BacTermFinder: Bacteria-agnostic Comprehensive Terminator Finder using a CNN Ensemble

by

A thesis submitted to the

Department of Computer Science

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

April 2024

St. John's Newfoundland and Labrador

Abstract

Terminator is a region in the DNA that ends the transcription process. Knowing the location of bacterial terminators will lead to a better understanding of how bacteria's transcription works. This might facilitate bio-engineering and support bacterial genomic studies. Currently, multiple tools are available for predicting bacterial terminators. However, most methods are specialized for certain bacteria or terminator types. In this work, we developed BacTermFinder, a tool that utilized Deep Learning models, specifically an ensemble of Convolutional Neural Networks (CNNs), with four different genomic representations trained on 46,386 bacterial terminators identified using RNA-seq technologies. Based on our results, BacTermFinder's average recall score is significantly higher than the next best approach (0.56 ± 0.19 vs 0.45 ± 0.20) in our diverse test set of five different bacteria while reducing the number of false positives. Moreover, BacTermFinder's model identifies both types of terminators (intrinsic and factor-dependent) and even generalizes to Archea. BacTermFinder is publicly available at https://github.com/BioinformaticsLabAtMUN/BacTermFinder

Acknowledgement

I am deeply grateful for Dr. Lourdes Pena-Castillo's efforts in this work. She was supportive, understanding and the best supervisor I could have for my Master's.

I also want to thank my labmates and friends who helped me during this research, especially Fatemeh (Mahta) Ghorbani, for her invaluable insights and support. I would also like to thank you, Rezvan Karaji, Gavin Hull, and all of my labmates in the Bioinformatics Lab at MUN (https://www.cs.mun.ca/ lourdes/public/) for their assistance during this project.

This research was partly enabled by support provided by Acenet (ace-net.ca) and the Digital Research Alliance of Canada (alliance can .ca). Also, funding for my Master's was partially provided by Memorial University's School of Graduate Studies (SGS). Figure 1.1 and Figure 1.3 are reproduced with permission from Springer Nature with the order number 5770250166265, which was taken from Santangelo and coworkers review of bacterial termination [1].

Contents

List of Figures

List of Tables

List of Acronyms

Chapter 1

Introduction

In response to environmental changes, bacterial cells continuously adjust transcription. Transcription is the process of copying DNA into RNA. Certain RNAs will then be translated into proteins. Transcription starts at the point the promoter sets, and terminators will stop the transcription. A terminator in DNA is a stretch of nucleic acid that indicates the end of a gene or operon[2].

The termination of transcription is a process crucial for the accurate synthesis of RNA. In prokaryotes, such as *Escherichia coli*, termination can occur through either factor-dependent or factor-independent mechanisms, with the latter known as Rho-independent termination or intrinsic termination. In *E. coli* intrinsic terminator is predominant, constituting approximately 70-80% of all terminations[3].

1.1 Intrinsic Terminator

Intrinsic termination involves forming a distinctive RNA hairpin structure followed by a uridine sequence, which acts as an intrinsic terminator, as shown in Figure 1.1. This terminator structure plays an important role in preventing backtracking, a phenomenon that stabilizes the elongation complex during transcription [2].

Specific sequences in the DNA template strand mark the Rho-independent terminators. When the RNA polymerase reaches the end of the transcribed gene, it reaches a region rich in Cytosine (C) and Guanine (G) nucleotides. The RNA created from this region binds with itself, and the C and G nucleotides pair together. A stable hairpin structure is the result of this process. The hairpin causes the polymerase to stall. The weak base pairing between the Adenine nucleotides of the DNA template and the uridine nucleotides of the RNA transcript let the transcript to detach from the template thus, terminating the transcription[1].

Extensive studies on termination mechanisms have revealed critical elements in the termination process. The 3'-terminal U of U-tract has been identified as essential for pausing at the termination site and the overall termination process. Mutations in this region not only prevent pausing but also eliminate termination. This emphasizes the crucial role of pausing at the U-tract for terminator formation [4].

Further analysis of terminator structures in vivo has shown that the most robust terminators possess near-perfect U-tracts associated with AT-rich downstream DNA sequences [5]. Additionally, A-tracts immediately upstream of the terminator con-

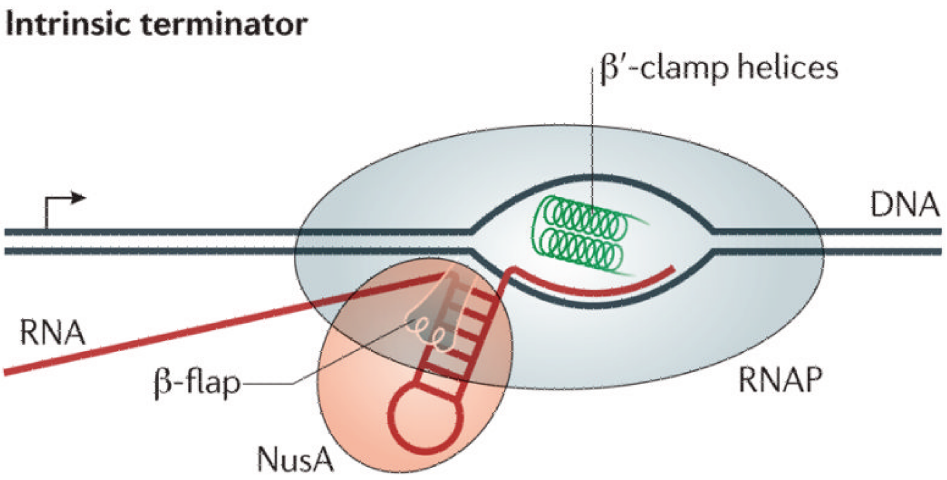

Figure 1.1: Intrinsic terminator transcription [1]. Reproduced with permission from Springer Nature.

tribute to increased termination efficiency, likely through complementarity with the U-tract. The A–U-tract pairing extends the terminator hairpin duplex, leading to the complete elimination of the RNA–DNA hybrid and further facilitating termination [5].

1.2 Factor-Dependent Terminator

Multiple factors in the cell may affect termination, including transcriptional factors that can bind Elongation Complex (EC) and modulate their stability at various steps along the termination pathway. Rho factor (Rho protein) plays a vital role in Rho-dependent termination. The terminator has the sequence for the Rho binding in the mRNA and the transcription stop point. For the Rho protein to catch up with RNA

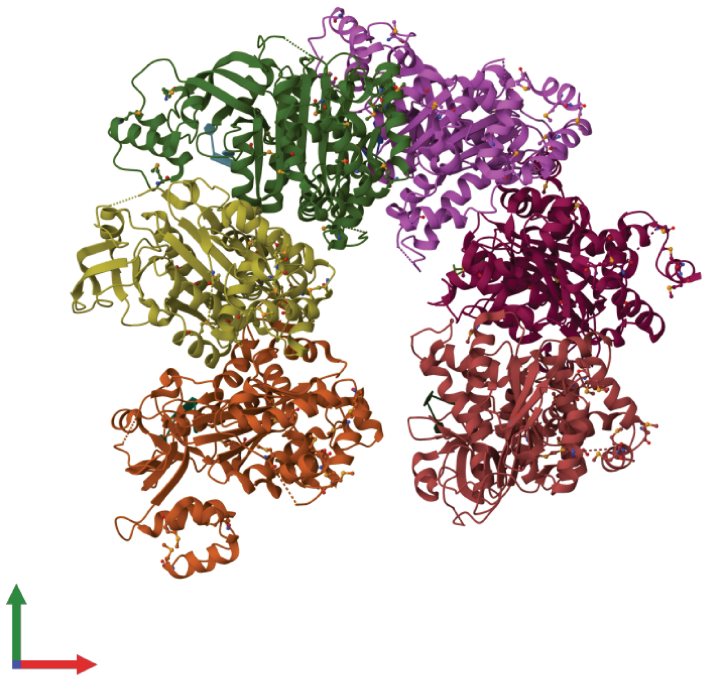

Figure 1.2: Top view of Rho protein. Image released to the public domain and available at the Protein Data Bank in Europe https://www.ebi.ac.uk/pdbe/entry/pdb/1pv4

polymerase, the sequence stalls the RNA polymerase [6]. Rho protein is a ring-shaped protein complex that binds to and travels along RNA strands, as shown in Figure 1.2 [7]. Rho attaches to the Rho binding site and moves up the RNA transcript in the 5' to 3' direction, approaching the polymerase. When Rho catches up to the polymerase, it causes the transcript to be released, thus terminating transcription, as shown in Figure 1.3.

Rho is controlled both positively and negatively by NusG. During elongation,

Rho binds RNAP and forms a pre-termination complex (PTC) with NusA and NusG [8]. PTC samples nascent transcripts continuously for a termination signal to trap the elongation complex before dissociating. In a strong intrinsic termination, Rho's presence did not change the efficiency of the termination, meaning that Rho does not affect strong intrinsic terminators [3].

Rho Utilization Site (RUT) sequences are upstream of the termination site and are approximately 80–90 nt long, allowing RNA to bind to each of Rho's six monomers [2]. Nearly all bacteria, with the exception of *Cyanobacteria*, *Negativicutes*, and *Streptococcaceae*, utilize the transcription termination factor Rho for the purpose of separating transcription units and regulating global gene expression [8, 9, 10, 11].

Recent research on the model Gram-positive bacterium *B. subtilis* produced an atlas of terminators. Mandell and co-authors found that NusA and NusG encourage some of the intrinsic terminators, and they also participate in Rho-dependent terminations. Their *in-vitro* research, merged with computational methods, showed that intrinsic terminators with weak hairpin secondary structure or weak T-tail are affected by Rho in a way that Rho helps the termination by preventing unhelpful RNA secondary structures. They also found some intrinsic terminators more effective when stimulated by Rho [12].

The accuracy of predicting terminators for genome annotation is crucial for biotechnology applications. Traditional biological experiments to identify terminators are time and labour-intensive. So, machine learning has been adopted as an efficient methodology for identifying terminator sequences [13].

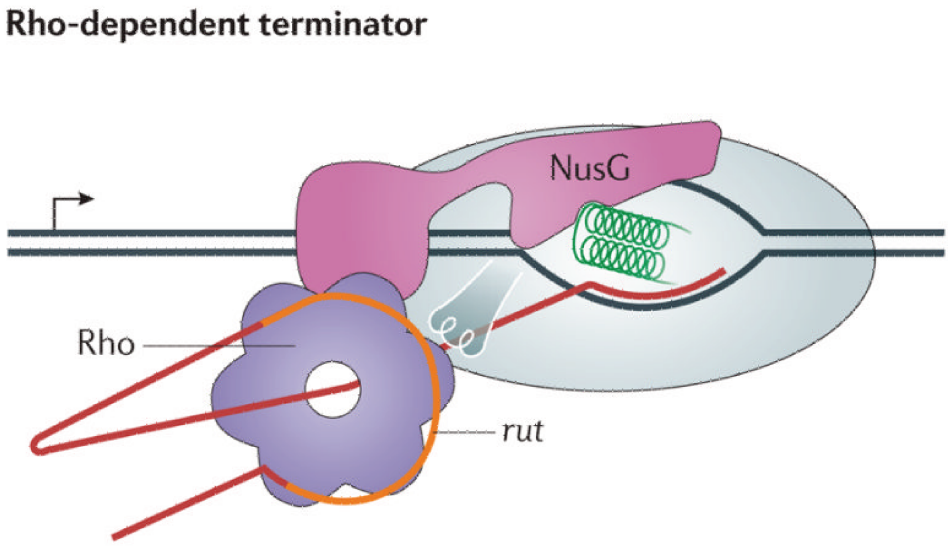

Figure 1.3: Rho-dependent termination of transcription[1]. Reproduced with permission from Springer Nature

In this thesis, our goal is to gain insights into bacterial transcription termination while developing a computational terminator prediction tool. To do that, we looked at existing computational approaches to predict terminators, which was done through our literature search discussed in Chapter 2. Later, we identified gaps in the existing methods and planned to overcome those in Chapter 3 by collecting data first and exploring different Machine Learning techniques. Our results are explained in Chapter 4, and insights into bacterial terminators are discussed there. At last, we conclude in Chapter 5 to summarize what we have done and point out some limitations and future extensions of our current method (BacTermFinder).

Chapter 2

Related Works

This Chapter describes some of the existing computational approaches for bacterial terminator prediction. These works are sorted in reverse chronological order. For an approach to be included in our search, it should be focused on finding bacterial terminators computationally (in-silico) using machine-learning algorithm or any sort of dynamic programming to find the terminators. Additionally, we restrict our literature search to works published from 2007 onwards, as most transcription terminator prediction software was published after this year.

2.1 TermNN

TermNN [14] is a neural-network-based approach focused on intrinsic terminator prediction. TermNN introduced a pre-training approach to mitigate the lack of high-volume data. The pre-training approach utilizes inverse folding to pre-train a neural network and then continue training on intrinsic terminator data consisting of 1175

terminators from 2 species. The inverse folding technique in DNA involves computationally predicting the nucleotide sequence that, when folded, forms a desired three-dimensional structure or shape. Brandenburg and co-authors showed that their pre-training on inverse fold data can increase the model's performance in intrinsic terminator prediction, and the trained model can explain the role of different parts of the intrinsic terminator structure in termination. TermNN uses deep learning methods such as Convolutional Neural Network (CNN) and Long Short-term Memory networks (LSTM) with one-hot feature encoding and base-pair matching encoding or matrix encoding (Figure 2.1). Matrix encoding is a 2D matrix (L x L) of the base pairs of sequences with a length of L. It represents the Watson-Crick pairing between different nucleotides by assigning normalized numbers for each pairing (like the A-U pair or C-G pair in RNA). TermNN outperformed decision rule-based software Arnold [15] by 0.04 points in F1-score in validation data of Monte-Carlo Cross Validation, on *E. coli* and *B. subtilis* intrinsic terminators. Arnold performed 0.88 F1-score, and the pre-trained matrix encoding TermNN scored 0.92 F1-score.

2.2 ITT prediction

In Intrinsic Transcription Termination (ITT) prediction, Gupta and colleagues [16] utilized Mfold software [17] to look for clusters of single hairpin structures at the end of transcripts identified by RNA-seq from 13 bacterial species. Gupta and co-authors' study shows hairpins concerted in clusters to form a highly effective ITT unit. They validated 81% of the ITT predictions with sequences from RNA-Seq-derived sites.

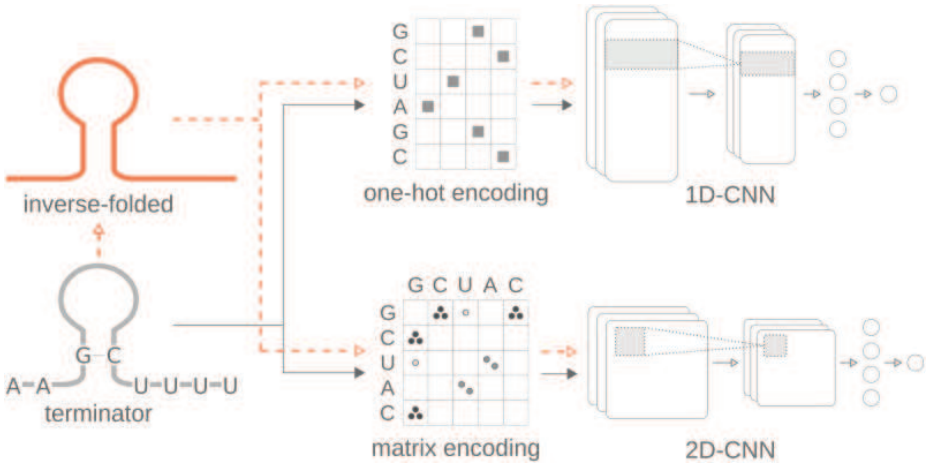

Figure 2.1: Graphical abstract of TermNN [14] indicates two CNN models pre-trained with inverse-folded sequences and fine-tuned on intrinsic terminators with two methods, one-hot encoding and matrix encoding. Figure is taken from [14] (CC-by 4.0).

Their identified locations and RNA-seq-derived locations overlap with an accuracy of 72%, with 98% of sites being located ≤ 80 bases downstream of the translational stop codon. They verified their method on 143 *E. coli* experimentally verified terminators in the inter-operon region, and 83 of these terminators were found. *E. coli* was not included in the 13 bacterial species of interest initially used for training.

2.3 iterb-PPse

Based on PseKNC I [18] and PseKNC II [19], Fan and co-authors [13] developed the "iterb-PPse" method for predicting terminators. PseKNC I & II are feature generation techniques which were used in iterb-PPse. PseKNC can generate 38 physicochemical properties for 2-mer and 12 physicochemical properties for 3-mer. iterb-PPse

was trained on *E. coli* and *B. subtilis*.

To enrich the data, they applied three new methods of feature extraction: K-PWM, base-content, and NucleotidePro, as well as a two-step feature selection method. The pipeline of their study is presented in Figure 2.2.

Five single models were compared with 16 ensemble models to identify terminators based on optimized features. They achieved 99.88% accuracy on 1615 *E. coli* intrinsic and factor-dependent terminators with their XGboost classifier. They improved over iTerm-PseKNC in accuracy by 3.8 % on average.

2.4 iTerm-PseKNC

iTerm-PseKNC [18] is a support-vector-machine-based approach to identify transcription terminators using pseudo-k-tuple nucleotide composition (PseKNC) as features. PseKNC is a feature-generation technique. Their training data consisted of 280 terminator and 560 non-terminator sequences from *E. coli*.

iTerm-PseKNC was evaluated by examining independent datasets representing experimentally confirmed intrinsic terminators for *E. coli* and *B. subtilis*. The analysis correctly identified all terminators in *E. coli* and 87.5% in *B. subtilis*. Their statistical results showed that the average lengths of terminators are around 50 bp. They published their software online (http://lin-group.cn/server/iTerm-PseKNC/).

