Spring 5-24-2019

Detecting CRISPR Arrays Using Long-Short Term Memory Network

Shantanu Deshmukh
San Jose State University

Detecting CRISPR Arrays Using Long-Short Term Memory Network

A Project

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Shantanu Deshmukh

May 2019

The Designated Project Committee Approves the Project Titled

Detecting CRISPR Arrays Using Long-Short Term Memory Network

by

Shantanu Deshmukh

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSÉ STATE UNIVERSITY

May 2019

Dr. Sami Khuri	Department of Computer Science
Dr. Natalia Khuri	Department of Bioengineering, Stanford University
Dr. Philip Heller	Department of Computer Science

ABSTRACT

Detecting CRISPR Arrays Using Long-Short Term Memory Network

by Shantanu Deshmukh

CRISPR (Clustered Regularly Interspaced Short Palindromic Repeat) is a sequence found in the DNA sequence of an organism. It provides provides immunity to the organism. Recently, it was found that the CRISPR-based immunity mechanism can be manipulated to perform genome editing. The problem is, it is hard to know the specificity of this system and in turn, making it highly specific is difficult. More research is required to improve this CRISPR-based genome editing. Detecting CRISPR arrays in the DNA sequence is the first step towards this research. In this work, a CRISPR array detection pipeline, CRISPRLstm, is proposed. CRISPRLstm leverages the power of artificial intelligence to improve its performance over existing CRISPR array detection programs. Why and how artificial intelligence, or specifically, Long-Short Term Memory (LSTM) models, can be used to tackle this problem effectively is explained in this report. The CRSIPR arrays detected by CRISPRLstm are in good agreement with other widely used and freely available CRISPR array detection tools. CRISPRLstm is available in form of a web-tool. It visualizes the detected CRISPR arrays in a highly interactive interface with options to view secondary structure of the repeat and spacer sequences, blast them, create sequence logos of repeat sequences, and more.

ACKNOWLEDGMENTS

I would like to thank my advisior Dr. Sami Khuri, for his continuous support, patience and encouragement. I really appreciate the efforts he took to conduct CRISPR-sessions during the summer break and arranging special sessions with the expert professors from the Biology department; both of which aroused my interest in CRISPR and its associated computational challenges.

A very special thanks to Dr. Natalia Khuri. I was extremely fortunate to have had a chance to work with her. Her guidance and feedback were very valuable. This was only possible because of her encouragement and continuous support.

Finally, I would like to thank Dr. Philip Heller for overseeing the progress throughout the course of this project and providing timely feedback.

TABLE OF CONTENTS

CHAPTER

1 Introduction 1

1.1 Background 1

1.2 Problem Definition 2

1.3 Report Organization 2

2 Literature Survey 4

2.1 CRF: Detection of CRISPR arrays using random forest classifier . 4

2.1.1 Performance analysis 6

2.1.2 Strengths and weaknesses 6

2.2 CRISPRDetect: A flexible algorithm to define CRISPR arrays . . 7

2.2.1 Performance Analysis 9

2.2.2 Strengths and weaknesses 10

2.3 CRISPRFinder: A web tool identify CRISPRs 10

2.3.1 Performance Analysis 11

2.3.2 Strengths and weaknesses 12

2.4 CRISPR Recognition Tool (CRT): a tool for detection of CRISPRs 12

2.4.1 Performance Analysis 13

2.4.2 Strengths and Weaknesses 13

3 Long-Short Term Memory models 15

3.1 What is a Long-Short Term Memory model? 15

3.2 LSTMs on DNA Sequences 16

4 CRISPR array detection using CRISPRLstm 17

4.1 Dataset description 17

4.2 CRISPR array detection pipeline 18

4.3 An Example 21

5 Experiments and Results 23

5.1 Comparison with CRF 23

5.2 rCRISPR metric on 8 known genome sequences 25

5.3 Comparison with CRT, PILER-CR and CRISPRDetect 26

5.4 Visualizing CRISPR array distribution detected by various programs 28

5.5 Specific Examples 29

5.6 User Interface 30

6 Conclusion 34

LIST OF REFERENCES 35

LIST OF TABLES

1	Published articles on CRISPR array detection	4
2	Curated Dataset's Statistics	17
3	CRF [1] Dataset's Statistics	17
4	Fold-wise classification accuracy of random forest and LSTM on the CRF's dataset	23
5	Fold-wise classification accuracy of random forest and LSTM on the self-curated dataset	23
6	Performance comparison between random forest and LSTM classifier	23
7	t-test result of 100 10-fold cross validation's on [1]'s dataset . . .	25
8	No. of CRISPR arrays detected by different programs on 8 genomic sequences	25

