Nature Communications*, 12(1), 2021.

[53] Maria Lluch-Senar Luis Serrano, Raul Burgos and Marc Weber. RNA-seq of Lon and FtsH conditional mutants of *Mycoplasma pneumoniae* under depleting and inducing conditions. `https://www.ebi.ac.uk/biostudies/arrayexpress/studies/E-MTAB-8537`, 2020.

[54] Stefano Campanaro, Fabio D. Pascale, Andrea Telatin, Riccardo Schiavon, Douglas H. Bartlett, and Giorgio Valle. The transcriptional landscape of the deep-sea

bacterium Photobacterium profundum in both a toxR mutant and its parental strain. *BMC Genomics*, 13(1):1, 2012.

[55] Bacteroid Differentiation, Suboptimal Symbiotic, Quentin Nicoud, Florian Lamouche, Anaïs Chaumeret, Thierry Balliau, Le Bars, Mickaël Bourge, and Fabienne Pierre. Bradyrhizobium diazoef fi ciens USDA110 Nodulation of Aeschynomene afraspera Is Associated with Atypical Terminal. *mSystems*, 6(3):1–19, 2021.

[56] Satya P. Singh, Madan Kumar Sharma, Aimé Lay-Ekuakille, Deepak Gangwar, and Sukrit Gupta. Deep ConvLSTM with Self-Attention for Human Activity Decoding Using Wearable Sensors. *IEEE Sensors Journal*, 21(6):8575–8582, 2021.

[57] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[58] Maria D Ermolaeva, Owen White, and Steven L Salzberg. Prediction of operons in microbial genomes. *Nucleic acids research*, 29(5):1216–1221, 2001.

[59] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine learning research*, 7:1–30, 2006.