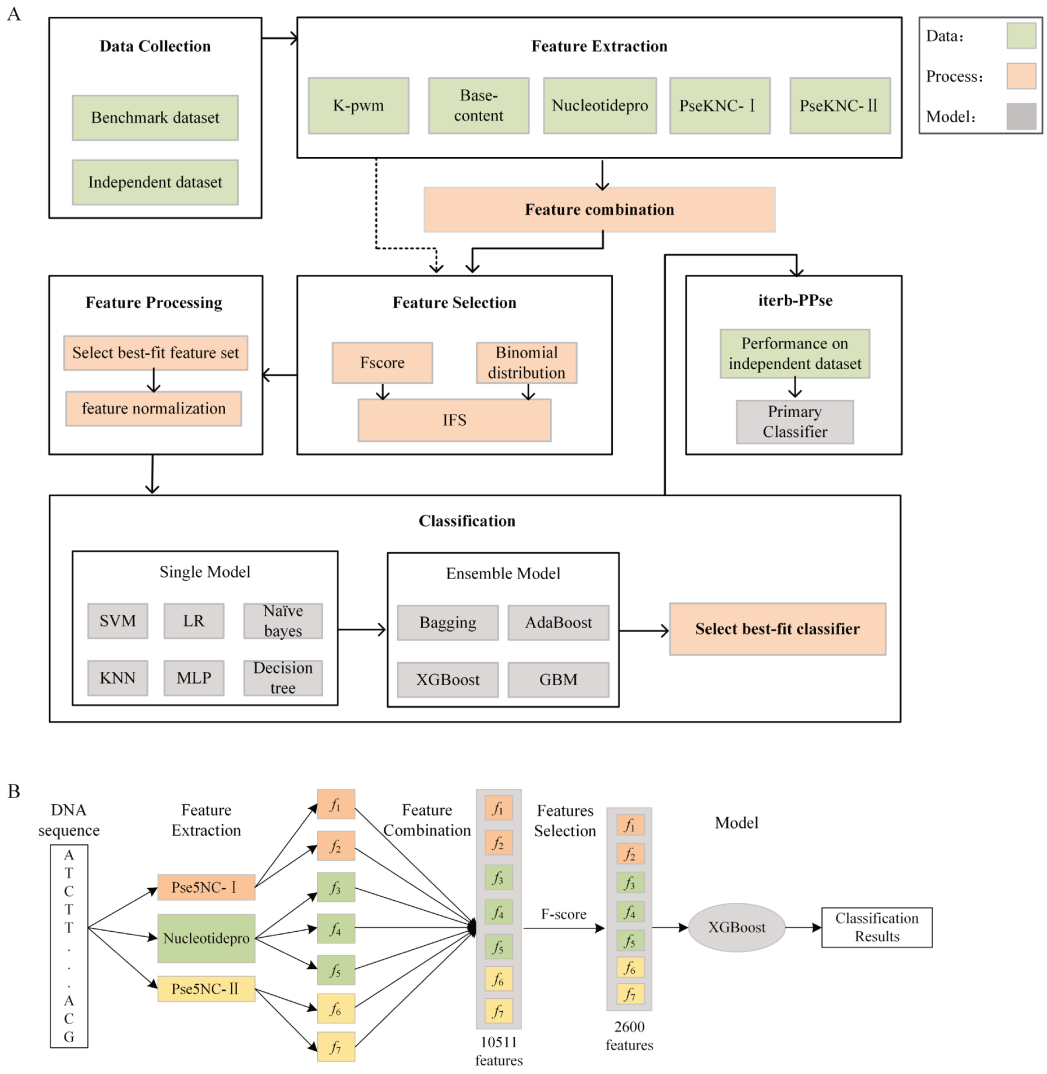

Figure 2.2: Pipeline of iterb-PPse, which contains data collection, feature extraction and combination and selection, and classification with ensemble models. A demonstrates the pipeline, and B is the pipeline of iTerm-PseKNC [18]. Figure taken from iterb-PPse paper [13] (CC BY 4.0).

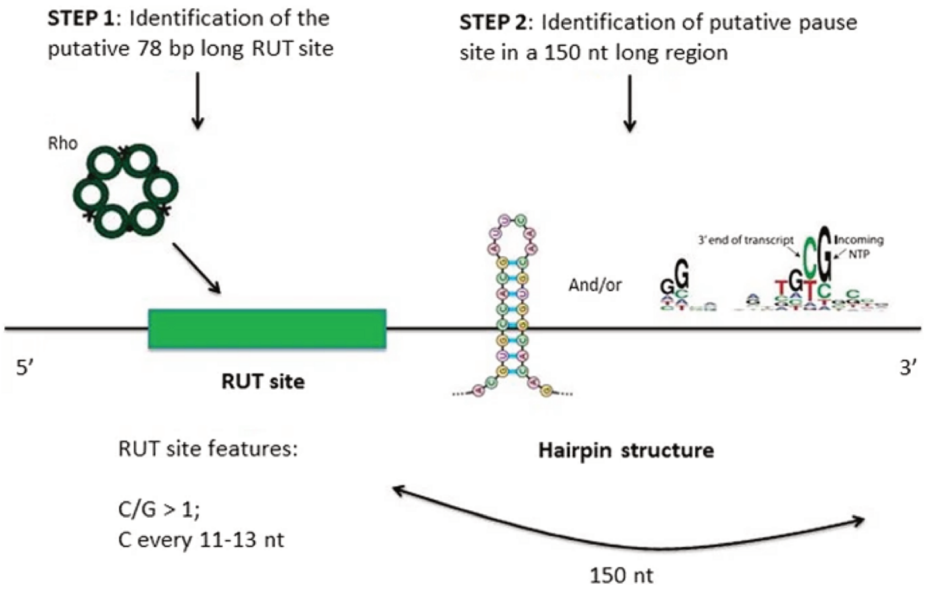

Figure 2.3: Depiction of Rho utilization site (RUT) and RNA Polymerase (RNAP), which the RhoTermPredict method is looking for in the genome. Figure taken from [20] (CC BY 4.0).

2.5 RhoTermPredict

RhoTermPredict [20] is an algorithm for predicting Rho-dependent terminators in bacterial genomes based on data from *E. coli, S. enterica* and *B. subtilis*. The RhoTermPredict algorithm identifies transcription termination motifs based on a previously proposed consensus motif [2]. As shown in Figure 2.3, the method looks for a 78 nt long RUT site that contains a C > G content and regularly spaced C residues, followed by a putative pause site for RNA polymerase. Di Salvo and co-authors achieved a 0.7 F1 score on 1298 experimentally verified terminators from the above-mentioned bacteria.

2.6 OPLS - DA

Nadiras and colleagues[21] used a 104 set of in-vitro termination data of *E. coli MG1655* and *S. enterica LT2* to develop an Orthogonal Projections to Latent Structures Discriminant Analysis (OPLS-DA) prediction method for Rho-dependent terminators. Using jackknife cross-validation, they trained a 3-class classifier of "None", "Weak", and "Strong" terminators. Nadiras and co-authors identified new factor-dependent signals and quantitative sequence descriptors with significant predictive value in biochemical and OPLS-DA analysis of previously uncharacterized genomic sequences. Descriptors relevant to Rho-RNA interaction include C>G skewness, secondary structure, and richness in 6'-carbons and 5'-cysteine dinucleotides regularly spaced. A collection of OPLS-DA descriptors yielded 85% accuracy in predicting factor-dependent termination on 7-fold jackknife cross-validation on *E. coli MG1655* and *S. enterica LT2*.

2.7 PASIFIC

PASIFIC (Prediction of Alternative Structures for the Identification of Cis-regulation)[22] is an algorithm based on Machine Learning that predicts if a gene has the two alternative structures characteristic of riboregulators employing conditional termination, given a 5'UTR of a gene. Effectively, it searches for terminator-antiterminator alternative structures. An anti-terminator will fold with one stem of the terminator and hinder the terminator's secondary structure from forming. An anti-antiterminator

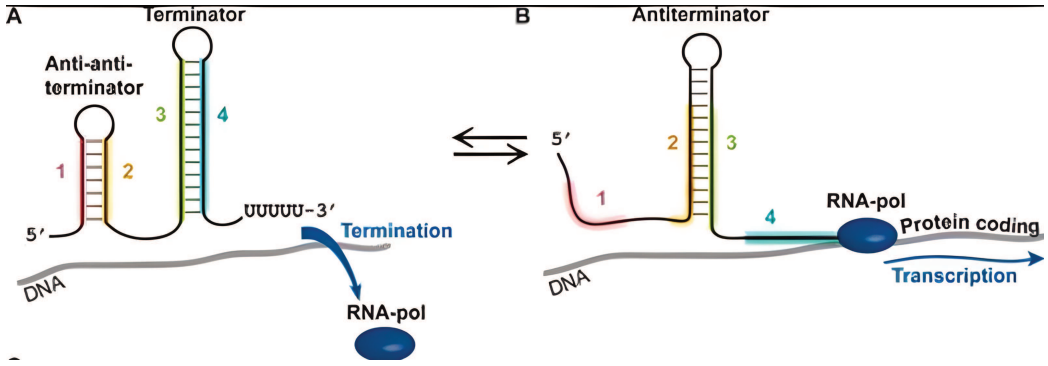

Figure 2.4: Structures of anti-antiterminator and terminator. 2.7.

Figure by [22] (CC-BY 4.0).

will cripple the anti-terminator by folding with it. It frees up the terminator stem to form a secondary structure that ends with poly U.

Their positive class included 312 regulators that had an intrinsic terminator belonging to 89 bacteria from Term-Seq [23]. Millman and co-authors achieved a high specificity of 80.6% and sensitivity of 82.5% with their Random Forest classifier on test data, which was 20% of the whole data. With the PASIFIC web application, new riboswitches and attenuators can be identified within the bacterial pangenome.

2.8 RNIE

RNIE [24] is a probabilistic approach to predict intrinsic termination. RNIE performance is 0.75 in Matthews Correlation Coefficient (MCC) against 485 terminators of experimentally verified *E. coli* and *B. subtilis*. RNIE's *M. tuberculosis* intrinsic terminator predictions comprise 80-90% of all highly structured regions near *M. tuberculosis* gene termini. The software, predictions, and alignments are open-source

and available at https://github.com/ppgardne/RNIE.

2.9 TransTermHP

TransTermHP [25] is a dynamic programming intrinsic terminator finder in bacteria. The program goes through the genome and uses folding techniques to find a stable secondary structure followed by a thymine-rich region. They calculate a score for each hairpin secondary structure based on structural features like the length of the T-rich and A-rich regions and feed that information into another function that outputs the terminator quality.

2.10 Summary

This section reviewed different methods for predicting terminators in bacteria to find what research gaps and limitations existed in the current approaches. Table 2.1 overviews the approaches reviewed in this section. Most of these tools focused on a few bacterial species or were only suitable for predicting one of the terminator types: factor-dependent or intrinsic terminators.

Methods	Year	Factor Dep. (e.g. Rho)	Method	Software Avail.	# of Exp. Terms.	# of species
2.1 - TermNN [14]	2022	Intrinsic	DL	Yes	1175	2
2.2 - ITT pred [16]	2021	Intrinsic	Statistical	No	137	1
2.3 - iterb-PPse [13]	2020	Both	ML	No	928	2
2.4 - iTerm-PseKNC [18]	2019	Both	ML	Yes	852	1
2.5 - RhoTermPredict [20]	2019	Rho-dep.	DP	Yes	1298	3
2.6 - OPLS-DA [21]	2018	Rho-dep.	ML	Yes	104	2
2.7 - PASIFIC [22]	2017	Intrinsic	ML	Yes	330	89
2.8 - RNIE [24]	2011	Intrinsic	DP	Yes	1062	2
2.9 - TransTermHP [25]	2007	Intrinsic	DP	Yes	N/A	N/A

Table 2.1: Software for predicting prokaryotic terminators. The number before the method name is the section in this chapter where that method is discussed. # of Exp. Terms. indicates how many experimentally verified terminators their software is trained/evaluated on. The # of species indicates how many prokaryotes species were involved in their studies. DL means Deep Learning, ML is Machine Learning, and DP is Dynamic Programming.

Chapter 3

Methodology

In this chapter, we first explain the data collection process. Then, we describe different feature generation methods explored to represent the data. Lastly, we describe alternative modelling techniques used to learn from the data.

3.1 Collecting BacTermFinder Dataset

The first step for our study was to collect genomic locations of bacterial terminators, including both intrinsic and factor-dependent terminators. We call the resulting bacterial terminator dataset BacTermData. This section describes in detail the steps taken to accomplish this.

3.1.1 Data Collection Pipeline

Figure 3.1 depicts our data collection pipeline. The following sections will provide more details on each step.

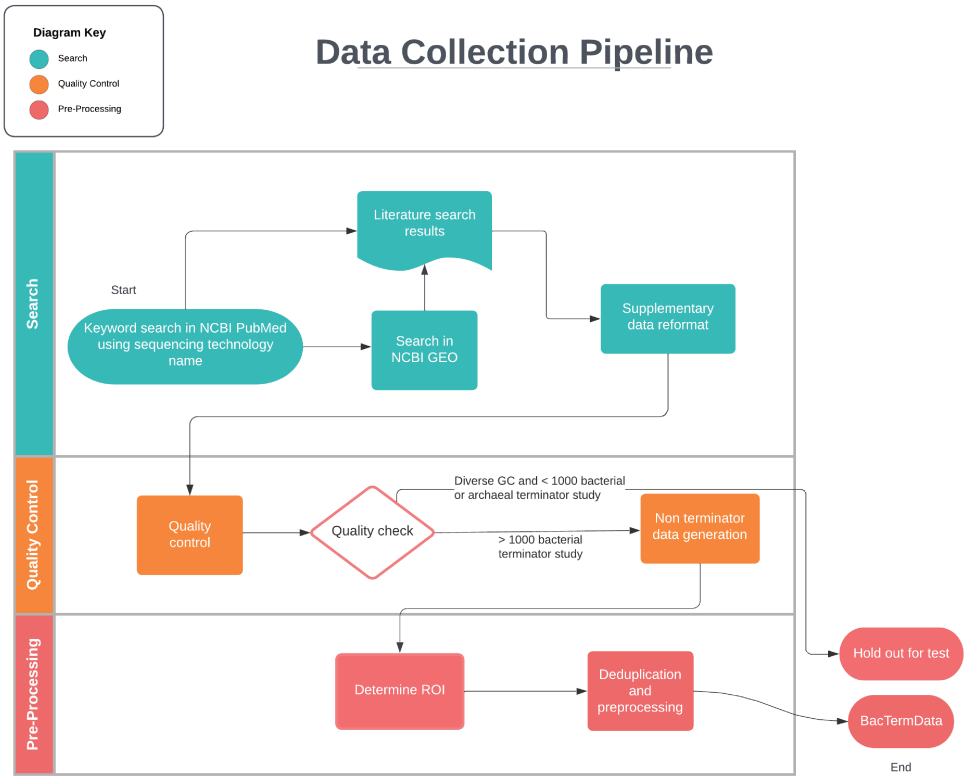

Figure 3.1: Data collection pipeline flow chart. ROI stands for Region Of Interest.

3.1.2 Terminator Data Collection

There are various sequencing technologies to get experimentally verified terminator locations, such as Term-Seq [23], Send-seq [26], SMRT-cappable [27], RendSeq [28], RNATag-seq [29], and dRNA-seq [30]. We searched the NCBI PubMed [31] and GEO database [32] using the names of these sequencing technologies as keywords to find published studies which identified bacterial Transcription Termination Sites (TTSs). Additionally, we collected the TTSs available in the following databases: RegulonDB

[33], DBTBS [34], and BSGatlasDB [35]. For each study, using the IPython Notebook [36] and pandas data manipulation library [37], we stored the genomic locations of identified TTSs (provided in the supplementary material of the corresponding publication) as Browser Extensible Data (BED) files [38]. With bedtools' [39] slopeBed and FastaFromBed commands, we extracted the genomic sequences corresponding to 50 nts flanking the TTSs on either side into a FASTA file. In each operation, strandedness was taken into account. Plasmids were disregarded from the terminator data because plasmids can be transferred between bacterial species or taken from the environment [40]. We also collected archaeal TTSs during this process and kept these for testing the generalizability of the final terminator finder model. The most challenging aspect during data collection was the lack of genome accession IDs in the corresponding publications and of standardization in file formats and terminator classification.

The studies included in our data set are listed in Table 3.1, and the corresponding species with their number of experimentally verified terminators are listed in Table 3.2.

3.1.3 Hold-out For Comparative Assessment

The five bacteria shown in Table 3.3 are used as a hold-out for comparative assessments of existing approaches. The held-out data were chosen because they have diverse GC content, and a relatively small number of terminators were identified in the corresponding study.

3.1.4 Non-Terminator Data Generation

To train machine-learning-based models, instances of non-terminators of the same length as the terminator-containing sequences are needed. We used bedtools shuffle to randomly generate genomic coordinates different from the TTSs to obtain these negative examples. We allowed a maximum sequence overlap between positive and negative sequences of 20 nts. A ratio of 1-10 positive to negative was used. Chevez-Guardado and Pena-Castillo [41] showed that there should be more negatives for each positive to lower the false-positive rate during genome scan. Furthermore, the terminator detection problem imposes a natural imbalance between terminator and non-terminator sequences, as the estimated number of terminators in a bacterial genome is relatively tiny compared to the number of possible non-terminator sequences of the same length.

3.1.5 Confirming The ROI For Terminator Identification

To confirm that the terminator sequence pattern was within the 100 nts extracted, we used relative nucleotide frequency graphs to visualize whether a distinct pattern was present within this region. The relative nucleotide frequency for a specific position was calculated by dividing the total count of each nucleotide in that position by the number of terminators in that set and then applying a log function. The log2 ratio of the relative nucleotide frequency in the ROI vs random genomic regions for each position and nucleotide was obtained by subtracting from the relative nucleotide frequency in the ROI the relative nucleotide frequency in randomly selected genomic

sequences of the same length. After visually analyzing the log2 ratio across Region of Interest (ROI), we confirmed the presence of a pattern within the 100 nts sequence length and decided to keep the 100 nts as the sequence length. The TTS was located in the middle of the ROI.

3.1.6 Data Quality Control

We used the graphs of the log2 ratio of the relative nucleotide frequencies described in Section 3.1.5 to double-check the data quality for each study. In most of the data, there was a clear pattern around the middle of the window. However, we did not observe a pattern for a few studies (not listed in Table 3.2), possibly due to using more permissive inclusion criteria and thus, the signal-to-noise ratio is lower than in other studies. After double-checking that this was not caused by a mistake in our pipeline, we discarded the data from these studies. Figures 3.2 and 3.3, respectively, show examples of low-quality and high-quality data.

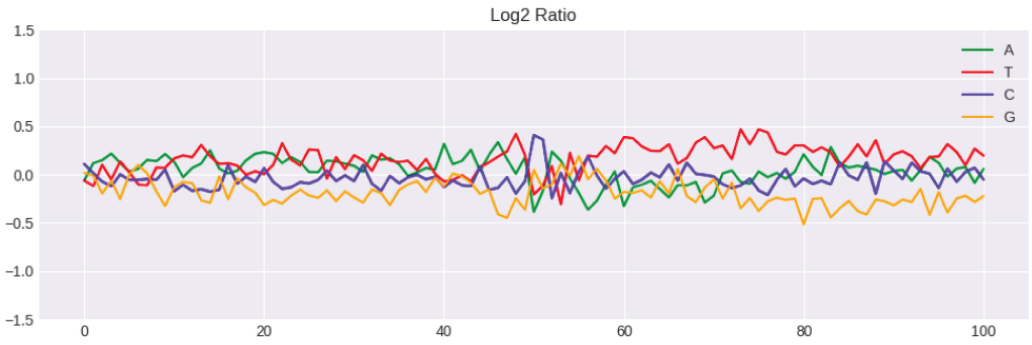

Figure 3.2: Log2 ratio of the relative nucleotide frequency per position within the ROI for an *E. coli* terminator data **without** a distinctive observable pattern

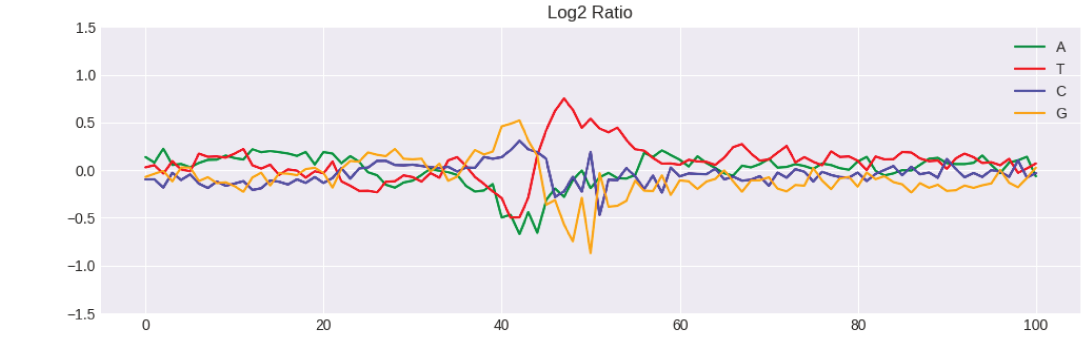

Figure 3.3: Log2 ratio of the relative nucleotide frequency per position within the ROI for an *E. coli* terminator data **with** a distinctive observable pattern

3.1.7 Deduplication And Preprocessing

We needed to remove duplicated sequences from our data as we collected multiple data sets for the same bacterial species (see Table 3.1). To achieve this, we merged ROI genomic coordinates if they had at least a 60% overlap. After merging, we took the middle of the overlap of these coordinates as the center of the TTS. That is, when the merged sequence was bigger than our 100 bp, we would recenter the data by removing the extra base-pairs from the beginning and the end of the sequence. Additionally, sequences containing ambiguous characters (such as N, Y, etc) were removed.