LIST OF FIGURES

1	Structure of a CRISPR Sequence	1
2	Stem loop structure of the repeat sequence [1].	5
3	Recurrent Neural Network [2]	15
4	Implementation pipeline of CRISPRLstm.	18
5	Architecture of the LSTM model used for scoring.	20
6	Fold-wise classification accuracy graphs	24
7	Performance measurement of CRISPR Detection programs	26
8	Visualizing the number of CRISPR arrays detected by CRT, PILER-CR, CRISPRDetect and CRISPRLstm on Godde and Bickerton's dataset	27
9	rCRISPR plot on Godde & Bickerton's dataset	28
10	Circos plots of detected CRISPR arrays in two genomes. The outermost circle represents the genome sequence. The gap on the upper-side of each circle is the start and end positions of the sequence. Each solid circle represents CRISPR arrays detected by PILER-CR (blue), CRISPRDetect (red), CRT (green) and CRISPRLstm (black) from the innermost to the outermost circle respectively.	29
11	Results page	31
12	Repeat weblogo popup	31
13	CRISPR Structure popup	32
14	Secondary structure of the clicked sequence with an option to BLAST the sequence	33

CHAPTER 1

Introduction

1.1 Background

CRISPR (Clustered Regularly Interspaced Short Palindromic Repeats) is a special nucleotide sequence in the DNA sequence which contains a short subsequence that is repeated at regular intervals. This short subsequence is called repeat sequence(repeats). The sequence between the repeats is called the spacer sequence. As an example, consider a DNA sequence 'XXXXRSRSRSRXXX' where X is a random sequence of nucleotides, R is the repeat sequence and S is the spacer sequence. The repeat sequences appear after regular intervals and in between them is the spacer sequence. Figure 1 shows the structure of the CRISPR sequence. Moreover, the repeat sequences are palindromic in nature. Thus, this special sequence 'RSRSRSR' has the name Clustered Regularly Interspaced Short Palindromic Repeats or CRISPR.

CRISPR and its associated system (Cas) is part of an organism's defense mechanism [3]. The spacer sequences in the CRISPR array are usually a part of a viral DNA. An organism uses these spacer sequences as templates to identify similar viral sequences in its DNA. When a similar sequence is found, the Cas system degenerates it, thus protecting the organism from the viral attack. Recently, the CRISPR system is gaining popularity as it can be engineered to perform low-cost genome editing [1]. It is important that this genome editing is precise and happens at the intended locations otherwise the consequences will be fatal for an organism. To improve the accuracy

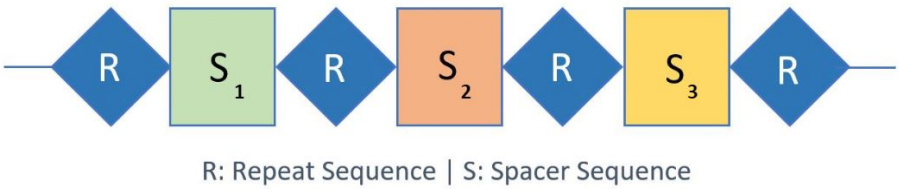


Figure 1: Structure of a CRISPR Sequence

of the CRISPR system more research needs to be conducted. The prerequisite for this analysis is the detection of CRISPR arrays in an organism's DNA sequence [4]. Detecting CRISPR arrays in the DNA sequence is the focus of this report.

1.2 Problem Definition

The key problem that is being tackled in this report is, given a DNA sequence, finding the number of CRISPR arrays present in the sequence, if any. And, for each CRISPR array, find its exact starting and ending positions along with the exact boundary of each of its spacer and repeat sequences.

Detecting CRISPR arrays in the DNA sequence is important because it will allow researchers to design efficient genome editing mechanisms [1]. It will also help them better understand the evolution of an organism within the species since organisms with shared ancestry will have identical spacer sequences [1]. Using CRISPR arrays, researchers can study spacer sequences and the viral DNAs to which the spacer sequences belong [5]. Moreover, the Cas genes are near the CRISPR array, and can be studied as well [3].

1.3 Report Organization

The report is organized as follows: Chapter 2 covers the literature survey, mainly, four existing articles, that focus on CRISPR Array detection, namely, CRISPR Random Forest [1], CRISPRDetect [4], CRISPRFinder [4] and CRT [6]. Each article has a section dedicated to it, which covers the performance analysis and the strengths and weaknesses of the detection methods proposed. Chapter 3 explains the Long short-term memory (LSTM) model and why it can be effective in detecting CRISPR sequences. Chapter 4 describes the proposed CRISPR Detection method (CRISPRLstm) along with an architecture diagram and a simple toy example to clearly understand the detection process. Chapter 5 then covers the results of the

experiments that were performed using CRISPRLstm. Finally, Chapter 6 concludes with a comparison of CRISPRLstm to the existing approaches and future work.