All data was compiled in a comma-separated file. The Comma Separated Values (CSV) file consists of the Reference Genome accession ID, Location of the Terminator, Strand, Sequence, Study PubMed Central (PMC) id, Specie, and Strain. The label column is a binary classification integer. Zero is for non-terminators, and one is for experimentally verified terminators.

BacTermData consists of different types of bacteria terminators (intrinsic and factor-dependent). Table 3.1 shows that BacTermData includes 22 species with varying GC content from three phyla (Pseudomonadota, Bacillota, Actinomycetota) obtained from 26 studies. The total number of bacterial terminators is 46,386.

Test Data

We selected test data with diverse GC content and factor-dependent terminator ratio (see Table 3.3). We call this dataset BacTermBench.

Two archaeal datasets were also used for testing BacTermFinder and TermNN to compare the generalizability of these software on archaea.

Study name	PMC ID or DOI	GEO or SRA or ENA project #	Sequencing Tech.
Dar and colleagues [23]	PMC5756622	PRJEB12568	Term-Seq
Dar and colleagues [42]	10.1038/nmicrobiol.2016.143	PRJEB12568	Term-Seq
Lee and colleagues [43]	PMC8780764	GSE118597	Term-Seq
Lee er. al. [44]	PMC6742748	PRJEB31507	Term-Seq
Hwang and colleagues [45]	PMC8269248	GSE138325	Term-Seq
Ju and colleagues [26]	PMC6814526	GSE117737	Send-Seq
Yan and colleagues [27]	PMC6131387	GSE117273	SMRT-Cappable-seq
Adams. and colleagues [46]	PMC7815308	PRJNA640168	Term-Seq
Choe and colleagues [47]	PMC9023263	PRJEB36932	Term-Seq
Takada and colleagues [48]	PMC9226507	GSE67058	Term-Seq
Mandel and colleagues [49]	PMC8060035	GSE154522	Term-Seq
Johnson and colleagues [50]	PMC7483943	GSE53767 GSE95211	Rend-seq
		GSE108295	
Hwang and colleagues [51]	PMC8914203	PRJEB40918	Term-Seq
Vera and colleagues [52]	PMC7566282	GSE139939	Term-Seq
Lee and colleagues [53]	PMC7738537	PRJEB40918 PRJEB31507	Term-Seq
		PRJEB36379 SRX6937123	
		SRX6937124 PRJEB36379	
Thomason and colleagues [54]	PMC6786874	PRJEB31965	Term-Seq
Mediati and colleagues [55]	PMC9217812	GSE158830	Term-Seq
Warrier and colleagues [56]	PMC6296669	SRP136114	Term-Seq
Lalanne and colleagues [57]	PMC5978003	GSE95211	Rend-seq
Fuchs and colleagues [58]	PMC8237595	GSE155167	RNAtag-Seq
Bastet and colleagues [59]	PMC8745188	Available upon Request	RNA-Seq
Slager and colleagues [60]	PMC6212727	SRP063763	SMRT-Cappable-seq
Dar and colleagues [61]	PMC6061677	GSE109766	Term-Seq
Al kadi and colleagues [62]	PMC8577284	prjna775855	Direct RNA-seq

Table 3.1: PubMed Central (PMC) identifiers, data accession numbers and sequencing technology per study used during data collection.

Study	Specie - Strain - Substrain	Genome Accession #	# of terminators	GC content %
Dar and colleagues [23] and [42]	*Bacillus subtilis - 168*	AL009126.3	1445	42.90%
Takada and colleagues [48], Mandel and colleagues [49] Lalanne and colleagues [57], Geissler and colleagues [35] Johnson and colleagues [50], de Hoon and colleagues [63]	*Bacillus subtilis - 168*	NC_000964.3	4863	42.90%
Lalanne and colleagues [57]	*Caulobacter crescentus - NA1000*	CP001340.1	341	66.22%
Fuchs and colleagues [58]	*Clostridioides difficile - 630*	CP010905.2	1620	28.63 %
Forquet and colleagues [64]	*Dickeya dadantii - 3937*	NC_014500.1	1782	55.50 %
Dar and colleagues [61]	*Escherichia coli - K-12 - BW25113*	CP009273.1	631	50.06 %
Lalanne and colleagues [57], Johnson and colleagues [50] Ju and colleagues [26], Santos-Zavaleta and colleagues [65] Yan and colleagues [27], Adams. and colleagues [46] Choe and colleagues [47]	*Escherichia coli - K-12 - MG1655*	NC_000913.3	4083	50.07 %
Thomason and colleagues [54]	*Pseudomonas aeruginosa - PAO1*	NC_002516.2	805	65.61 %
Mediati and colleagues [55]	*Staphylococcus aureus - JKD6009*	LR027876.1	970	32.41 %
Bastet and colleagues [59]	*Staphylococcus aureus - NCTC 8325*	NC_007795.1	320	32.40 %
Slager and colleagues [60]	*Streptococcus pneumoniae - D39V*	CP027540.1	686	39.14 %
Warrier and colleagues [56]	*Streptococcus pneumoniae - TIGR4*	NC_003028.3	1783	39.13 %
Lee and colleagues [43] and [53]	*Streptomyces avermitilis - MA-4680*	BA000030.4	2004	69.72 %
Hwang and colleagues [45] and Lee and colleagues [53]	*Streptomyces clavuligerus - ATCC27064*	CP027858.1	1581	71.64 %
Lee and colleagues [53]	*Streptomyces coelicolor - M145*	NC_003888.3	1308	71.10 %
Lee and colleagues [53] and Hwang and colleagues [51]	*Streptomyces griseus - NBRC13350*	NC_010572.1	2722	71.21 %
Lee and colleagues [44] and [53]	*Streptomyces lividans - TK24*	CP009124.1	1735	71.22 %
Lee and colleagues [53]	*Streptomyces tsukubaensis - NBRC108819*	CP020700.1	1283	70.85 %
Lalanne and colleagues [57]	*Vibrio natriegens - ATCC 14048*	CP009977.1	905	44.66 %
Lalanne and colleagues [57]	*Vibrio natriegens - ATCC 14048*	CP009978.1	257	44.10 %
Al kadi and colleagues [62]	*Vibrio parahaemolyticus - O3:K6 - RIMD 2210633*	NC_004603.1	1849	44.74 %
Vera and colleagues [52]	*Zymomonas mobilis - ZM4 = ATCC 31821*	CP023715.1	2040	45.67 %
D'Halluin and colleagues [66]	*Mycobacterium tuberculosis - H37Rv*	AL123456	2202	64.69 %

Table 3.2: Number of experimentally verified terminators per species. The number of terminators is the deduplicated number of Terminators. See Section 3.1.7 for more details.

Study	Specie - Strain	Genome Accession #	# of terminators	GC %	Seq. Tech.
Bacteria					
Rosinski-Chupin and colleagues [67]	*Streptococcus agalactiae - NEM316*	NC_004368.1	655	35.12 %	dRNA-Seq
Lee and colleagues [53]	*Streptomyces venezuelae - ATCC15439*	CP059991.1	870	70.73 %	Term-Seq
Cho and colleagues [68]	*Synechocystis - PCC 6803*	NC_000911.1	553	47.04 %	Term-Seq
Jeong and colleagues [69]	*Synechocystis PCC 7338*	CP054306.1	346	47.14 %	Term-Seq
Halluin and colleagues [70]	*Mycobacterium tuberculosis H37Rv*	AL123456.3	2202	64.69 %	Term-Seq
Archaea					
Berkemer and colleagues [71]	*Haloferax volacno - DS2*	NC_013967	1227	65.69 %	Term-Seq
Li and colleagues [72]	*Methanococus maripaludis - S2*	NC_005791	2354	32.63 %	Term-Seq

Table 3.3: Selected bacteria and archaea for comparative assessment

3.2 Feature Generation and Engineering

There are several libraries or software (e.g., MathFeature [73], iLearnPlus [74], RepDNA [75]) to generate features from DNA sequences. We decided to use ILearnPlus [74] because it allows us to extract and analyze various sequence-based features. This section describes how we generate and select the features used in our final model. The ILearnPlus software was slightly modified to better generate features for large amounts of data.

3.2.1 How To Represent Sequences For Machine-learning

The BacTermData consists of DNA sequences that are made up of ATCG characters. However, machine-learning methods expect to receive a numerical representation of the sequences as input. There are many approaches to numerically representing sequences, such as one-hot encoding, k-mer frequencies, etc. As it is not feasible

to determine a priori which representation would generate the "best-performing" machine-learning model, one needs to try out several distinct representations. Here, the best-performing model refers to a model that maximizes a specific performance metric, such as the f1-score or area under the precision-recall curve.

3.2.2 ILearnPlus

ILearnPlus is a comprehensive software for sequence analysis. It can generate features, use unsupervised and supervised machine learning approaches and is user-friendly. ILearnPlus feature generation capabilities were utilized to create features from BacTermData. With ILearnPlus, we generated 6208 features. Some of these features might not be informative to detect bacterial terminators and thus reduce the signal-to-noise ratio in the data. We applied feature engineering and feature selection methods to identify the informative features.

3.2.3 Feature Engineering And Selection

To measure the importance of the features, we used two methods together, SHapley Additive exPlanations (SHAP) [76] and Gini measure of Light Gradient Boosting Machine (LightGBM) [77]. We used an iterative algorithm to drop features if both feature importance methods agree on dropping them (i.e. for both feature importances, the features should be in the bottom 20% after sorting the features based on their importance). Then, we trained a new model with a smaller feature set to recalculate the feature importance values and remove features in the bottom 20%.

This process was repeated until the average of the area under the precision-recall curve (Area Under the Precision-Recall Curve (AUPRC)) obtained during 10-fold cross-validation decreased. We observed a decrease in the AUPRC after reaching 1696 features, down from 6208 features (Figure 3.5). Those 1696 features were then used for training our machine-learning models. Table 3.4 shows the 1696 features encodings. Figure 3.4 shows one hot encoding for positive and negative data, which was found useful by our feature engineering methods.

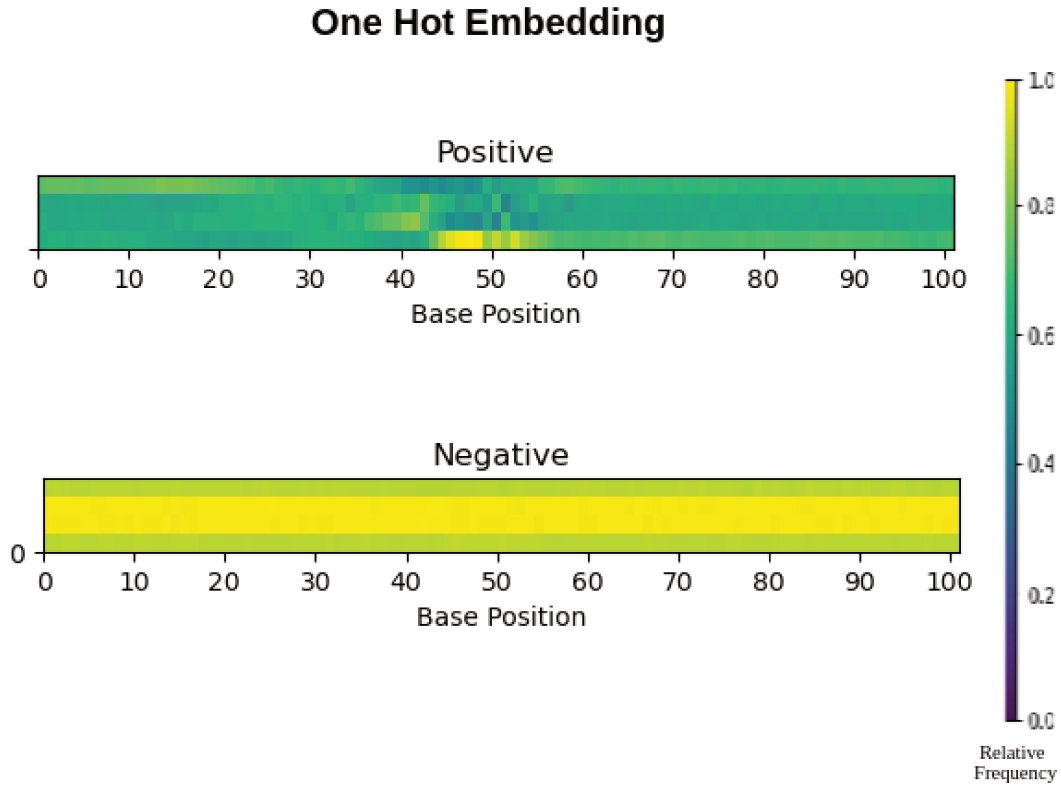

Figure 3.4: One-Hot embedding with Terminator class above and generated non-Terminator class below.

SHAP

SHAP (SHapley Additive exPlanations) [76] is a method used to explain the output of any machine learning model. Shapley values are an assessment method used in cooperation theory; each player is given a value equal to the amount they contribute individually toward the overall goal. In machine learning, SHAP assigns a value to each feature that corresponds to its importance in making predictions. By observing the behaviour of a model, we can learn what features drive its output and how important each feature is in making a specific prediction. In addition to offering a more intuitive way of comparing different models, SHAP can also measure the relative importance of features across different models. Unlike traditional feature importance methods (such as permutation and gain), SHAP is accurate at determining local features [76]. We used the SHAPTreeExplainer function to assess how each feature is helping to detect terminators. The function would result in a real number of positive and negative effects for each class. We used the absolute of those values for each MCCV's test set.

LightGBM Feature Importance

LightGBM [77] is an efficient and scalable gradient-boosting framework that uses tree-based learning algorithms. LightGBM can calculate the feature importance by using gain (Gini index) or split. Gain calculates how much information is gained in each when splitting. In the split, the frequency of the features is counted and reported as important. We used the split to calculate the importance. The split method presumes

that the more frequent feature is the more important one.

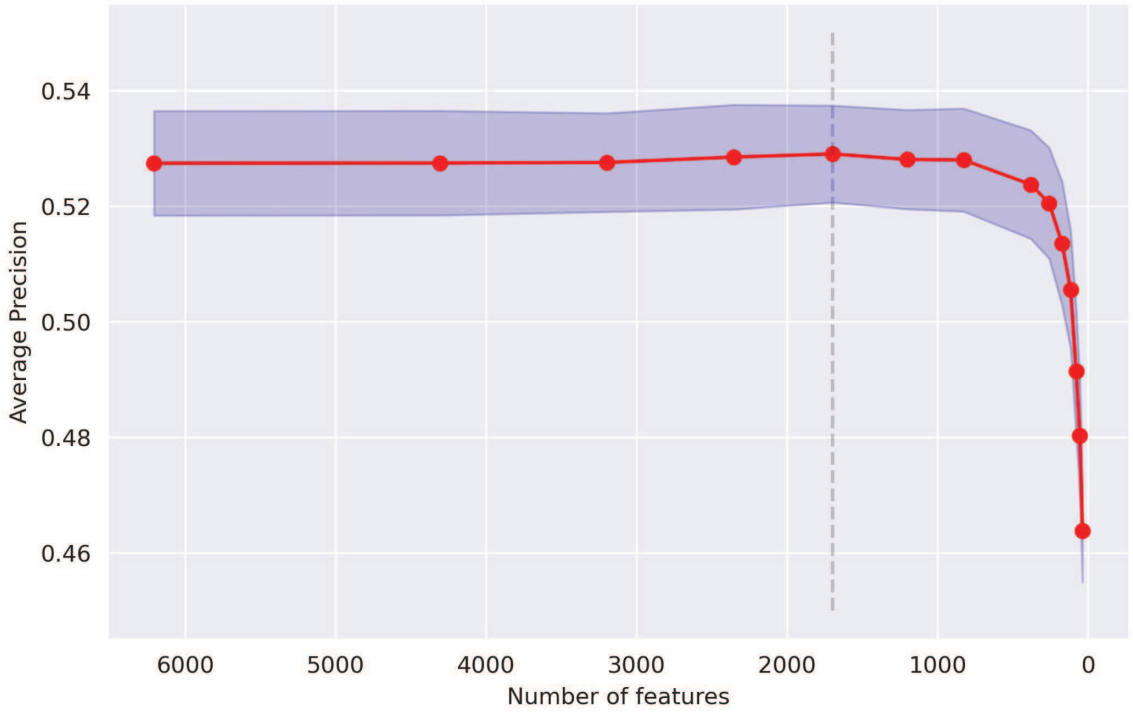

Figure 3.5: Average precision (an estimate of AUPRC) as a function of the number of features included in the training set. Each dot indicates an iteration of the algorithm used to remove unimportant features. The shaded area is the standard deviation of 10 folds.

Six-Feature-set and Full-Feature-set

Generating the 1696 features with ILearnPlus is computationally intensive (roughly, it processes 30 sequences/second in a High-Performance Computing Cluster (High Performance Computing (HPC) cluster). Intending to reduce the computational requirements, we selected the most important feature sets even after filtering out the

Feature set	Descriptor group	# of important features	Description
Geary	Autocorrelation	255	The distribution of amino acid properties throughout the sequence [78].
NMBroto	Autocorrelation	255	The distribution of amino acid properties throughout the sequence [79].
ENAC	Nucleic acid composition	251	Computes the frequency of each nucleic acid type using a sliding sequence window [80].
PseKNC	Pseudo nucleic acid composition	181	K-tuple nucleotide composition [81, 82].
SCPseDNC	Pseudo nucleic acid composition	104	Series correlation pseudo dinucleotide composition information [81, 82].
TACC	Autocorrelation and cross-covariance	100	The relationship between either the same or different physicochemical indices for trinucleotides separated by a lag distance along the sequence [81].
SCPseTNC	Pseudo nucleic acid composition	70	Correlation pseudo trinucleotide composition [81, 82].
PCPseTNC	Pseudo nucleic acid composition	60	Considers parallel correlation pseudo trinucleotide composition information [81, 82].
DACC	Autocorrelation and cross-covariance	57	The relationship between either the same or different physicochemical indices for dinucleotides separated by a lag distance along the sequence [81, 83, 84].
Z_curve_144bit	Nucleic acid composition	56	Frequencies of phase-specific tri-nucleotides [85].
Mismatch	Nucleic acid composition	43	The occurrence of kmers, allowing at most m mismatches [86].
PS2	Residue composition	41	Encoded 16 pairs of adjacent pairwise nucleotides (dinucleotides) [86, 87].
CKSNAP	Nucleic acid composition	39	K-spaced nucleic acid pairs [80].
NCP	Nucleic acid composition	35	Nucleotide chemical property [88].
ANF	Nucleic acid composition	28	Accumulated nucleotide frequency [88].
MMI	Mutual information	26	Multivariate mutual information [89].
binary	Residue composition	22	Each amino acid is represented by a 4-dimensional binary vector [90, 91].
RCKmer	Nucleic acid composition	19	Reverse complement kmer [92, 93].
Z_curve_48bit	Nucleic acid composition	18	Frequencies of phase independent tri-nucleotides [85].
PCPseDNC	Pseudo nucleic acid composition	17	Parallel correlation pseudo dinucleotide composition [75, 82].
Z_curve_9bit	Nucleic acid composition	9	Phase-specific mononucleotides [85].
LPDF	Nucleic acid composition	8	Local position-specific dinucleotide frequency [94].

Table 3.4: Table of selected important features based on SHAP and Feature Importance of LightGBM

unimportant features. A feature set contains the features created by one feature generation method. For example, the PS2 method generates 16 features per nucleotide, and all 64 generated features comprise the PS2 feature set. Figure 3.6 shows the PS2 embedding and the nucleotide frequency of all terminators side by side. Initially, we were computing 28 feature sets, and after feature selection, 22 feature sets remained. We selected the six feature sets that comprised 60% of the 1696 features and called it the Six-Feature-set.

3.3 Machine Learning Modelling

This section describes how we trained ML methods, what methods were used, and how we did the validation and comparative assessments with existing approaches.

3.3.1 Training Methods

We used Stratified Monte Carlo cross-validation (SMCCV) [95] to select the optimal hyperparameters of the models. Monte Carlo cross-validation is a method that will randomly determine the training and validation set in different iterations. The positive-to-negative data ratio is maintained in each iteration as it is stratified. We used SMCCV with 100 folds and 10 iterations to find the optimal hyper-parameters for the models. With 100 folds, 99 % of the data would be for training and the remaining for testing. With 10 iterations, different test sets were chosen each time, resulting in a better representation of the whole data for the test set.

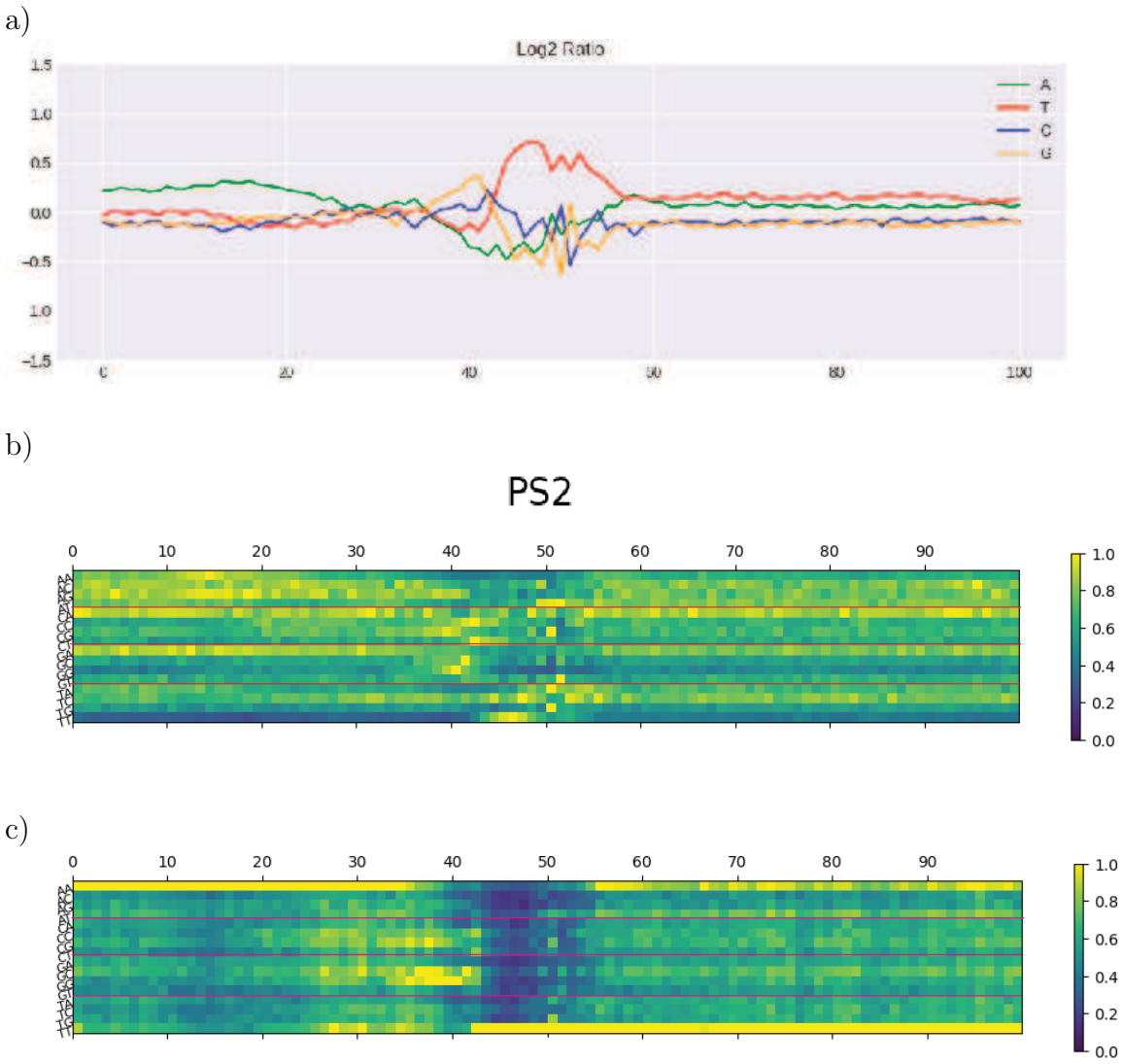

Figure 3.6: Nucleotide frequency and PS2 embedding side by side. A) shows nucleotide frequency of all of BacTermData. B) shows the PS2 embedding normalized row-wise, and C) shows the same, but normalized column-wise, for visualization purposes.

3.3.2 Machine-learning Approaches Used

Deep learning models were utilized as they have been shown to perform well in various domains [96]. We also used Boosting models because of their ability to handle tabular data better than other techniques [97].

Deep Learning Models

Deep Learning (DL) models are Neural Networks with many layers. As a rule of thumb, ML practitioners call a Neural Network with more than three layers a Fully Connected Neural Network (FCNN). Different Neural Network architectures can be leveraged to classify sequences, such as Convolutional Neural Networks (CNN) [96], Recurrent Neural Networks (RNNs) [96], and Transformers [98].

FCNN: A Fully Connected Neural Network FCNN, often called a Deep Learning model, is a class of artificial neural networks designed to model and analyze complex relationships in data. Unlike traditional neural networks, FCNNs are characterized by their depth, comprising multiple hidden layers between the input and output layers.

CNN: A Convolutional Neural Network (CNN) is a deep learning architecture designed primarily for processing structured grid-like data, such as images. CNNs use sliding learnable kernels in their core to find patterns inside the data and often utilize some dense layer to classify the results. In 1D convolution, a 1D data vector is used as input to the model to process the numerical encoding of a sequence.

We started by trying PromotechCNN [99] architecture out of the box. PromotechCNN [99] architecture is designed to tackle separated patterns with its dilated

CNN layers. Then, we added dense layers after the dilated CNNs to improve classification. We also tried a simple CNN such as the one used in TermNN and scaled it up by adding more layers and filters. We varied the sequence encodings, kernel sizes, pooling layers, regularization strategies, and ways to combine CNNs to find the architecture that maximized AUPRC through trial and error. The list below describes the various architectures that were tried.

- CNN
  - **Single**: A series of CNNs layers followed by multiple layers of FCNN, using a single feature set, either One Hot, ENAC, PS2, PS3, or NCP.
  - **Append**: Different feature sets appended together as input of a series of CNNs, followed by multiple layers of FCNN.
  - **Fusion**: Combining the results of different feature set's CNNs during the training.
    - **Single CNN**: Four different embeddings (One Hot, ENAC, PS2, NCP) were fused to a 6-Feature-set, one at a time, resulting in 4 different models: One CNN per Feature set connected to layers of FCNN.
    - **All CNNs**: Features (One Hot, ENAC, PS2, NCP) as inputs to CNNs then concatenated to fed into FCNN layers.
  - **Ensemble**: The outputs of single CNNs (one per feature set) trained separately were averaged to obtain an aggregated output. One CNN block

out of 4 blocks is described in Figure 3.7.

- FCNN
  - **Cat-1696**: 1696 features as input of layers of FCNN. The code for the architecture is available in Figure 3.8.
  - **6-Features-set**: A multi-layer FCNN is trained with the 6-Feature-set as input.

Boosting models

LightGBM [77] is a gradient-boosting framework that has gained popularity due to its efficiency and high performance. It combines multiple decision trees with the gradient boosting technique and is well-suited for large datasets. LightGBM [77] introduces innovative features like histogram-based learning and leaf-wise growth, making it faster and more memory-efficient than traditional gradient-boosting algorithms [97, 101].

Figure 3.9 shows the range of the hyperparameters considered to optimize LightGBM models. We used Randomized Cross-validation with 50 iterations to find the best hyper-parameters. We used the Python LightGBM implementation version 3.3.3.

```
x = Conv1D(filters=64, kernel_size=10, activation='PReLU',
    input_shape=shape,)(input)
x = AveragePooling1D(pool_size=(2))(x)
x = Conv1D(filters=64, kernel_size=10, activation='PReLU',)(x) #
    going every ten bases
x = AveragePooling1D(pool_size=(2))(x)
bn_2 = tf.keras.layers.BatchNormalization()(x)
x = Conv1D(filters=64, kernel_size=10, activation='PReLU',)(bn_2) #
    going every ten bases again to have a wide view of the sequence
d_3 = Dropout(0.1)(x)
x = Flatten()(d_3)
h_1 = Dense(500, activation='PReLU')(x)
d_3 = Dropout(0.3)(h_1)
h_2 = Dense(600, activation='PReLU')(x)
d_3 = Dropout(0.3)(h_2)
output = Dense(600,activation="PReLU")(d_3)
output = Dropout(0.3)(output)
output = Dense(200, activation='PReLU')(output)
output = Dropout(0.4)(output)
output = Dense(200, activation='PReLU')(output)
output = Dropout(0.4)(output)
output = Dense(1, activation='sigmoid')(output)
```

Figure 3.7: The code describing a single CNN block out of many CNN blocks followed by FCNNs in Python3 Keras [100]

```
h_3 = Dense(400, activation='relu')(categorical_input)

d_4 = Dropout(0.3)(h_3)

h_4 = Dense(400, activation='relu')(d_4)

d_5 = Dropout(0.3)(h_4)

h_4 = Dense(400, activation='relu')(d_5)

d_5 = Dropout(0.3)(h_4)

h_4 = Dense(400, activation='relu')(d_5)

d_5 = Dropout(0.3)(h_4)

h_4 = Dense(400, activation='relu')(d_5)

d_5 = Dropout(0.3)(h_4)

h_4 = Dense(400, activation='relu')(d_5)

d_5 = Dropout(0.3)(h_4)

h_4 = Dense(400, activation='relu')(d_5)

d_5 = Dropout(0.3)(h_4)

output = Dense(200, activation='PReLU')(d_5)

output = Dropout(0.4)(output)

output = Dense(200, activation='PReLU')(output)

output = Dropout(0.4)(output)

output = Dense(1, activation='sigmoid')(output)
```

Figure 3.8: The code describing single FCNNs in Python3 Keras [100]

```
'n_estimators': [1000, 2000, 3000],
'max_depth': list(set([randint(3, 12) for _ in range(5)])),
'learning_rate': [0.1, 0.01, 0.001],
'num_leaves': [2**i for i in range(4, 8)],
'boosting' : ['gbdt,' 'goss'],
'subsample': [0.5, 0.7, 1],
'min_child_weight': [1, 3, 5],
'feature_fraction': [0.7, 0.8, 0.9, 1],
'n_jobs': [-1],
'random_state': [42],
```

Figure 3.9: Hyperparamter ranges for the LightGBM model.

3.3.3 Performance Data Analysis

We used SMCCV to assess the models' performance. Then, different metrics were extracted per species, studies, strands, and GC content. We calculated score thresholds for classification that maximize the F0.5, F1, and F2 scores. The performance metrics used are described below.

Average Precision

$$\text{AP} = \sum_n (R_n - R_{n-1}) P_n$$

where R_n and P_n are the precision and recall at the n^{th} threshold. Recall measures the proportion of actual positive instances correctly identified by a model, while

precision measures the proportion of predicted positive instances that are actually true positives.

F-score

$$FScore = (1 + \beta^2) \frac{2 * Precision * Recall}{\beta^2 * Precision + Recall}$$

where β can be 0.5, 1, or 2 for the corresponding F Score.

Recall

Recall is used for comparative assessment (Section 3.3.4) of selected existing methods. The recall formula is as follows:

$$Recall = \frac{TP}{TP+FN}$$

Where TP is True Positive, and FN is False Negative

3.3.4 Comparative Assessment

We compared BacTermFinder's performance with state-of-the-art approaches for bacterial terminator identification. We selected these approaches based on software availability, number of citations, publication year and ability to find different kinds of terminators. Table 3.5 lists the approaches used for the comparative assessment.

Performance Metrics In Comparative Assessment

As there is no set of experimentally verified non-terminators across whole bacterial genomes, we assessed each existing method's performance with the average recall over different overlapping thresholds between the actual and predicted terminator

Methods	Year	Factor Dep. (eg. Rho)	Method	Software Avail.	# of Exp. Terms.	# of species
2.1 - TermNN [14]	2022	Intrinsic	DL	Yes	1175	2
2.4 - iTerm-PseKNC [18]	2019	Both	ML	Yes	852	2
2.5 - RhoTermPredict [20]	2019	Rho-dep.	DP	Yes	1298	3
2.9 - TransTermHP [25]	2007	Intrinsic	DP	Yes	N/A	N/A

Table 3.5: Methods used for the comparative assessment. "# of Exp. Terms." is the number of experimental terminators used to train the algorithm, if applicable, and the "# of species" is the number of species the algorithm was trained on.

locations. The method that generates fewer predictions with higher recall would have fewer false positives among its predictions. We averaged the recall over ten thresholds of percentage sequence overlap between the predicted and experimentally verified terminators in the hold-out data. The overlap thresholds ranged from 10 percent to 100 percent overlap.

3.4 Summary

This chapter covered data collection, feature generation, and machine learning modelling. At first, the BacTermFinder dataset is explained, including genomic locations of bacterial terminators from various sources, and a detailed data collection pipeline is outlined. Then, a hold-out dataset is designated for comparative assessments, and non-terminator data is generated for machine learning training. Data quality control measures involving nucleotide frequency analysis and deduplication were described.

Following data preparation, the chapter goes into feature generation and engineering using ILearnPlus, emphasizing the numerical representation of DNA sequences for machine learning. Feature engineering involves selecting informative features and reducing the initial set of 6208 features to 1696 using SHAP and LightGBM feature importance measures. The chosen features are then utilized for training machine learning models.

The machine learning section introduces Stratified Monte Carlo cross-validation for hyperparameter optimization and explores deep learning models such as Fully Connected Neural Networks (FCNN) and Convolutional Neural Networks (CNN). LightGBM, as a statistical machine learning method, was also employed for its efficiency in handling tabular data. The evaluation metrics encompass Average Precision, F0.5, F1, and F2 scores, while the comparative assessment involves benchmarking BacTermFinder against existing approaches, considering recall over various sequence overlap thresholds between the experimentally determined terminators and the predicted terminators.

Chapter 4

Results and Discussion

This section shows the results of our data processing and modelling. Additionally, we provide the results of the comparative assessment of our method (BacTermFinder) and other approaches. Finally, we interpret these results and discuss some limitations of our method.

4.1 Finding The Region Of Interest To Identify Terminators

Based on the nucleotide frequency plot (Figure 4.1) explained in section 3.1.5, we observe that after 20nts downstream of the Transcription Termination Site (TTS), there is no observable pattern. That enabled us to select 100 base-pair long sequences with the TTS in the middle as the regions of interest.

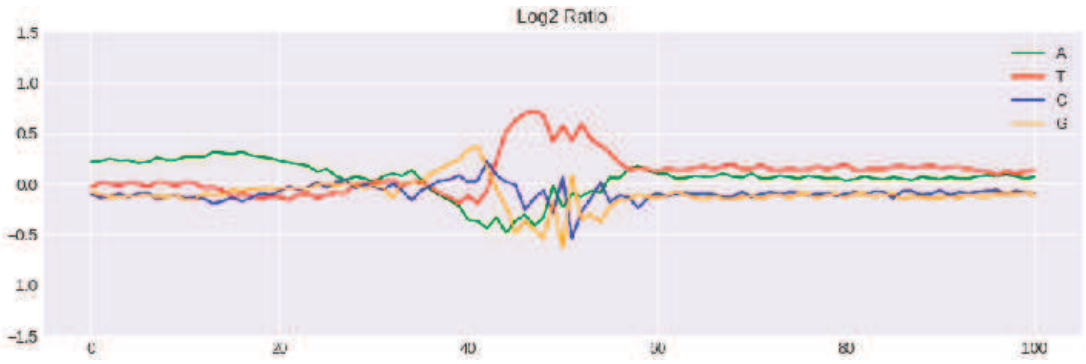

Figure 4.1: Log2 Ratio of the nucleotide frequency aggregated on all terminator sequences in BacTermData

4.2 Model Selection

We comprehensively assessed various Machine Learning (ML) approaches and sequence encodings combinations. This assessment involved training these models using Stratified Monte-Carlo Cross Validation (SMCCV). The outcomes of this evaluation are presented in Table 4.1, and they indicate that an ensemble consisting of different Convolutional Neural Networks (CNNs) achieves an average precision of 0.7080 $\pm$ 0.0248, outperforming individual CNNs and LightGBM classifiers.

While the Light Gradient Boosting Machine (LightGBM) demonstrated performance comparable to that achieved by single CNNs, LGBM requires 1696 features to achieve this performance. Acquiring and processing such a large feature set is resource-intensive and costly. In contrast, some DL models outperformed LGBM models using a single feature set.