CHAPTER 2

Literature Survey

A lot of work is being done targeting this specific problem. Table 1 mentions six articles that suggest various ways to find CRISPR arrays in a genome sequence. Later in this chapter, some CRISPR detection techniques are explained in details.

Table 1: Published articles on CRISPR array detection

Published Year	Method	Validation Dataset	Software available as	Reference
2007	PILER-CR	Jensen et al. [7] and Godde & Bickerton [8]	Stand-alone app	[9]
2007	CRT	5 Organisms with accession numbers: AE015450, AE004439, AE017282, AP006627, BX470251 27 randomly selected species from 101 species of Godde and Bickerton [8] documented CRISPRs	Stand-alone and Web-service app	[6]
2007	CRISPRFinder	Godde & Bickerton [8]	Web-service app	[3]
2016	CRISPRDetect	Bacterial and archaeal genomes from GenBank/genomes (5262 sequences) CRISPRDb [10]	Stand-alone and Web-service app	[4]
2016	CRISPRDigger	Clostridium genus (bacterium) & Methanocaldococcus genus (archeal)	Stand-alone app	[5]
2017	CRF	CRISPRDb [10]	Web-service app	[1]

2.1 CRF: Detection of CRISPR arrays using random forest classifier

Wang and Liang [1] have applied a random forest classifier to identify the valid CRISPR arrays based on the repeat sequence and the structural features of CRISPR. Their method has 3 major steps:

1. Search CRISPR candidates
2. Identify valid CRISPR arrays from the candidates
3. Remove tandem repeats

In the first step, they search for candidate CRISPR arrays. For that, they use an existing tool, called CRISPR recognition tool (CRT) proposed by Bland et al. [6], with lenient default parameters. CRT recognizes CRISPR by first detecting a repeating

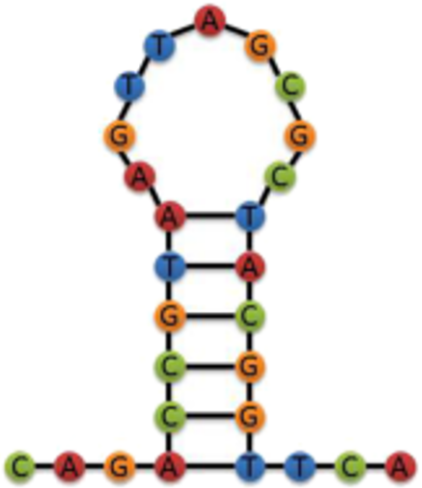


Figure 2: Stem loop structure of the repeat sequence [1].

seed region and then expanding it one base pair at a time ensuring a certain matching threshold is met after each expansion. To detect all the CRISPR candidates, the authors modified 2 parameters. They reduced the default length requirement of the seed region and reduced the matching threshold. This resulted in the detection of all the probable CRISPR arrays.

For the second step, Wang and Liang trained a random forest classifier to detect valid repeat sequences. The main idea behind the algorithm is that, if a candidate CRISPR array has a valid repeat sequence, it is a valid CRISPR array. Since a repeat sequence is palindromic in nature, it forms a stable stem loop, as shown in Figure 2 [1]. This structural information of the repeat sequence is cleverly extracted with the help of triplets and is used for training. Once the random forest classifier model is trained, it is used to detect valid repeat sequences among the candidate CRISPR arrays detected in the first step.

Finally, in the third step, the valid CRISPR arrays are further analyzed, this time as a complete sequence. The similarity between spacers is calculated using Hamming distance and if two spacers are found to be more than 50% similar in a sequence, then

it is discarded as an invalid CRISPR. Each valid sequence is also checked for being a tandem repeat. A tandem repeat occurs when a pattern of nucleotides is repeated, and the repetitions are directly adjacent to each other.

2.1.1 Performance analysis

Wang and Liang [1] claim that the proposed random forest classifier can differentiate real CRISPR repeats from invalid CRISPR repeats. They compared it with three other programs: PILER-CR by Edgar and Myers [9], CRISPRDetect by Biswas et. al. [4] and CRT by Bland et. al. [6], against the authentic CRISPR arrays found in CRISPRdb [10]. PILER-CR detected more CRISPR arrays which began or ended at wrong positions. CRISPRDetect performed better but missed certain portions of CRISPR arrays. CRT was better than the other two tools, but it was not able to filter out tandem repeat sequences. Furthermore, the results also showed that this method was effective in excluding non authentic CRISPR arrays compared to other programs.