In our assessment, we observed the effect of network architectures. For example,

ML Method	Model Name	Features	Average Precision (± STD)
CNN	CNN_Fusion_all	PS2, ENAC, OH, and 6-feature-set	0.5955 ± 0.0232
	CNN_Fusion_single	OH and 6 feature set	0.6553 ± 0.0252
	CNN_Append	PS2, ENAC, OH, and 6 feature set	0.5949 ± 0.0257
	CNN_Single	PS2	0.6738 ± 0.0253
	CNN_Single	PS3	0.6128 ± 0.0213
	CNN_Single	OH	0.6775 ± 0.0214
	CNN_Single	ENAC	0.6544 ± 0.0284
	CNN_Single	NCP	0.6730 ± 0.0243
	CNN_Ensemble	Mean of PS2, ENAC, OH, NCP	**0.7080 ± 0.0248**
FCNN	FCNN_Cat	6 feature set	0.5012 ± 0.0319
LGBM	LGBM_Feature_set	6 feature set	0.5933 ± 0.0269
	LGBM_Full	1696 features	0.6476 ± 0.0217

Table 4.1: The table of SMCCV results on training data.

the models CNN_Fusion_all and CNN_Ensemble use similar feature sets and differ in their architecture. CNN_Fusion_all concatenates the outputs of CNNs on each feature while training, while CNN_Ensemble trains a separate CNN per feature set and then averages their output after training is done. This difference resulted in an approximate 0.11 improvement in average precision (from 0.59 to 0.70). As the CNN_Ensemble model has the highest average precision, we selected this as our final model (referred to as BacTermFinder).

We investigated the performance of the BacTermFinder per bacterium. We aggregated all SMCCV results across all iterations for each bacterium and visualized the results (Figure 4.2). As can be seen from this figure, there is a wide range ([0.49,

	ref_gene	avg_prc	% of val	precision	recall	true_pos_neg	pred_pos_neg	TP	FN	FP	TN	length	f1	f0.5	f2	accuracy
0	CP009977.1	0.970000	2.870000	0.970000	0.920000	[98, 1073]	[93, 1078]	90	8	3	1070	1171	0.940000	0.960000	0.930000	0.990000
1	CP027540.1	0.940000	1.880000	0.930000	0.850000	[66, 699]	[60, 705]	56	10	4	695	765	0.890000	0.920000	0.860000	0.980000
2	CP001340.1	0.930000	0.970000	0.970000	0.880000	[34, 361]	[31, 364]	30	4	1	360	395	0.920000	0.950000	0.900000	0.990000
3	NC_007795.1	0.900000	1.700000	0.930000	0.790000	[71, 623]	[60, 634]	56	15	4	619	694	0.850000	0.900000	0.810000	0.970000
4	CP009978.1	0.900000	0.740000	0.880000	0.830000	[18, 285]	[17, 286]	15	3	2	283	303	0.860000	0.870000	0.840000	0.980000
5	NC_002516.2	0.890000	2.210000	0.930000	0.670000	[81, 822]	[58, 845]	54	27	4	818	903	0.780000	0.860000	0.710000	0.970000
6	AL009126.3	0.880000	4.680000	0.930000	0.820000	[168, 1738]	[147, 1759]	137	31	10	1728	1906	0.870000	0.910000	0.840000	0.980000
7	CP010905.2	0.840000	4.440000	0.880000	0.720000	[167, 1644]	[136, 1675]	120	47	16	1628	1811	0.790000	0.840000	0.750000	0.970000
8	LR027876.1	0.810000	2.620000	0.930000	0.560000	[91, 977]	[55, 1013]	51	40	4	973	1068	0.700000	0.820000	0.610000	0.960000
9	CP009273.1	0.810000	3.030000	0.890000	0.640000	[101, 1135]	[73, 1163]	65	36	8	1127	1236	0.750000	0.830000	0.680000	0.960000
10	NC_010572.1	0.760000	7.150000	0.930000	0.420000	[275, 2640]	[125, 2790]	116	159	9	2631	2915	0.580000	0.750000	0.470000	0.940000
11	NC_003888.3	0.730000	3.600000	0.980000	0.350000	[116, 1351]	[42, 1425]	41	75	1	1350	1467	0.520000	0.720000	0.410000	0.950000
12	NC_000964.3	0.720000	13.070000	0.940000	0.540000	[487, 4842]	[282, 5047]	265	222	17	4825	5329	0.690000	0.820000	0.590000	0.960000
13	NC_004603.1	0.670000	5.900000	0.910000	0.530000	[176, 2230]	[103, 2303]	94	82	9	2221	2406	0.670000	0.800000	0.580000	0.960000
14	NC_003028.3	0.670000	4.990000	0.850000	0.420000	[181, 1853]	[89, 1945]	76	105	13	1840	2034	0.560000	0.710000	0.470000	0.940000
15	BA000030.4	0.630000	5.510000	0.910000	0.350000	[208, 2039]	[80, 2167]	73	135	7	2032	2247	0.510000	0.690000	0.400000	0.940000
16	CP027858.1	0.620000	4.200000	0.870000	0.310000	[156, 1558]	[55, 1659]	48	108	7	1551	1714	0.450000	0.640000	0.350000	0.930000
17	NC_000913.3	0.620000	11.360000	0.910000	0.350000	[449, 4182]	[173, 4458]	157	292	16	4166	4631	0.500000	0.690000	0.400000	0.930000
18	CP009124.1	0.590000	5.350000	0.880000	0.300000	[197, 1986]	[67, 2116]	59	138	8	1978	2183	0.450000	0.630000	0.350000	0.930000
19	CP020700.1	0.550000	3.390000	0.910000	0.280000	[106, 1275]	[33, 1348]	30	76	3	1272	1381	0.430000	0.630000	0.330000	0.940000
20	CP023715.1	0.540000	5.560000	0.810000	0.380000	[190, 2075]	[89, 2176]	72	118	17	2058	2265	0.520000	0.660000	0.420000	0.940000
21	NC_014500.1	0.490000	4.770000	0.850000	0.230000	[175, 1771]	[47, 1899]	40	135	7	1764	1946	0.360000	0.550000	0.270000	0.930000

Figure 4.2: BacTermFinder performance per bacterium. Colouring indicates performance ranking where dark red means the highest average precision, dark blue means the lowest average precision and white means closer to the overall average precision. Table 3.1 has the mapping of genome accession to the corresponding species name.

0.97] and [0.36, 0.94]) in performance per bacterium in terms of average precision and F1 score, respectively. Figure 4.3 shows the Precision-Recall curve and Receiver Operating Characteristic Receiver-Operating-Characteristic curve (ROC) curve of BacTermFinder over all iterations of SMCCV. Clearly, BacTermFinder's performance is well above a random classifier's performance (dashed lines in Figure 4.3).

To look further into the variation in performance across bacterial species, we visualized the average precision per bacterium vs their GC content and coloured it by the phylum to see if there was any pattern. A linear regression line in Figure 4.4 indicates a relationship between GC content and average precision. As the GC content in different genomes increases, the performance decreases slightly. For a 10 %

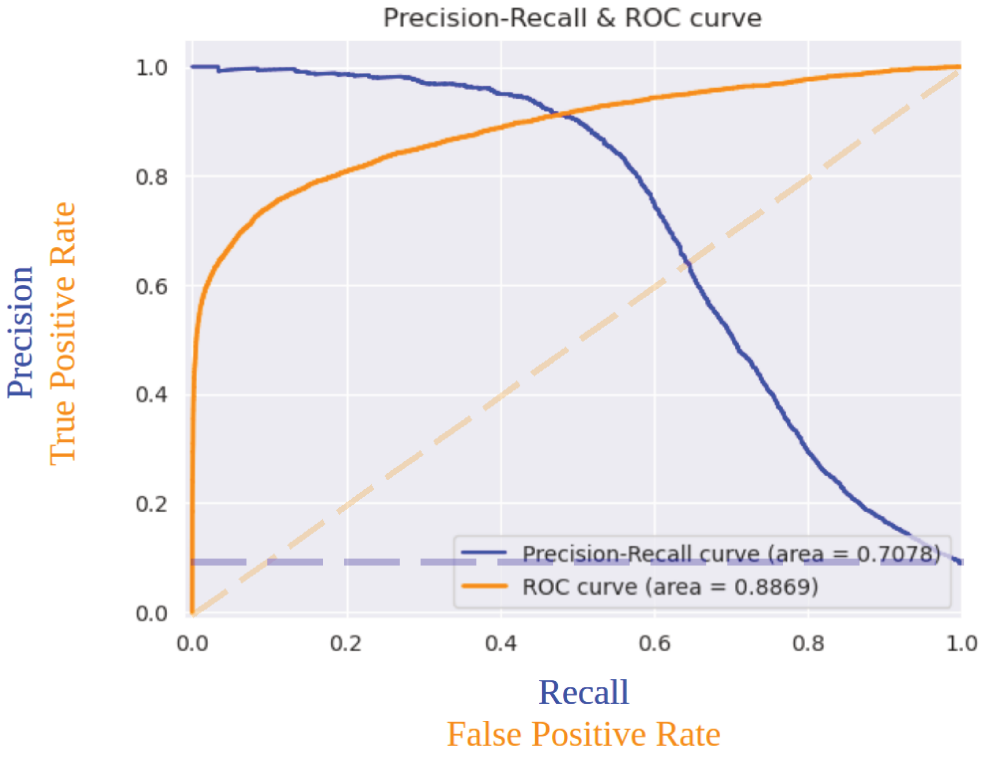

Figure 4.3: Precision-Recall curve and ROC of BacTermFinder on the aggregation of the validation data of SMCCV iterations. Dashed lines show the performance of a random classifier on SMCCV data for each curve. The Orange dashed line is for ROC, and the blue is for the Precision Recall curve.

increase in GC content, the average precision decreases by 0.038. This indicates that BacTermFinder tends to achieve higher average precision in bacteria with lower GC content. Bacteria with high GC content tend to have more factor-dependent terminators [66] and the rut site is likely outside our ROI. Factor-dependent terminators do not always have the hairpin structure of intrinsic terminators [102], and thus, their sequence motif is weaker, which might explain why BacTermFinder's performance is lower on this type of bacteria. In Figure 4.4, we also observe that BacTermFinder's performance is more consistent (with less variation) for Bacilota and Actinomycetota than for Pseudomonadota. Further investigation is needed to understand the reasons for this. Figure 4.5 shows the same plot as 4.4 but is coloured based on individual genome accession numbers.

4.3 Finding The Threshold For Classification

We established our model selection criteria, primarily focusing on threshold-independent metrics, such as the Average Precision score. This strategic choice ensured that our model selection process remained impartial and not inclined toward any specific class within the imbalanced classification. However, a set probability threshold is usually desired to decide whether a sequence is a terminator.

Consequently, we adopted the F1 score to determine a decision threshold to classify sequences into terminators and non-terminators. We examined our Stratified Monte-Carlo Cross Validation SMCCV results to identify an optimal threshold, seeking thresholds that maximized the F1, F0.5, and F2 scores. We opted for an F1 score

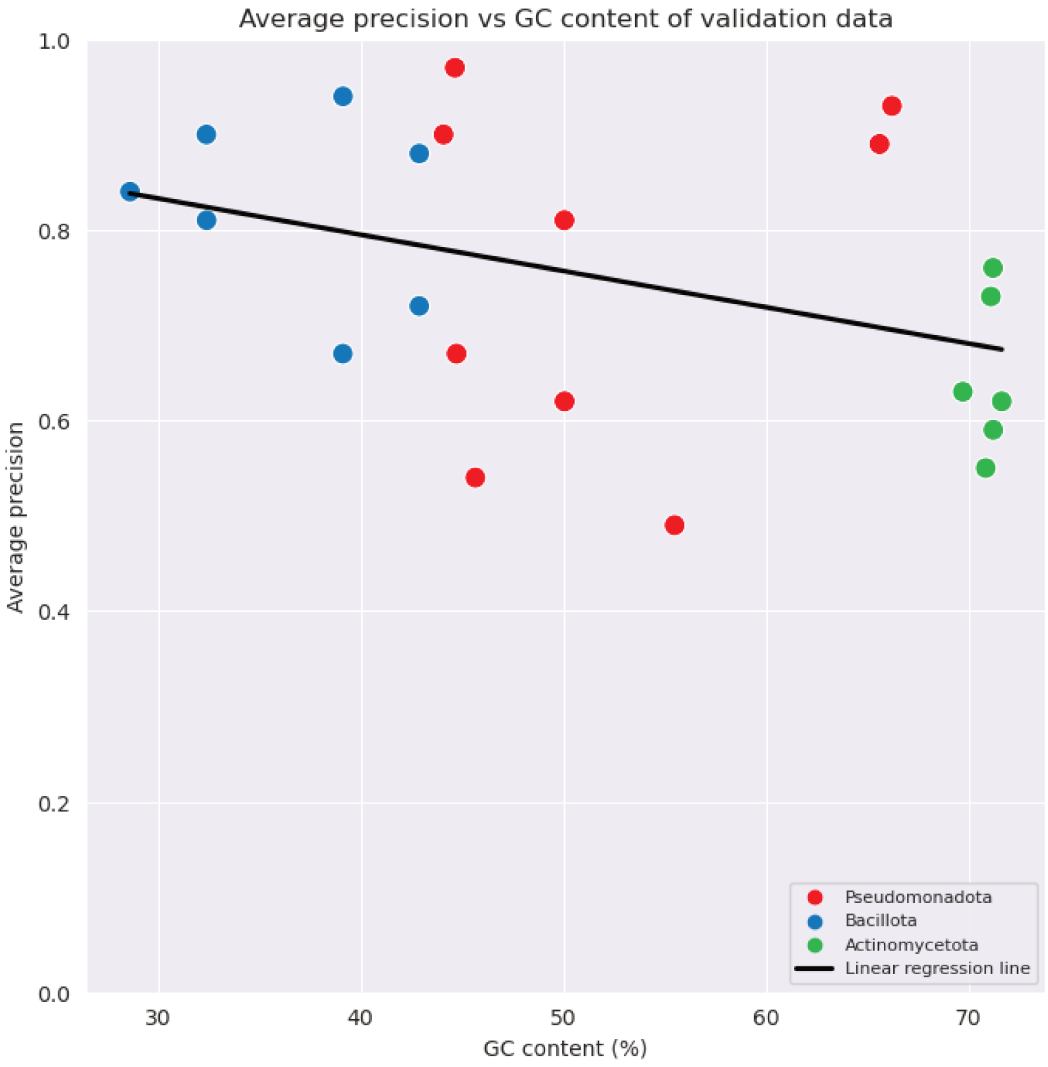

Figure 4.4: Average precision vs GC content for BacTermFinder trained and validated with SMCCV. Coloured based on bacterial phylum

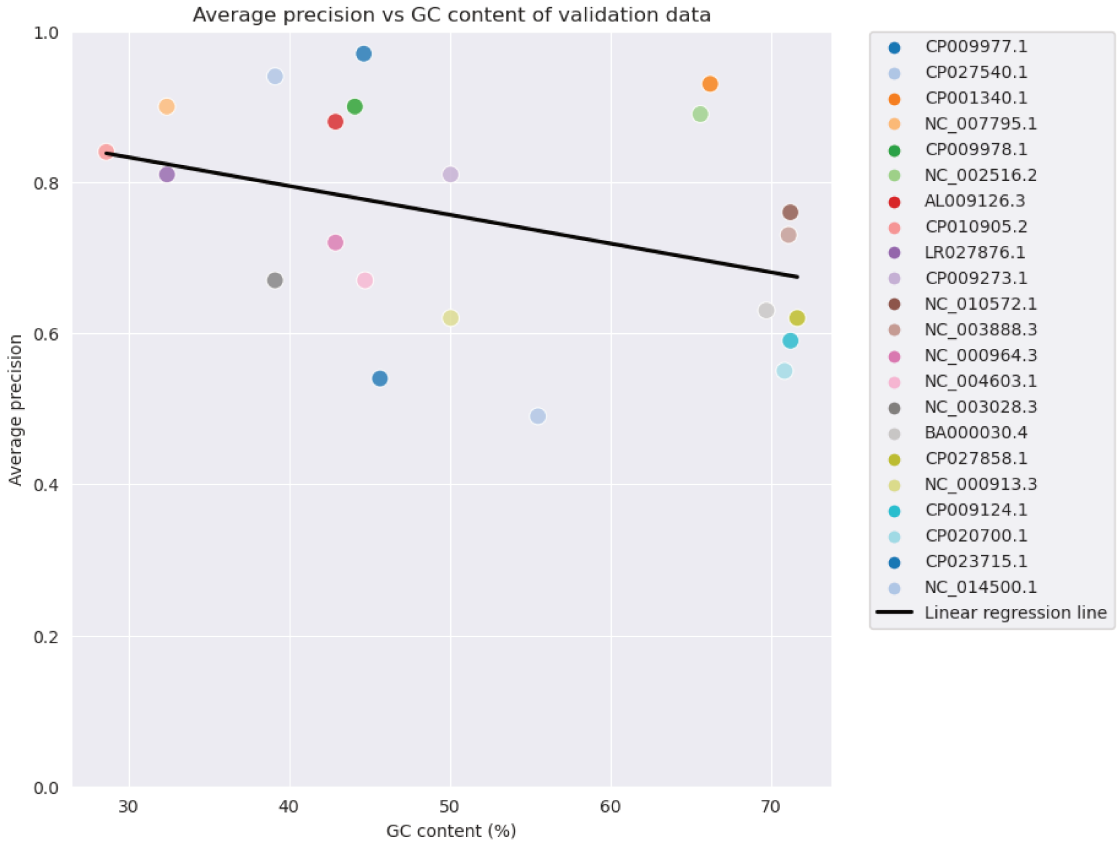

Figure 4.5: Average precision vs GC content for BacTermFinder trained and validated with SMCCV. Coloured based on genome accession number

with a maximum threshold of 0.30 to balance precision and recall.

To assess the biological validity of our prediction with this threshold, we generated a visual representation; a sample of it is shown in Figure 4.6, where our predictions are represented in blue and actual genes are depicted in solid red across the lines of the genome. The horizontal red dashed line represents the 0.30 threshold. An IPython notebook is available on our GitHub repository for those interested in exploring these predictions in greater detail. Our observation revealed terminators with high probabilities located at the end and middle of genes. Terminators in the middle of genes might be due to the existence of small unannotated transcripts. However, further investigation is needed to explain this observation. For instance, one could look at codon usage frequency before terminators in the middle of genes vs similar regions in genes without terminators in the middle.

4.4 Interpreting BacTermFinder Model

Once we have chosen the BacTermFinder final model (i.e., CNN_Ensemble), we tested this model on an independent set of terminators from bacteria not included in the training data (Table 3.3). First, we looked at whether predicted terminators have a nucleotide frequency similar to experimentally verified terminators. As it can be seen from Figure 4.7 for *Streptococcus agalactiae*, the nucleotide frequency of predicted terminators is very similar to that of experimentally verified terminators. The line smoothness of Figure 4.7(b) is due to being a larger number of predicted terminators (12,813) vs experimentally validated terminators (655).

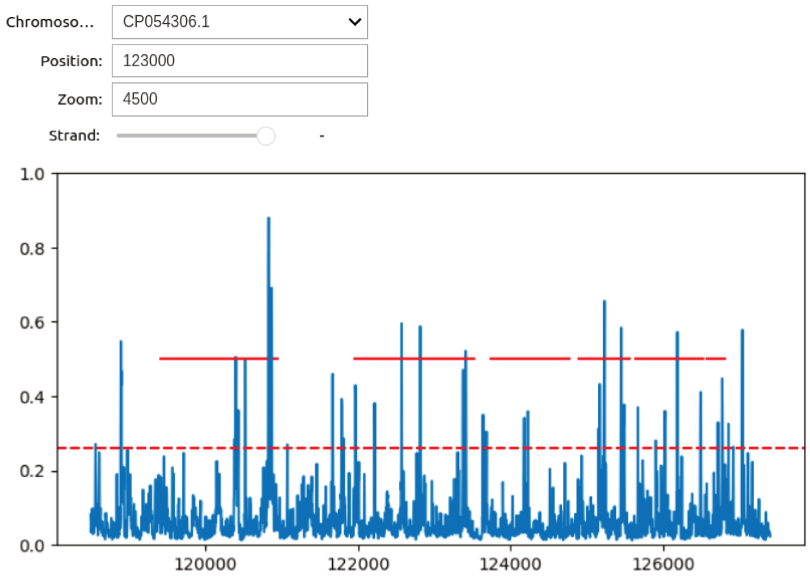

Figure 4.6: A view of the Gene Viewer IPython Notebook. The blue line is the predicted probability of a terminator being present in any given location of the reference genome. Solid red lines are genes, and the red dashed line is the threshold for classification.

The saliency map (Figure 4.7(c)) shows how each nucleotide is responsible for the model's decision. Negative numbers reduce the probability of being a terminator, while positive numbers increase the probability of being a terminator. Figure 4.8 shows the same analysis for the archaea *Methanococus maripaludis*, suggesting that BacTermFinder can detect archaeal terminators.

These results indicate that BacTermFinder prediction has similar sequence motifs to experimentally determined terminators and that BacTermFinder is also suitable for finding archaeal terminators.

4.5 Comparative Assessment

We tested several Bacterial Terminator Finder programs on BacTermBench data to compare BacTermFinder's performance with that of existing methods. These software were executed for the whole genome of each bacterium, and the results were compared with the experimentally determined data of BacTermBench. The intersection of the predicted results and BacTermBench was used to calculate a recall score. We ran bedtools intersect with ten different overlap thresholds between predicted terminators and actual terminators to calculate recall at different overlaps and reported the mean recall in Table 4.2.

We then used the best two programs as per the results shown in Table 4.2 (TermNN and BacTermFinder) and RhoTermPredict to predict terminators of *Mycobacterium tuberculosis*. RhoTermPredict was included in this comparison because *M. tuberculosis* is reported to have a large proportion (up to 54%) of factor-dependent termina-

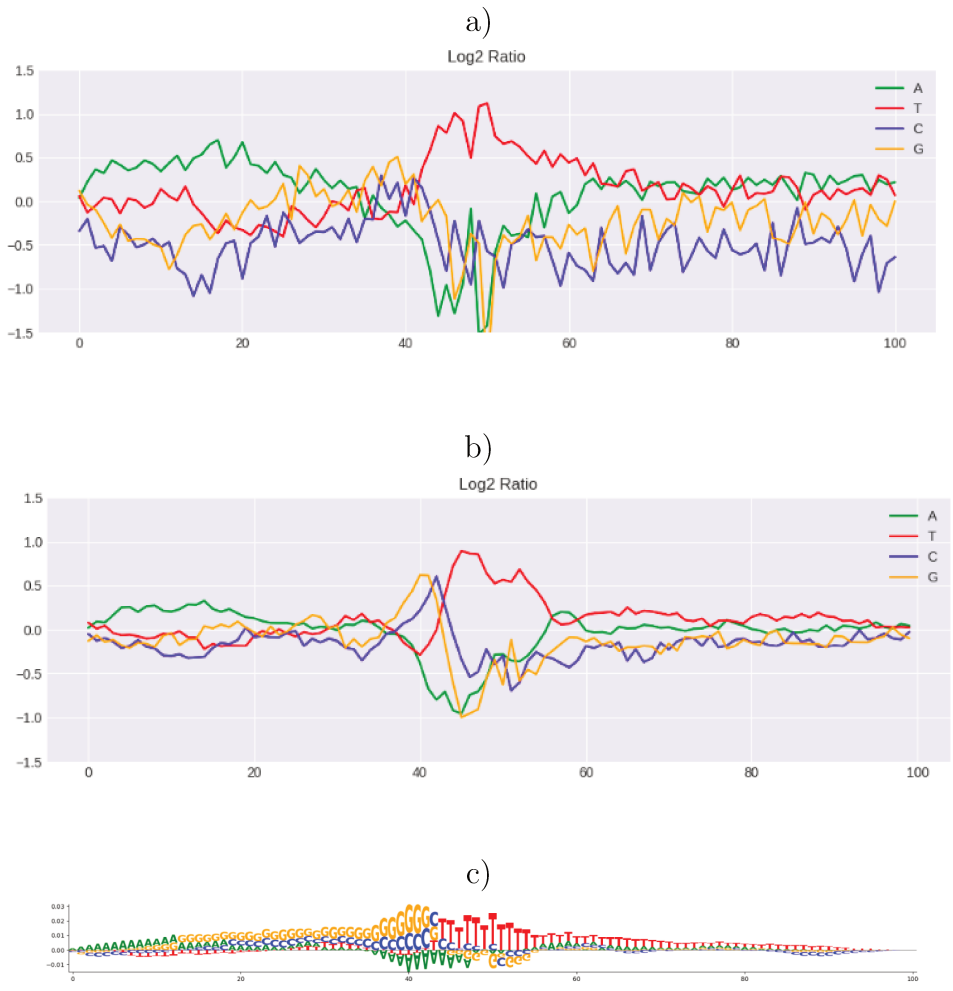

Figure 4.7: *Streptococcus agalactiae* Visualization of **Bacterial** testing data vs model predictions nucleotide frequencies and saliency map to aid in interpreting the model. a) Nucleotide frequency of experimentally determined terminators b) Nucleotide frequency of genome-wide predicted terminators. c) Saliency map.

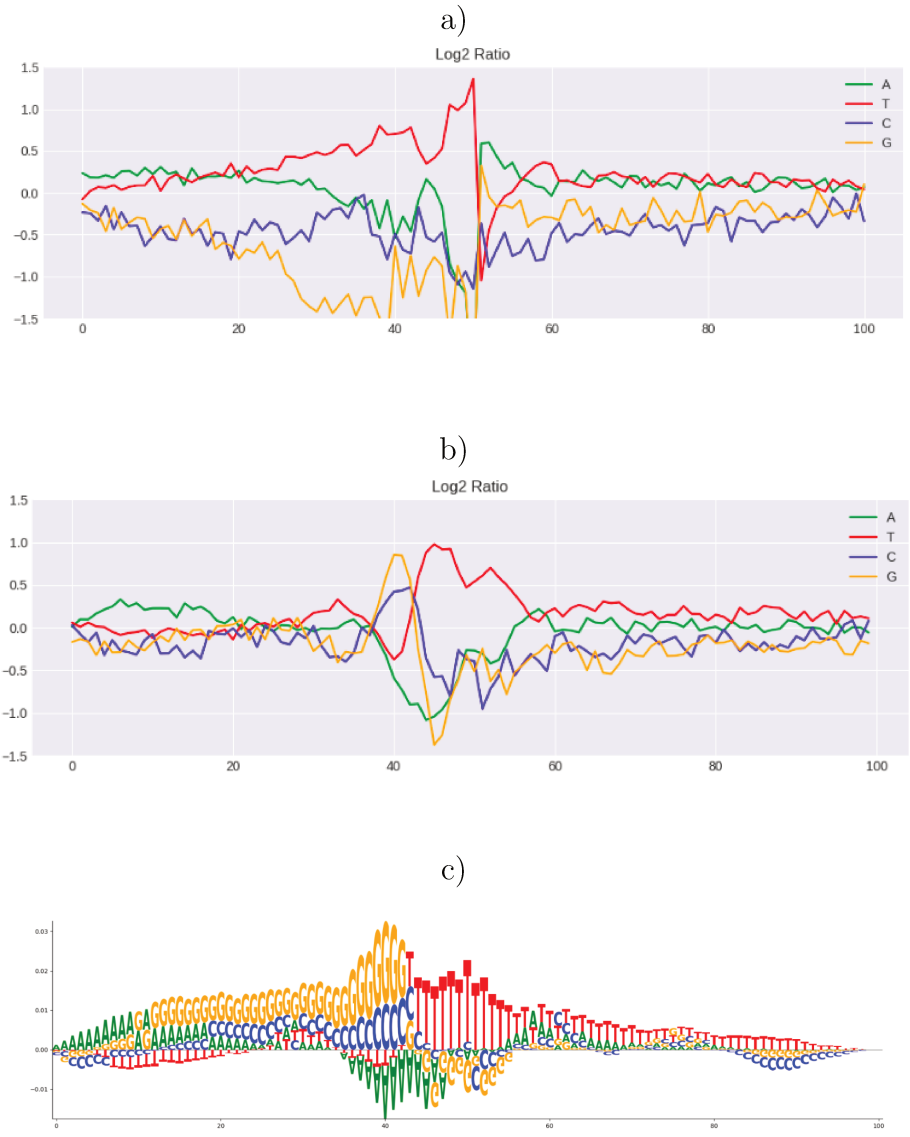

Figure 4.8: *Methanococus maripaludis* Visualization of **Archeal** testing data vs model predictions nucleotide frequencies and saliency map to aid in interpreting the model. a) Nucleotide frequency of experimentally determined terminators b) Nucleotide frequency of genome-wide predicted terminators. c) Saliency map.

tors [66]. As RhoTermPredict was designed to predict factor-dependent terminators, we expected it would work well for this bacterium. Table 4.3 shows the results of this comparison. This result indicates that BacTermFinder's performance in finding factor-dependent terminators is comparable to RhoTermPredict, which was developed specifically to identify factor-dependent terminators. Finally, we used TermNN and BacTermFinder to predict archaea terminators to test the generalizability of these models (Table 4.4).

Bacteria Specie	RhoTermPredict	ITerm-PseKNC	TransTermHP	TermNN	BacTermFinder
Streptococcus agaletea	0.0188 ± 0.0062	0.2402 ± 0.1256	0.5596 ± 0.2656	0.7715 ± 0.3196	**0.8209 ± 0.3002**
NC_004368.1 - GC=35.12 %					
Streptomyces venezuela - (gardneri)	0.2499 ± 0.1263	0.0101 ± 0.0042	0.1709 ± 0.1035	0.3690 ± 0.2227	**0.6029 ± 0.2177**
CP059991.1 - GC=70.73 %					
Synechocytis 6803	0.1210 ± 0.0789	0.1012 ± 0.0544	0.2226 ± 0.1035	0.4817 ± 0.2427	**0.5450 ± 0.1818**
NC_000911.1 - GC=47.04 %					
Synechocytis 7338	0.1040 ± 0.0443	0.1178 ± 0.0523	0.2786 ± 0.1488	0.5136 ± 0.2643	**0.6361 ± 0.2206**
CP054306.1 - GC=47.14 %					

Table 4.2: Average of recall over ten overlap threshold for the existing approaches running with bacterial data.

Bacteria Specie	RhoTermPredict	TermNN	BacTermFinder
Mycobacterium tuberculosis H37Rv	**0.2424 ± 0.164**	0.1396 ± 0.0927	0.2306 ± 0.1191
AL123456.3 - GC=64.69 %			

Table 4.3: Average of recall over ten overlap threshold for the best factor-dependent and intrinsic terminator finder as per Table 4.2 compared with BacTermFinder for *Mycobacterium tuberculosis*.

Archaea Specie	TermNN	BacTermFinder
Haloferax volacno	0.1536 ± 0.1067	**0.3158 ± 0.1969**
NC_013967 - GC=65.69 %		
Methanococus maripaludis	0.4000 ± 0.2235	**0.5712 ± 0.2468**
NC_005791 -GC=32.63%		

Table 4.4: Average recall over ten overlap threshold for the best two approaches trained on bacterial data.

4.6 Comparative Assessment Discussion

Some of the software included in this comparative assessment was designed to identify only one type of terminator, and some of their low recall scores in Table 4.2 can be caused by this. We observed that RhoTermPredict performs better in the higher GC bacteria, which tend to have a larger proportion of factor-dependent terminators [66].

ITerm-PseKNC and BacTermFinder are the only software that predicts both types of terminators. ITerm-PseKNC has low recall across all bacteria on BacTermBench, but its recall is an order of magnitude lower on *Streptomyces venezuelae* with the highest GC content (70.7%). Our performance also decreases in higher GC bacteria. One of our limitations could be that we are not finding factor-dependent terminators effectively. TransTermHP achieves its highest recall in the low GC bacteria because they would have more intrinsic terminators. As shown in Table 4.4, we perform better

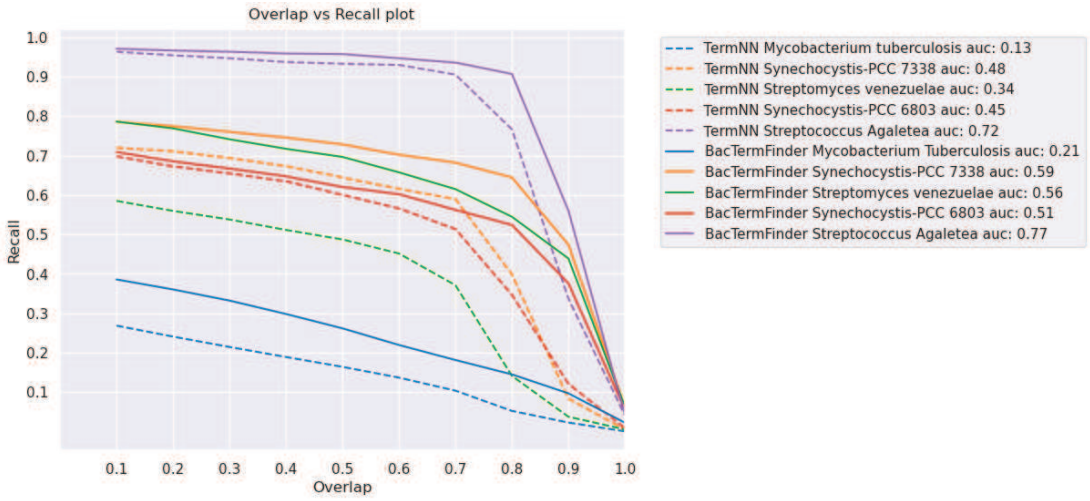

Figure 4.9: Recall as a function of percentage overlap between predicted and actual terminators. All sequences are 100 nts long. The dotted lines are TermNN, and the solid lines are BacTermFinder. Each colour represents a bacterial species in the test dataset. The area under the curves is shown in the legend.

than TermNN in predicting archaeal terminators. This indicates that our software is not only bacteria agnostic, but it could be prokaryote agnostic as well. Including archaeal terminators in the training, data could aid in improving the performance of BacTermFinder in these prokaryotic organisms.

We assessed the precision of our predictions by averaging various overlaps between predicted and experimental terminators. This allowed us to gauge the preciseness of our predictions and determine how closely they aligned with the actual terminator location. To visualize these results, we plotted the recall rate as a function of the overlap percentage of the predicted terminators with the actual terminators (Figure 4.9). We hypothesized that we would observe a declining trend as we moved towards stricter overlap thresholds. In Figure 4.9, we compared our overlap vs recall with that

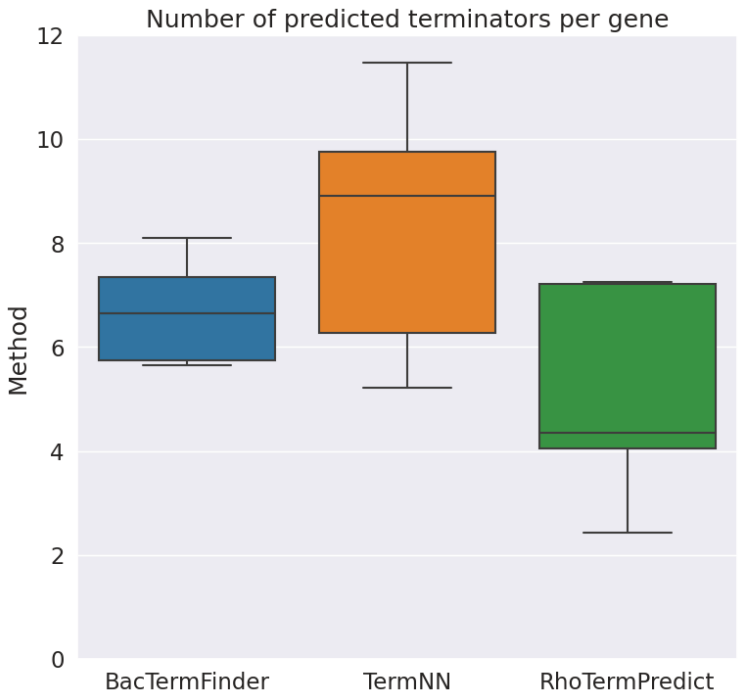

Figure 4.10: Box plot of predicted terminator per gene on BacTermBench. The horizontal line inside each box indicates the median value, and the bottom and top of each box indicate the 25 and 75 percentile, respectively

of TermNN. Figure 4.9 shows that 1) on every overlap threshold, BacTermFinder has a higher recall than TermNN, and 2) BacTermFinder's recall drops at higher overlaps than TermNN's recall. The latter indicates that BacTermFinder can find the location of terminators more accurately than TermNN. However, BacTermFinder's recall sharply decreases at overlap thresholds of 0.9 and 1.0, which suggests the potential need for a nucleotide-wise segmentation approach to achieve accuracy at the nucleotide level.

As we have an incomplete annotation of all terminators in any given bacterial genome, it is not easy to estimate the false positive rate of the software since a predic-

tion might indeed be correct even though a terminator might not have yet been determined experimentally in that location. However, the number of predicted terminators per gene can provide an estimate of the number of false positives. Figure 4.10 shows the distribution of number of terminators predicted per gene for bacterial species included in BacTermBench data by TermNN, RhoTermPredict and BacTermFinder. BacTermFinder displays less variation in the number of predicted terminators per gene across bacterial species. BacTermFinder predicts, on average, 6.62 $\pm$ 1.18 terminators per gene, while TermNN and RhoTermPredict predict 8.89 $\pm$ 3.52 and 4.30 $\pm$ 3.20, respectively. This result and the results provided in Table 4.2 suggest that BacTermFinder's false positive rate is lower than that of TermNN while achieving a higher recall rate.