To further analyze the results, Wang and Liang [1] compared the length distribution of repeat sequences detected by all the four programs. CRT, CRISPRDetect, and CRF had similar distribution patterns, whereas PILER-CR had a different pattern supporting the earlier finding that PILER-CR detected wrong start and end positions.

2.1.2 Strengths and weaknesses

CRISPR random forest uses structural information of the repeats. Since the repeats are palindromic in nature, using structural information of the repeat sequences strengthens CRF's performance. Furthermore, it filters out tandem or adjacent repeats. This results in the repeat length distribution of the CRISPRs detected using CRF being very similar to the experimentally verified CRISPRs in CRISPRDb.

Besides these strengths, CRF has certain weaknesses. First, it relies only on the

similarity of repeats and does not consider certain CRISPR properties. Secondly, since the random forest models are inherently difficult to interpret, the CRF's detection model is not interpretable. Finally, CRF does not assign direction to the CRISPR arrays.

2.2 CRISPRDetect: A flexible algorithm to define CRISPR arrays

Biswas et. al. [4] claim that the existing CRISPR detection techniques do not utilize the recently discovered CRISPR loci features. In CRISPRDetect, they have proposed to leverage these features to improve the detection accuracy. CRISPRDetect detects CRISPR arrays by performing the following five steps:

1. Repeat detection to give putative CRISPRs
2. Removal of CRISPR-like tandem repeats
3. Refinement
4. Determination of direction and similarity to characterized repeat families
5. Quality scoring

In the first step, candidate CRISPR arrays are detected using repeat sequences. Two short similar sequences (repeat sequences) separated by a dissimilar sequence are identified. CRISPRDetect uses the default size of the repeat sequence (repeat_word_length) as 11. The length of the dissimilar sequence i.e. the spacer length, should be in the range (min_spacer_length, max_spacer_length), calculated as follows:

$$min\_spacer\_length = 30 - repeating\_word\_length$$
$$max\_spacer\_length = 125 + repeating\_word\_length$$

Experimentally, it has been verified that the spacer length is greater than 20nt and the repeat length is greater than 23nt [4]. The range equations ensure that, given the

repeat sequence length, the spacer sequence length is appropriately being considered and also reduces the user input to just the repeat sequence length.

The genomic regions containing the above putative CRISPRs are, in the second step, divided into segments, beginning with the repeated sequence. Each of the segments is then aligned using ClustalW [11]. The initial repeat length is increased if the alignment is similar. If spacers between the repeats of these putative CRISPRs are found to have <5 unaligned columns, it is marked as a tandem repeat and is discarded.

The third step, refinement, is an important step and is further divided into eight subroutines as follows:

(a) Extending the repeat end: Mutations at the end of the repeats may result in a repeat being included in the spacer. CRISPRDetect progressively extends the repeat on both sides, comparing the bases from adjacent columns and maintaining a 'minimum column identity' of 75%. 'Alternate column identity' is permitted for one column; by default it is 50% for arrays with less than 7 repeats and is 40% for longer arrays.

(b) Selecting the representative repeats: The most common repeat is being considered as the 'representative' repeat and the second most common repeat is the 'alternate' repeat.

(c) Extend the array: Each candidate CRISPR array is extended by progressively checking its flanking region within the distance of its representative repeat length + 1.33 times the median length of its spacers. The checking is done using the Smith-Waterman algorithm and it is extended if the 'minimum repeat identity' (default is >=67%) is met.

(d) Refine the repeats: The predicted repeats may contain bases at the end that correctly belong to the spacers. CRISPRDetect corrects the spacer/repeat boundary by comparing the repeats with a library of known repeats, with known motifs found in the end of repeats and with repeat end region degeneracy.

(e) Trim the array: Degenerated repeats can be falsely included after dynamically extending the CRISPRs. In this step, the terminal repeats that poorly match the representative repeats are removed.

(f) Correct gaps at repeat ends: CRISPRDetect uses matching bases from initially predicted spacers to refine repeat ends.

(g) Representation of insertions in a small number of repeats of an array.

(h) Identify mutated repeats initially predicted to be long spacers: Substantial portion of a repeat and/or repeat-spacer junction may be deleted. In such cases, the 'minimum percentage identity' is not retained and step (a) could erroneously consider it as a spacer, making it an unusually long spacer. In this step, spacers longer than the median spacer length and having high percentage identity are labelled and then present in the output.