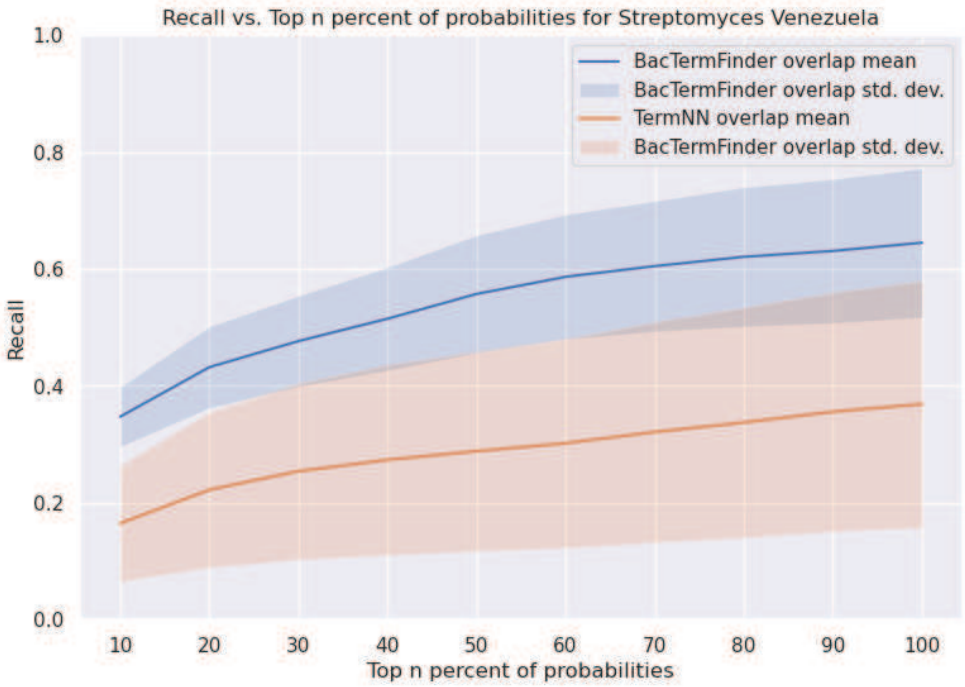

Figure 4.11: Recall versus top n percent of predictions.

We aimed to assess the accuracy of the confidence levels of the TermNN and BacTermFinder in their predictions. The predictions were sorted based on their probabilities, and we selected only the top n% to calculate recall at ten different percentage overlaps with actual terminators. Subsequently, we calculated the mean recall and standard deviation across the percentage overlap levels. Figure 4.11 shows the plot for *Streptomyces venezuela*. We can observe that BacTermFinder has less variation in recall across different overlap thresholds than TermNN. For this bacterium, the worst recall level of BacTermFinder is similar to or higher than the best recall level of TermNN across all top n% of predictions.

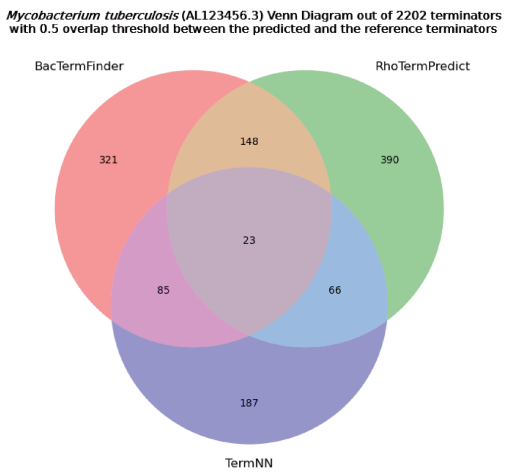

Figure 4.12: Venn Diagram for *M. tuberculosis*

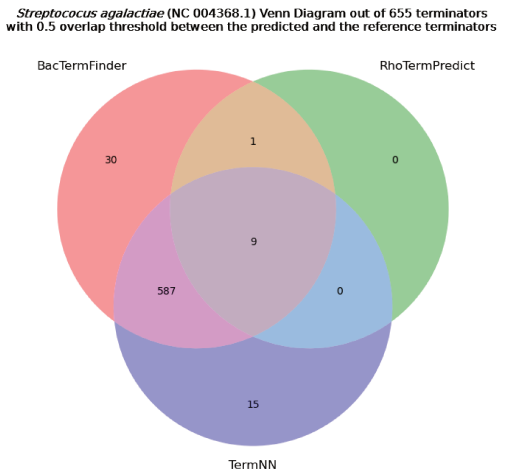

Figure 4.13: Venn Diagram for *S. agalactiae*

To see the agreement among RhoTermFinder, TermNN and BactermFinder, we generated a Venn diagram for bacteria in the extremes of the GC content axis in our BacTermBench. That is *Mycobacterium tuberculosis* with 64.69% GC content and *Streptococus agaletea* with 35.12% GC content. Figures 4.12 and 4.13 show the num-

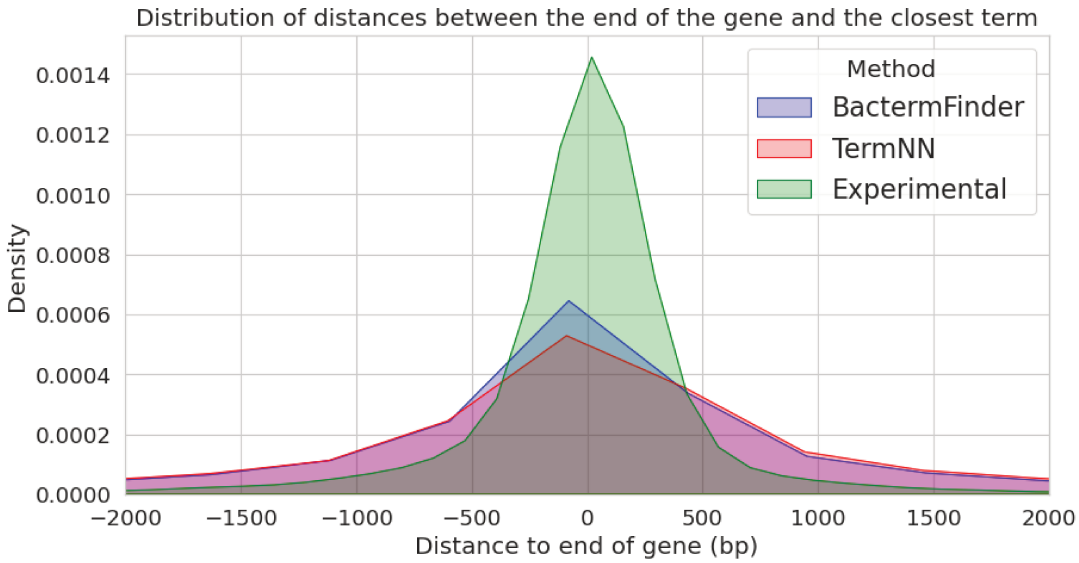

Figure 4.14: Density of the distances to the end of the gene for TermNN and BacTermFinder and experimentally determined terminators.

ber of experimentally determined terminators predicted by each software. In Figure 4.12, most of the agreement is between BacTermFinder and RhoTermPredcit. In Figure 4.13, as it is a low GC bacteria, the most agreement is between BacTermFinder and TermNN (pink area on the bottom left). The number of terminators predicted by the three software is quite low, 23 (or 1.9%) and 9 (or 1.4%) for *M. tuberculosis* and *S. agaletea*, respectively.

Terminators, as the name suggests, are usually at the end of a gene. This inspired us to see where the predicted terminators are relative to the TTS for BacTermFinder and TermNN. We measured the distance of the predicted terminators to the end of the genes and plotted their density in Figure 4.14. The experimentally determined terminators are very close to the end of the genes. Also, BacTermFinder distribution is denser around zero distance compared to the TermNN. This means

that BacTermFinder-predicted terminators are closer to the end of the genes than TermNN-predicted terminators.

4.7 Running Time And Hardware Specifications

The computational requirements and efficiency of BacTermFinder are as follows:

- CPU - Efficiency with eight cores = 40.13%
- RAM - with 10k batches for feature generation: Less than 27.65 GB
- Disk - 2.5 GB for 2 Million base pairs
- Time - 3285 seconds for 2 million base pairs with 10k batch size.
- Average speed 530 100nt-sequences per second

Chapter 5

Conclusion

In this chapter, we list the main contributions of this research project, then discuss the limitations of BacTermFinder, pointing out avenues for future work.

5.1 Summary of Contributions

1. BacTermData: BacTermData is, to our knowledge, currently the largest data set of experimentally verified terminators in bacteria gathered from high-throughput sequencing technologies. This comprehensive dataset includes various sequencing technologies and terminators from diverse bacteria as the data source. We expect this data to be valuable for further developments and analyses.

2. Insights into relevant features for identifying bacterial terminators: We have ranked features based on their relevance for identifying bacterial terminators. This ranking provides insights into the features one should explore for this task.

3. BacTermFinder: BacTermFinder is a general model for finding bacterial terminators, and we have shown that it performs better in terms of recall than other existing software in an independent validation data set containing bacterial and archaeal terminators (BacTermBench). BacTermFinder's average recall over bacterial datasets is 0.5671 ± 0.1919, and TermNN (the second-best software) recall is 0.4550 ± 0.2055. BacTermFinder recall in all prokaryotic data (bacterial and archaeal) is 0.5317 ± 0.1846, and TermNN's is 0.4041 ± 0.2024.

4. Insights into terminator motifs: With the saliency maps of our BacTermFinder model, we could see what nucleotides and positions are more important to recognize a bacterial terminator.

5. Generalizability of BacTermFinder to Archaea: Our BacTermFinder can classify terminators in Archaea, a separated domain of prokaryotes. This suggests similarities between bacterial terminators and archaeal terminators, and our model can find both even though archaeal data was not included in the training.

5.1.1 Limitations

Our work has limitations, but it opens new research doors for BacTermFinder2. Some of those limitations are:

- Small region of interest (100 bp) that does not include factor binding sites, like the Rho binding site.

- Computational efficiency could be improved. New methods like quantization

and knowledge extraction could be done to make the prediction faster. Also, feature generation can be done more efficiently by utilizing GPUs.

- Inability to determine the terminator type: BacTermFinder can find both terminator types but does not indicate which type is found. With new data or different AI explainability methods, this could be achievable.

Bibliography

[1] Thomas J. Santangelo and Irina Artsimovitch. Termination and antitermination: RNA polymerase runs a stop sign. *Nature Reviews Microbiology 2011 9:5*, 9:319–329, 4 2011.

[2] Ananya Ray-Soni, Michael J. Bellecourt, and Robert Landick. Mechanisms of bacterial transcription termination: All good things must end. *https://doi.org/10.1146/annurev-biochem-060815-014844*, 85:319–347, 6 2016.

[3] Ezaz Ahmad, Varsha Mahapatra, V. M. Vanishree, and Valakunja Nagaraja. Intrinsic and Rho-dependent termination cooperate for efficient transcription termination at 3' untranslated regions. *Biochemical and Biophysical Research Communications*, 628:123–132, 11 2022.

[4] Ivan Gusarov and Evgeny Nudler. The Mechanism of Intrinsic Transcription Termination. *Molecular Cell*, 3(4):495–504, April 1999.

[5] Ying Ja Chen, Peng Liu, Alec A.K. Nielsen, Jennifer A.N. Brophy, Kevin Clancy, Todd Peterson, and Christopher A. Voigt. Characterization of 582 nat-

ural and synthetic terminators and quantification of their design constraints. *Nature Methods 2013 10:7*, 10:659–664, 6 2013.

[6] Lislott V. Richardson and John P. Richardson. Rho-dependent Termination of Transcription Is Governed Primarily by the Upstream Rho Utilization (rut) Sequences of a Terminator. *Journal of Biological Chemistry*, 271(35):21597–21603, August 1996.

[7] Emmanuel Skordalakes and James M. Berger. Structure of the Rho transcription terminator: Mechanism of mRNA recognition and helicase loading. *Cell*, 114:135–146, 7 2003.

[8] Zhitai Hao, Vladimir Svetlov, and Evgeny Nudler. Rho-dependent transcription termination: a revisionist view. *Transcription*, 12:171, 2021.

[9] Amber Riaz-Bradley. Transcription in cyanobacteria: a distinctive machinery and putative mechanisms. *Biochemical Society transactions*, 47:679–689, 2019.

[10] Aleksandra Grylak-Mielnicka, Vladimir Bidnenko, Jacek Bardowski, and Elena Bidnenko. Transcription termination factor Rho: a hub linking diverse physiological processes in bacteria. *Microbiology (Reading, England)*, 162:433–447, 2 2016.

[11] Vladimir Svetlov and Evgeny Nudler. Towards the unified principles of transcription termination. *The EMBO Journal*, 39, 2 2020.

[12] Zachary F. Mandell, Rishi K. Vishwakarma, Helen Yakhnin, Katsuhiko S. Murakami, Mikhail Kashlev, and Paul Babitzke. Comprehensive transcription terminator atlas for *Bacillus subtilis*. *Nature Microbiology 2022 7:11*, 7:1918–1931, 10 2022.

[13] Yongxian Fan, Wanru Wang, and Qingqi Zhu. iterb-PPse: Identification of transcriptional terminators in bacterial by incorporating nucleotide properties into PseKNC. *PloS One*, 15(5):e0228479, 2020.

[14] Vivian B. Brandenburg, Franz Narberhaus, and Axel Mosig. Inverse folding based pre-training for the reliable identification of intrinsic transcription terminators. *PLOS Computational Biology*, 18:e1010240, 7 2022.

[15] Magali Naville, Adrien Ghuillot-Gaudeffroy, Antonin Marchais, and Daniel Gautheret. ARNold: A web tool for the prediction of Rho-independent transcription terminators. *http://dx.doi.org/10.4161/rna.8.1.13346*, 8:11–13, 2011.

[16] Swati Gupta and Debnath Pal. Clusters of hairpins induce intrinsic transcription termination in bacteria. *Scientific Reports*, 11(1):16194, August 2021.

[17] Michael Zuker. Mfold web server for nucleic acid folding and hybridization prediction. *Nucleic Acids Research*, 31:3406, 7 2003.

[18] Chao-Qin Feng, Zhao-Yue Zhang, Xiao-Juan Zhu, Yan Lin, Wei Chen, Hua Tang, and Hao Lin. iTerm-PseKNC: a sequence-based tool for predicting bacterial transcriptional terminators. *Bioinformatics*, 35(9):1469–1477, May 2019.

[19] Hao Lin, En-Ze Deng, Hui Ding, Wei Chen, and Kuo-Chen Chou. iPro54-PseKNC: a sequence-based predictor for identifying sigma-54 promoters in prokaryote with pseudo k-tuple nucleotide composition. *Nucleic Acids Research*, 42(21):12961–12972, 10 2014.

[20] Marco Di Salvo, Simone Puccio, Clelia Peano, Stephan Lacour, and Pietro Alifano. RhoTermPredict: an algorithm for predicting Rho-dependent transcription terminators based on *Escherichia coli*, *Bacillus subtilis* and *Salmonella enterica* databases. *BMC bioinformatics*, 20(1):117, March 2019.

[21] Cédric Nadiras, Eric Eveno, Annie Schwartz, Nara Figueroa-Bossi, and Marc Boudvillain. A multivariate prediction model for Rho-dependent termination of transcription. *Nucleic Acids Research*, 46(16):8245–8260, September 2018.

[22] Adi Millman, Daniel Dar, Maya Shamir, and Rotem Sorek. Computational prediction of regulatory, premature transcription termination in bacteria. *Nucleic Acids Research*, 45(2):886–893, January 2017.

[23] Daniel Dar, Maya Shamir, J. R. Mellin, Mikael Koutero, Noam Stern-Ginossar, Pascale Cossart, and Rotem Sorek. Term-seq reveals abundant ribo-regulation of antibiotics resistance in bacteria. *Science*, 352(6282):aad9822, 2016.

[24] Paul P. Gardner, Lars Barquist, Alex Bateman, Eric P. Nawrocki, and Zasha Weinberg. RNIE: genome-wide prediction of bacterial intrinsic terminators. *Nucleic Acids Research*, 39(14):5845–5852, August 2011.

[25] Carleton L. Kingsford, Kunmi Ayanbule, and Steven L. Salzberg. Rapid, accurate, computational discovery of Rho-independent transcription terminators illuminates their relationship to DNA uptake. *Genome Biology*, 8:1–12, 2 2007.

[26] Xiangwu Ju, Dayi Li, and Shixin Liu. Full-length RNA profiling reveals pervasive bidirectional transcription terminators in bacteria. *Nature microbiology*, 4:1907, 11 2019.

[27] Bo Yan, Matthew Boitano, Tyson A. Clark, and Laurence Ettwiller. SMRT-Cappable-seq reveals complex operon variants in bacteria. *Nature Communications*, 9, 12 2018.

[28] Jean Benoît Lalanne, James C. Taggart, Monica S. Guo, Lydia Herzel, Ariel Schieler, and Gene Wei Li. Evolutionary convergence of pathway-specific enzyme expression stoichiometry. *Cell*, 173:749–761.e38, 4 2018.

[29] Alexander A. Shishkin, Georgia Giannoukos, Alper Kucukural, Dawn Ciulla, Michele Busby, Christine Surka, Jenny Chen, Roby P. Bhattacharyya, Robert F. Rudy, Milesh M. Patel, Nathaniel Novod, Deborah T. Hung, Andreas Gnirke, Manuel Garber, Mitchell Guttman, and Jonathan Livny. Simultaneous generation of many RNA-seq libraries in a single reaction. *Nature methods*, 12:323, 3 2015.

[30] Cynthia M. Sharma and Jörg Vogel. Differential RNA-seq: the approach behind and the biological insight gained. *Current Opinion in Microbiology*, 19:97–105, 6 2014.

[31] National center for biotechnology information (NCBI) pubmed. https://pubmed.ncbi.nlm.nih.gov/, [1988] – [2023]. Accessed: 2023-11-10.

[32] National center for biotechnology information (NCBI) gene expression omnibus (GEO). https://www.ncbi.nlm.nih.gov/geo/, 1999 – [2023]. Accessed: 2023-11-10.

[33] Socorro Gama-Castro, Heladia Salgado, Alberto Santos-Zavaleta, Daniela Ledezma-Tejeida, Luis Muñiz-Rascado, Jair Santiago García-Sotelo, Kevin Alquicira-Hernández, Irma Martínez-Flores, Lucia Pannier, Jaime Abraham Castro-Mondragón, Alejandra Medina-Rivera, Hilda Solano-Lira, César Bonavides-Martínez, Ernesto Pérez-Rueda, Shirley Alquicira-Hernández, Liliana Porrón-Sotelo, Alejandra López-Fuentes, Anastasia Hernández-Koutoucheva, Víctor Del Moral-Chavez, Fabio Rinaldi, and Julio Collado-Vides. RegulonDB version 9.0: high-level integration of gene regulation, coexpression, motif clustering and beyond. *Nucleic Acids Research*, 44:D133–D143, 1 2016.

[34] Takahiro Ishii, Ken Ichi Yoshida, Goro Terai, Yasutaro Fujita, and Kenta Nakai. DBTBS: a database of *Bacillus subtilis* promoters and transcription factors. *Nucleic Acids Research*, 29:278–280, 1 2001.

[35] Adrian Sven Geissler, Christian Anthon, Ferhat Alkan, Enrique González-Tortuero, Line Dahl Poulsen, Thomas Beuchert Kallehauge, Anne Breüner, Stefan Ernst Seemann, Jeppe Vinther, and Jan Gorodkin. BSGatlas: a unified

Bacillus subtilis genome and transcriptome annotation atlas with enhanced information access. *Microbial Genomics*, 7:524, 2021.

[36] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, and Carol Willing. Jupyter notebooks – a publishing format for reproducible computational workflows. In F. Loizides and B. Schmidt, editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87 – 90. IOS Press, 2016.

[37] The pandas development team. pandas-dev/pandas: Pandas, February 2020. `https://doi.org/10.5281/zenodo.3509134`.

[38] W. James Kent, Charles W. Sugnet, Terrence S. Furey, Krishna M. Roskin, Tom H. Pringle, Alan M. Zahler, , and David Haussler. The human genome browser at UCSC. *Genome Research*, 12:996, 6 2002.

[39] Aaron R. Quinlan and Ira M. Hall. BEDTools: a flexible suite of utilities for comparing genomic features. *Bioinformatics (Oxford, England)*, 26:841–842, 1 2010.

[40] Michael J. Bottery. Ecological dynamics of plasmid transfer and persistence in microbial communities. *Current Opinion in Microbiology*, 68:None, 8 2022.

[41] Ruben Chevez-Guardado and Lourdes Peña-Castillo. Promotech: a general tool for bacterial promoter recognition. *Genome Biology*, 22:1–16, 12 2021.

[42] Daniel Dar, Daniela Prasse, Ruth A. Schmitz, and Rotem Sorek. Widespread formation of alternative 3' UTR isoforms via transcription termination in archaea. *Nature Microbiology 2016 1:10*, 1:1–9, 8 2016.

[43] Yongjae Lee, Namil Lee, Soonkyu Hwang, Woori Kim, Suhyung Cho, Bernhard O. Palsson, and Byung Kwan Cho. Genome-scale analysis of genetic regulatory elements in *Streptomyces avermitilis MA-4680* using transcript boundary information. *BMC Genomics*, 23, 12 2022.

[44] Yongjae Lee, Namil Lee, Yujin Jeong, Soonkyu Hwang, Woori Kim, Suhyung Cho, Bernhard O. Palsson, and Byung Kwan Cho. The transcription unit architecture of *Streptomyces lividans TK24*. *Frontiers in Microbiology*, 10:2074, 9 2019.

[45] Soonkyu Hwang, Namil Lee, Donghui Choe, Yongjae Lee, Woori Kim, Yujin Jeong, Suhyung Cho, Bernhard O. Palsson, and Byung-Kwan Cho. Elucidating the regulatory elements for transcription termination and posttranscriptional processing in the *Streptomyces clavuligerus* genome. *mSystems*, 6, 6 2021.

[46] Philip P. Adams, Gabriele Baniulyte, Caroline Esnault, Kavya Chegireddy, Navjot Singh, Molly Monge, Ryan K. Dale, Gisela Storz, and Joseph T. Wade. Regulatory roles of *Escherichia coli* 5' UTR and ORF-internal RNAs detected by 3' end mapping. *eLife*, 10:1–33, 1 2021.

[47] Donghui Choe, Kangsan Kim, Minjeong Kang, Seung Goo Lee, Suhyung Cho, Bernhard Palsson, and Byung Kwan Cho. Synthetic 3'-UTR valves for optimal

metabolic flux control in *Escherichia coli*. *Nucleic Acids Research*, 50:4171, 4 2022.

[48] Hiraku Takada, Zachary F. Mandell, Helen Yakhnin, Anastasiya Glazyrina, Shinobu Chiba, Tatsuaki Kurata, Kelvin J.Y. Wu, Ben I.C. Tresco, Andrew G. Myers, Gemma C. Aktinson, Paul Babitzke, and Vasili Hauryliuk. Expression of *Bacillus subtilis* ABCF antibiotic resistance factor vmlr is regulated by RNA polymerase pausing, transcription attenuation, translation attenuation and (p)ppgpp. *Nucleic Acids Research*, 50:6174, 6 2022.

[49] Zachary F. Mandell, Reid T. Oshiro, Alexander V. Yakhnin, Rishi Vishwakarma, Mikhail Kashlev, Daniel B. Kearns, and Paul Babitzke. NusG is an intrinsic transcription termination factor that stimulates motility and coordinates gene expression with NusA. *eLife*, 10, 2021.

[50] Grace E. Johnson, Jean Benoît Lalanne, Michelle L. Peters, and Gene Wei Li. Functionally uncoupled transcription-translation in *Bacillus subtilis*. *Nature*, 585:124, 9 2020.

[51] Soonkyu Hwang, Namil Lee, Donghui Choe, Yongjae Lee, Woori Kim, Ji Hun Kim, Gahyeon Kim, Hyeseong Kim, Neung Ho Ahn, Byoung Hee Lee, Bernhard O. Palsson, and Byung Kwan Cho. System-level analysis of transcriptional and translational regulatory elements in *Streptomyces griseus*. *Frontiers in Bioengineering and Biotechnology*, 10, 2 2022.

[52] Jessica M. Vera, Indro Neil Ghosh, Yaoping Zhang, Alex S. Hebert, Joshua J. Coon, and Robert Landick. Genome-scale transcription-translation mapping reveals features of *Zymomonas mobilis* transcription units and promoters. *mSystems*, 5, 8 2020.

[53] Yongjae Lee, Namil Lee, Soonkyu Hwang, Woori Kim, Yujin Jeong, Suhyung Cho, Bernhard O. Palsson, and Byung Kwan Cho. Genome-scale determination of 5´ and 3´ boundaries of RNA transcripts in *Streptomyces genomes*. *Scientific Data*, 7, 12 2020.

[54] Maureen K. Thomason, Maya Voichek, Daniel Dar, Victoria Addis, David Fitzgerald, Susan Gottesman, Rotem Sorek, and E. Peter Greenberga. A rhli 5' UTR-derived sRNA regulates RhlR-dependent quorum sensing in *Pseudomonas aeruginosa*. *mBio*, 10, 2019.

[55] Daniel G. Mediati, Julia L. Wong, Wei Gao, Stuart McKellar, Chi Nam Ignatius Pang, Sylvania Wu, Winton Wu, Brandon Sy, Ian R. Monk, Joanna M. Biazik, Marc R. Wilkins, Benjamin P. Howden, Timothy P. Stinear, Sander Granneman, and Jai J. Tree. RNase III-CLASH of multi-drug resistant *Staphylococcus aureus* reveals a regulatory mRNA 3'UTR required for intermediate vancomycin resistance. *Nature Communications*, 13, 12 2022.

[56] Indu Warrier, Nikhil Ram-Mohan, Zeyu Zhu, Ariana Hazery, Haley Echlin, Jason Rosch, Michelle M. Meyer, and Tim van Opijnen. The transcriptional landscape of *Streptococcus pneumoniae TIGR4* reveals a complex operon archi-

tecture and abundant riboregulation critical for growth and virulence. *PLoS Pathogens*, 14, 12 2018.

[57] Jean Benoît Lalanne, James C. Taggart, Monica S. Guo, Lydia Herzel, Ariel Schieler, and Gene Wei Li. Evolutionary convergence of pathway-specific enzyme expression stoichiometry. *Cell*, 173:749, 4 2018.

[58] Manuela Fuchs, Vanessa Lamm-Schmidt, Johannes Sulzer, Falk Ponath, Laura Jenniches, Joseph A. Kirk, Robert P. Fagan, Lars Barquist, Jörg Vogel, and Franziska Faber. An RNA-centric global view of *Clostridioides difficile* reveals broad activity of Hfq in a clinically important gram-positive bacterium. *Proceedings of the National Academy of Sciences of the United States of America*, 118, 6 2021.

[59] Laurène Bastet, Pilar Bustos-Sanmamed, Arancha Catalan-Moreno, Carlos J. Caballero, Sergio Cuesta, Leticia Matilla-Cuenca, Maite Villanueva, Jaione Valle, Iñigo Lasa, and Alejandro Toledo-Arana. Regulation of heterogenous LexA expression in *Staphylococcus aureus* by an antisense RNA originating from transcriptional read-through upon natural mispairings in the sbrB intrinsic terminator. *International Journal of Molecular Sciences*, 23, 1 2022.

[60] Jelle Slager, Rieza Aprianto, and Jan Willem Veening. Deep genome annotation of the opportunistic human pathogen *Streptococcus pneumoniae D39*. *Nucleic Acids Research*, 46:9971, 11 2018.

[61] Daniel Dar and Rotem Sorek. High-resolution RNA 3'-ends mapping of bacterial Rho-dependent transcripts. *Nucleic Acids Research*, 46:6797, 7 2018.

[62] Mohamad Al kadi, Eiji Ishii, Dang Tat Truong, Daisuke Motooka, Shigeaki Matsuda, Tetsuya Iida, Toshio Kodama, and Daisuke Okuzaki. Direct RNA sequencing unfolds the complex transcriptome of *Vibrio parahaemolyticus*. *mSystems*, 6, 12 2021.

[63] Michael J.L. De Hoon, Yuko Makita, Kersta Nakai, and Satora Msyasio. Prediction of transcriptional terminators in *Bacillus subtilis* and related species. *PLoS Computational Biology*, 1:0212–0221, 2 2005.

[64] Raphaël Forquet, Xuejiao Jiang, William Nasser, Florence Hommais, Sylvie Reverchon, and Sam Meyer. Mapping the complex transcriptional landscape of the Phytopathogenic bacterium *Dickeya dadantii*. *mBio*, 13, 6 2022.

[65] Alberto Santos-Zavaleta, Heladia Salgado, Socorro Gama-Castro, Mishael Sánchez-Pérez, Laura Gómez-Romero, Daniela Ledezma-Tejeida, Jair Santiago García-Sotelo, Kevin Alquicira-Hernández, Luis José Muñiz-Rascado, Pablo Peña-Loredo, Cecilia Ishida-Gutiérrez, David A. Velázquez-Ramírez, Víctor Del Moral-Chávez, César Bonavides-Martínez, Carlos Francisco Méndez-Cruz, James Galagan, and Julio Collado-Vides. RegulonDB v 10.5: tackling challenges to unify classic and high throughput knowledge of gene regulation in *E. coli K-12*. *Nucleic Acids Research*, 47:D212, 1 2019.

[66] Alexandre D ' Halluin, Peter Polgar, Terry Kipkorir, Zaynah Patel, Teresa Cortes, and Kristine B Arnvig. Conditional termination of transcription is shaped by Rho and translated uORFS in *Mycobacterium tuberculosis*. *bioRxiv*, page 2022.06.01.494293, 3 2023.

[67] Isabelle Rosinski-Chupin, Elisabeth Sauvage, Odile Sismeiro, Adrien Villain, Violette Da Cunha, Marie Elise Caliot, Marie Agnès Dillies, Patrick Trieu-Cuot, Philippe Bouloc, Marie Frédérique Lartigue, and Philippe Glaser. Single nucleotide resolution RNA-seq uncovers new regulatory mechanisms in the opportunistic pathogen *Streptococcus agalactiae*. *BMC Genomics*, 16, 5 2015.

[68] Sang-Hyeok Cho, Yujin Jeong, Seong-Joo Hong, Hookeun Lee, Hyung-Kyoon Choi, Dong-Myung Kim, Choul-Gyun Lee, Suhyung Cho, and Byung-Kwan Cho. Different regulatory modes of *Synechocystis sp. PCC 6803* in response to photosynthesis inhibitory conditions. *mSystems*, 6, 12 2021.

[69] Yujin Jeong, Seong Joo Hong, Sang Hyeok Cho, Seonghoon Yoon, Hookeun Lee, Hyung Kyoon Choi, Dong Myung Kim, Choul Gyun Lee, Suhyung Cho, and Byung Kwan Cho. Multi-omic analyses reveal habitat adaptation of marine cyanobacterium *Synechocystis sp. PCC 7338*. *Frontiers in Microbiology*, 12, 5 2021.

[70] Alexandre D ' Halluin, Peter Polgar, Terry Kipkorir, Zaynah Patel, Teresa Cortes, and Kristine B Arnvig. Conditional termination of transcription is

shaped by Rho and translated uORFS in *Mycobacterium tuberculosis*. *bioRxiv*, page 2022.06.01.494293, 3 2023.

[71] Sarah J. Berkemer, Lisa Katharina Maier, Fabian Amman, Stephan H. Bernhart, Julia Wörtz, Pascal Märkle, Friedhelm Pfeiffer, Peter F. Stadler, and Anita Marchfelder. Identification of RNA 3´ ends and termination sites in *Haloferax volcanii*. *RNA Biology*, 17:663, 5 2020.

[72] Jie Li, Lei Yue, Zhihua Li, Wenting Zhang, Bing Zhang, Fangqing Zhao, and Xiuzhu Dong. aCPSF1 cooperates with terminator U-tract to dictate archaeal transcription termination efficacy. *eLife*, 10:70464, 12 2021.

[73] Robson P. Bonidia, Douglas S. Domingues, Danilo S. Sanches, and André C.P.L.F. De Carvalho. MathFeature: feature extraction package for DNA, RNA and protein sequences based on mathematical descriptors. *Briefings in Bioinformatics*, 23:1–10, 1 2022.

[74] Zhen Chen, Pei Zhao, Chen Li, Fuyi Li, Dongxu Xiang, Yong Zi Chen, Tatsuya Akutsu, Roger J. Daly, Geoffrey I. Webb, Quanzhi Zhao, Lukasz Kurgan, and Jiangning Song. iLearnPlus: a comprehensive and automated machine-learning platform for nucleic acid and protein sequence analysis, prediction and visualization. *Nucleic Acids Research*, 49:e60–e60, 6 2021.

[75] Bin Liu, Fule Liu, Longyun Fang, Xiaolong Wang, and Kuo Chen Chou. repDNA: a Python package to generate various modes of feature vectors for

DNA sequences by incorporating user-defined physicochemical properties and sequence-order effects. *Bioinformatics (Oxford, England)*, 31:1307–1309, 4 2015.

[76] Scott M. Lundberg, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M. Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. From local explanations to global understanding with explainable ai for trees. *Nature Machine Intelligence*, 2(1):2522–5839, 2020.

[77] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. LightGBM: A highly efficient gradient boosting decision tree. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[78] Robert R. Sokal and Barbara A. Thomson. Population structure inferred by local spatial autocorrelation: an example from an amerindian tribal population. *American journal of physical anthropology*, 129:121–131, 1 2006.

[79] David S. Horne. Prediction of protein helix content from an autocorrelation analysis of sequence hydrophobicities. *Biopolymers*, 27:451–477, 1988.

[80] Zhen Chen, Pei Zhao, Fuyi Li, Tatiana T. Marquez-Lago, André Leier, Jerico Revote, Yan Zhu, David R. Powell, Tatsuya Akutsu, Geoffrey I. Webb, Kuo Chen Chou, A. Ian Smith, Roger J. Daly, Jian Li, and Jiangning Song. iLearn: an integrated platform and meta-learner for feature engineering,

machine-learning analysis and modeling of DNA, RNA and protein sequence data. *Briefings in bioinformatics*, 21:1047–1057, 5 2020.

[81] Bin Liu, Fule Liu, Longyun Fang, Xiaolong Wang, and Kuo Chen Chou. repDNA: a python package to generate various modes of feature vectors for DNA sequences by incorporating user-defined physicochemical properties and sequence-order effects. *Bioinformatics*, 31:1307–1309, 4 2015.

[82] Bin Liu, Fule Liu, Xiaolong Wang, Junjie Chen, Longyun Fang, and Kuo Chen Chou. Pse-in-One: a web server for generating various modes of pseudo components of DNA, RNA, and protein sequences. *Nucleic Acids Research*, 43:W65, 7 2015.

[83] Qiwen Dong, Shuigeng Zhou, and Jihong Guan. A new taxonomy-based protein fold recognition approach based on autocross-covariance transformation. *Bioinformatics (Oxford, England)*, 25:2655–2662, 2009.

[84] Yanzhi Guo, Lezheng Yu, Zhining Wen, and Menglong Li. Using support vector machine combined with auto covariance to predict protein-protein interactions from protein sequences. *Nucleic acids research*, 36:3025–3030, 5 2008.

[85] Feng Gao and Chun Ting Zhang. Comparison of various algorithms for recognizing short coding sequences of human genes. *Bioinformatics (Oxford, England)*, 20:673–681, 3 2004.

[86] Bin Liu, Xin Gao, and Hanyu Zhang. BioSeq-Analysis2.0: an updated platform for analyzing DNA, RNA and protein sequences at sequence level and residue

level based on machine learning approaches. *Nucleic acids research*, 47:E127–E127, 11 2019.

[87] John G. Doench, Nicolo Fusi, Meagan Sullender, Mudra Hegde, Emma W. Vaimberg, Katherine F. Donovan, Ian Smith, Zuzana Tothova, Craig Wilen, Robert Orchard, Herbert W. Virgin, Jennifer Listgarten, and David E. Root. Optimized sgRNA design to maximize activity and minimize off-target effects of CRISPR-Cas9. *Nature biotechnology*, 34:184–191, 2 2016.

[88] Wei Chen, Hong Tran, Zhiyong Liang, Hao Lin, and Liqing Zhang. Identification and analysis of the N(6)-methyladenosine in the *Saccharomyces cerevisiae* transcriptome. *Scientific reports*, 5, 9 2015.

[89] Leyi Wei, Ran Su, Shasha Luan, Zhijun Liao, Balachandran Manavalan, Quan Zou, and Xiaolong Shi. Iterative feature representations improve N4-methylcytosine site prediction. *Bioinformatics (Oxford, England)*, 35:4930–4937, 12 2019.

[90] Zhen Chen, Yuan Zhou, Jiangning Song, and Ziding Zhang. hcksaap_ubsite: improved prediction of human ubiquitination sites by exploiting amino acid pattern and properties. *Biochimica et biophysica acta*, 1834:1461–1467, 2013.

[91] Zhen Chen, Yong Zi Chen, Xiao Feng Wang, Chuan Wang, Ren Xiang Yan, and Ziding Zhang. Prediction of ubiquitination sites by using the composition of k-spaced amino acid pairs. *PloS one*, 6, 2011.

[92] William Stafford Noble, Scott Kuehn, Robert Thurman, Man Yu, and John Stamatoyannopoulos. Predicting the in vivo signature of human gene regulatory sequences. *Bioinformatics (Oxford, England)*, 21 Suppl 1, 2005.

[93] Shobhit Gupta, Jonathan Dennis, Robert E. Thurman, Robert Kingston, John A. Stamatoyannopoulos, and William Stafford Noble. Predicting human nucleosome occupancy from primary sequence. *PLoS computational biology*, 4, 2008.

[94] Xiaoli Qiang, Huangrong Chen, Xiucai Ye, Ran Su, and Leyi Wei. M6AMRFS: Robust prediction of N6-methyladenosine sites with sequence-based features in multiple species. *Frontiers in genetics*, 9, 10 2018.

[95] Qing Song Xu and Yi Zeng Liang. Monte carlo cross validation. *Chemometrics and Intelligent Laboratory Systems*, 56:1–11, 4 2001.

[96] Yann LeCun, Y. Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–44, 05 2015.

[97] Leo Grinsztajn, Edouard Oyallon, and Gael Varoquaux. Why do tree-based models still outperform deep learning on typical tabular data?, 6 2022.

[98] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[99] Delano Thomas and Lourdes Pena Castillo. Github - bioinformaticslabatmun/promotech-cnn: Bacterial promoter prediction using a cnn.

[100] François Chollet et al. Keras. `https://keras.io`, 2015.

[101] Randal S. Olson, William La Cava, Zairah Mustahsan, Akshay Varik, and Jason H. Moore. Data-driven advice for applying machine learning to bioinformatics problems. *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, 23:192, 2018.

[102] E. V. Wong and Open Textbook Library. Cells : molecules and mechanisms. pages 1–276, 2009.