2.2.1 Performance Analysis

CRISPRDetect algorithm was run on 2,806 complete bacterial and archaeal genomes. A total of 3,901 CRISPR arrays were found out of which 3,870 had good scores and were labeled as 'good' arrays. In the 'good' arrays, 12% were not identical to their representative repeats, with 50 repeats less than 70% identical and 399 less than 80% identical. About half of the 'good' arrays were corrected for direction and 160 were flagged as likely direct repeats.

The same genomes were also run for prediction on three other programs: PILER-

CR [9], CRT [6] and CRISPRFinder [3]. CRISPRDetect has high concordance with PILER-CR and CRT. It predicted 345 additional CRISPR arrays compared to all the others. Only 10 arrays that were predicted by all the others were not predicted by CRISPRDetect.

2.2.2 Strengths and weaknesses

CRISPRDetect uses recently discovered biological features of CRISPR arrays. Furthermore, since it uses traditional computer science algorithms, it is easily interpretable. Biswas et. al. [4] have also incorporated a direction determination algorithm in the detection pipeline. CRISPRDetect is able to assign directions to the detected CRISPRs. Another important strength of CRISPRDetect is, it can identify mutations at the 3' end and thus can detect short or degenerate CRISPRs. The only weakness of CRISPRDetect is it misses certain portions of CRISPR arrays, that is because it does not consider structural information of repeats.

2.3 CRISPRFinder: A web tool identify CRISPRs

Grissa et. al. [3] released one of the earliest tools for finding CRISPR arrays in a genome sequence called CRISPRFinder. In the paper, they have compared their tool with certain generic pattern matching tools that were not CRISPR specific but were widely in use. They claim that the generic tools are not good at CRISPR detection because they miss out on certain CRISPR properties and require certain pre-processing and post processing steps after the detection is complete to filter out valid CRISPRs. Also, CRISPRFinder provided a very intuitive web-based interface that none of the tools provided back when the tool was released. Like CRISPRDetect, Grissa et. al.[1] have mostly relied on CRISPR properties to improve CRISPR detection. They use properties like repeat and spacer length, the similarity between repeats and spacers and CRISPR evolution rules. Moreover, they use a linear time suffix tree-based algorithm

to find patterns. The exact steps that CRISPRFinder uses to detect CRISPR arrays are as follows:

1. Browsing the maximal repeats
    (a) Repeats of 23-55 bp interspaced by 25-60 bp sequence
2. Select the DR consensus
    (a) Number of occurrences of the DR sequence in the whole genome
    (b) Privilege internal mismatches between DRs rather than mismatch in first or last nucleotide
3. Defining candidate CRISPRs by checking if they fit the CRISPR definition
4. Eliminating residual tandem repeats

2.3.1 Performance Analysis

CRISPRFinder is efficient at finding most of the CRISPR-like sequences in a genome, specifically short CRISPRs and the ones with a degenerate repeat. CRISPRFinder can also detect direct repeat boundaries accurately to a base pair resolution. Furthermore, the interface of the CRISPRFinder tool is easy to understand and it has an option to easily extract and blast spacers against different databases. Grissa et. al. [3] compared their tool with other available tools, to verify if CRISPRFinder could efficiently recover all the CRISPRs from a genome. The data was generally in good agreement with the other tools. There were a few discrepancies though, that were found in the DR boundaries' identification (CRISPRFinder was more accurate) and the number of motifs (spacer+repeat) found. Truncated DRs and short CRISPRs are sometimes neglected by other tools.

2.3.2 Strengths and weaknesses

CRISPRFinder, being one of the oldest CRISPR array detection tools, is still very efficient in finding CRISPR arrays. The algorithm is very interpretable and easy to understand. It relies mostly on CRISPR properties and uses suffix tree-based pattern algorithm which makes it linear in time and space complexities. Since CRISPRFinder considers CRISPR evolution rules during the detection, it is usually good at detecting any CRISPR-like structures as a candidate CRISPR and can filter any invalid candidates. Also, like CRF, CRISPRFinder filters out tandem or adjacent repeats. But there are few weaknesses with CRISPRFinder. The most important one is, like CRISPRDetect, it does not consider the structural properties of the repeats and thus misses certain portions of CRISPR arrays. Secondly, unlike CRISPRDetect, CRISPRFinder does not assign direction to the detected CRISPRs. Grissa et. al. [3] have compared their tools with generic pattern matching tools and as expected, CRISPRFinder performs very well compared to others. Generic pattern matching tools ignore CRISPRs containing less than three spacers. The users need to define specific pattern before using those tools and the output of these tools contains large quantities of noised data which needs further filtering.

2.4 CRISPR Recognition Tool (CRT): a tool for detection of CRISPRs

Bland et. al. [6] developed a simple but very effective method for detecting CRISPR arrays. It is based on finding exact k-mer repeats that are separated by a similar distance and then extending each repeat, maintaining certain threshold, till an approximate repeat length [6]. Their method involves three steps:

1. Find short exact repeats of length k: In the first step, the algorithm looks out for an exact matching repeat sequence of length k which is separated by some distance. This sequence of length k, is called the seed region. This value k is one

of the input parameters of the algorithm. Having a large k, means less chances of finding an exact match, whereas having a very small k will make the sequence almost always an exact match. The default value of k used by the algorithm is 8, which is not too large nor too small.

2. Extend the k-mer exact repeats: Once the k-mer repeats are recoginized, each of the k-mer is extended from the 3' and the 5' ends. The base occurrence percentage of each of the bases to the 3' and the 5' ends of all the k-mer repeats is computed. If a certain base has occurred more than a set threshold (input parameter) then the k-mer is extended to a k+1-mer. The default extension threshold is 75% on both 3' and 5' ends.

3. Filter repeats that do not meet CRISPR specific requirements: Finally, each of candidate CRISPRs are validated for CRISPR specific requirements and the invalid CRSIPR arrays are filtered out.

2.4.1 Performance Analysis

Like CRISPRFinder, CRT was one of the earliest CRISPR array detection tools. The authors, have compared its performance to Patscan, which was one of the pattern detection tools available then. The authors claim that CRT is very fast and memory efficient. CRT, since it just uses string-matching algorithm, has a linear running time. The algorithm is also linear in space, since no major structures are required [6]. The authors also claim that CRT has a very high recall i.e. ratio of the number instances of correctly identified to the total number of instances that are CRISPRs [6].

2.4.2 Strengths and Weaknesses

In CRT one can explain what exactly is happening at each step, since it mostly relies on traditional string matching techniques. Furthermore, as the authors claim, it has a very high recall and a descent precision, which means CRT can be followed

by robust filtering steps to give a very good CRISPR detection algorithm. Also, it is fast compared to pilerCR and Patscan. But besides these strengths, there are a few weaknesses. CRT does not consider the structural properties of the repeats. It also, does not assign direction to the detected CRISPR arrays.

In the next chapter, the Long short-term memory (LSTM) model and why it can be effective in detecting CRISPR sequences is explained.

CHAPTER 3

Long-Short Term Memory models

3.1 What is a Long-Short Term Memory model?

Long-Short Term Memory (LSTM) models are a type of Recurrent Neural Network (RNN) models. These models are good at recognizing patterns in sequential data such as text since they take into account the time and order of the data while processing it. A simple RNN model looks like the left-hand side of Figure 3 [2]. An input x_t, is given as input to the Neural Network A, to get an output ht, the t here signifies the time. For the ease of understanding, the right-hand side of Figure 3 shows the same model in an unwrapped fashion. The input x_0,x_1,x_2,...,x_t is each character of the input sequence whereas the output h_0,h_1,h_2,...,h_t is the output for each input sequence. The horizontal arrows from one A to other signifies that the output of one input time stamp is forwarded as the next input along with x_t at that timestamp.

Thus, RNNs are good while working on the sequential data. RNNs have one problem though, they are bad at learning long-term dependencies. For example [2], if we want the RNN model to predict the next word in the sentence 'Trees are _____', it can easily predict 'green'. But if the sentence is long, like 'I live in France... I speak fluent _____', here, for the model to answer 'French' it needs to remember the word France that had occurred long back. Theoretically, RNNs should be able to remember

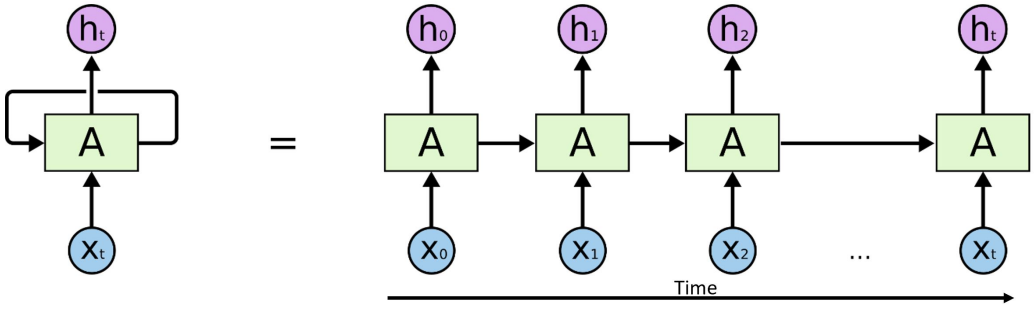


Figure 3: Recurrent Neural Network [2]

encountered words and answer correctly, but practically, they tend to forget long term context. This is where the special type of RNN model, a LSTM model, comes into play. LSTMs are designed to tackle the long-term dependency problem [2].

3.2 LSTMs on DNA Sequences

In our case, for the DNA sequence, the input x_0,x_1,x_2,...,x_t will be the sequence of nucleotides in the DNA sequence and there will only be one output at the end of the complete sequence. That output will tell if the given sequence is a valid repeat sequence or not.

H. Hassanzadeh and M. Wang [12], report that LSTM is highly effective in finding patterns in the DNA which is simply a long sequence of 4 alphabets (nucleotides). The power of LSTMs can be leveraged to find the CRISPR sequences in the DNA. An LSTM model can be trained on the repeat sequences in the valid CRISPR arrays present in CRISPRDb. This way the model could learn the patterns found in the valid repeat sequences. Later, building upon the existing tools, CRT with very lenient default parameters, can be used to extract candidate CRISPRS. On the candidate CRISPRs, the trained LSTM model can be used to filter out valid CRISPRs. These valid CRISPRs can then further be checked for CRISPR properties as done with CRISPRDetect and CRISPRFinder.

The next chapter describes the proposed CRISPR Detection method (CRISPRL-stm) along with an architecture diagram and a simple example to clearly understand the detection process.

CHAPTER 4

CRISPR array detection using CRISPRLstm

4.1 Dataset description

Three datasets were used in the project: the self-curated dataset, the publicly available CRF's [1] dataset and the dataset made available by Godde and Bickerton in [8]. The self-curated dataset is partially acquired from CRISPRDb. CRISPRDb is a publicly available dataset with known CRISPR sequences. It is regularly updated and can be queried to fetch CRISPR sequences. At the time of writing this report, 3254 CRISPR sequences were present in CRISPRDb. From each of those CRISPR sequences, a consensus repeat sequence is provided in CRISPRDb and can be downloaded. Thus, 3254 repeat sequences were downloaded. These repeat sequences were labelled as valid repeats because they belonged to valid CRISPR arrays. An equal number of DNA sequences with the same length distribution were randomly created. These sequences were labelled as invalid repeat sequences, since they do not belong to any CRISPR array. The second dataset, used and made publicly available by CRF [1], contains 11,407 valid repeat sequences and 12,000 invalid repeat sequences. Tables 2 and 3 show the details of the first and second datasets respectively.

Table 2: Curated Dataset's Statistics

Sequence	Count
Valid repeats	3,254
Invalid repeats	3,254
Total	7,508

Table 3: CRF [1] Dataset's Statistics

Sequence	Count
Valid repeats	11,407
Invalid repeats	12,000
Total	23,407

Lastly, the Godde and Bickerton's dataset, consists of around 400 repeat sequences found in around 160 organisms, along with the name of the organism, number of times that repeat has appeared in the organism and the size of the repeat sequence. First two datasets were used to train and assess the performance of the LSTM model,

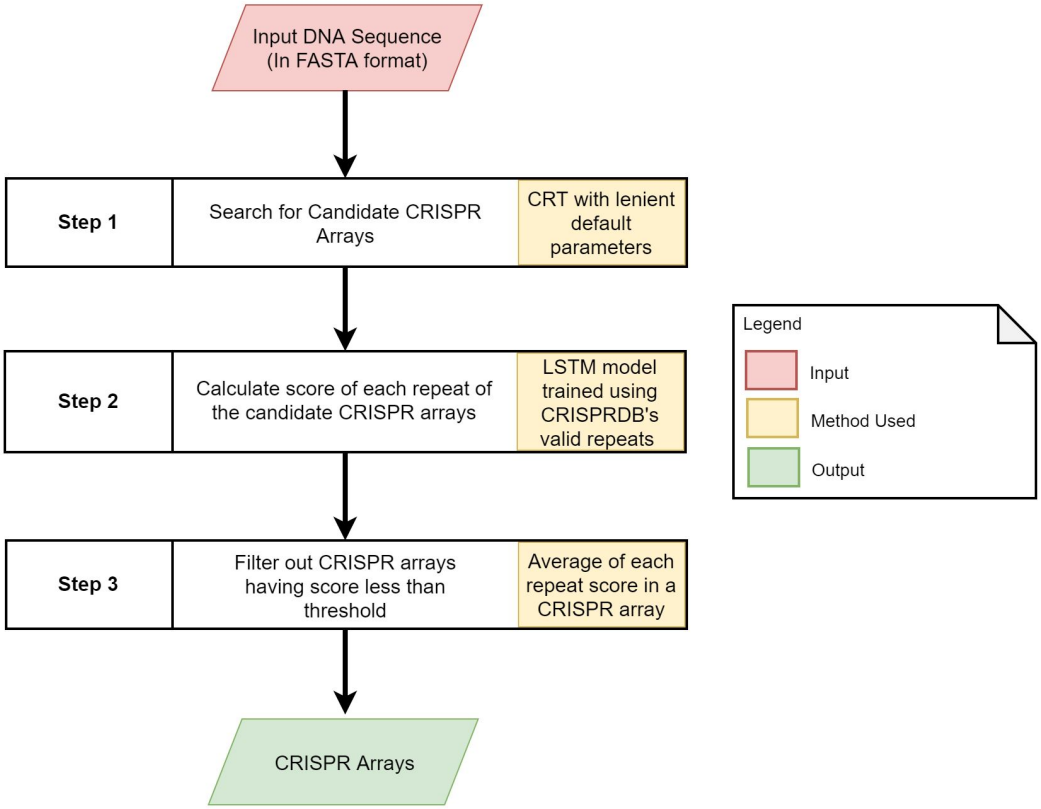


Figure 4: Implementation pipeline of CRISPRLstm.

where as the third datase was used to validate the complete CRISPRLstm detection pipeline.

4.2 CRISPR array detection pipeline

The proposed method, CRISPRLstm, is inspired from CRF [1]. Instead of using the random forest classifier as in CRF [1], CRISPRLstm uses Long-Short Term Memory model as mentioned in Chapter 3. CRISPRLstm is divided in three main parts, as shown in Figure 4.

1. `Finding candidate CRISPR Arrays`

   CRISPR Recoginition Tool is a widely used tool for finding CRISPR Arrays [1]. The authors of CRT [6] claim that it has high recall but not so high precision.

High recall means that it does not miss any of the valid CRISPR array in a given sequence, on the other hand, low precision means the CRISPR arrays detected as valid by CRT are often false positives. This means, the arrays detected by CRT need further filtering to increase the precision and in turn the accuracy of detection. Having a high recall, makes CRT a perfect candidate to find candidate CRISPR arrays. Thus, in the first step, CRT uses a lenient default parameters to get a list of candidate CRISPR arrays. Making the default parameters lenient, ensures that none of the probable CRISPR arrays are missed.

As discussed in Chapter 2, CRT accepts the seed region length (k) and the base occurrence percent threshold as the input parameters. As in CRF, both these parameters are modified to make the CRISPR detection lenient. The length of the seed region (k) is changed from 8 nt to 5 nt. The base occurrence percent threshold, which is used to extend the seed region, is reduced to 0.6 for the 3' end, whereas on the 5' end it is kept the same (0.75). This is because the 3' end of the repeats appear to have more variations than the 5' end [1, 13].

2. `Scoring repeat sequence of each candidate CRISPR Array`

   In this step, each of the repeat sequences of the candidate CRISPR arrays, found in the first step, are extracted and scored for being a valid repeat. The scoring is done using a LSTM model with a sigmoid activation function as the last layer. The model, thus, gives out a score between 0 and 1. Higher the score, the higher the likelihood of the repeat being valid. The architecture of the LSTM model is shown in Figure 5. Keras library [14] was used to build and train the LSTM model.

   The first input layer in the model accepts a vector of length 45. The vector should only consist of positive integers. So, as a prepossessing step, all the input

Layer (type)	Output Shape	Param #
inputs (InputLayer)	(None, 45)	0
embedding_1 (Embedding)	(None, 45, 50)	500
lstm_1 (LSTM)	(None, 32)	10624
FC1 (Dense)	(None, 128)	4224
activation_1 (Activation)	(None, 128)	0
dropout_1 (Dropout)	(None, 128)	0
out_layer (Dense)	(None, 1)	129
activation_2 (Activation)	(None, 1)	0

Figure 5: Architecture of the LSTM model used for scoring.

repeat sequences were first tokenized i.e. each nucleotide was assigned an integer value and then the whole input sequence was converted to an integer vector. For example, consider 'AAGTCGT' it is converted to a vector [1123423]. If the length of the vector is greater than 45, only the first 45 integers were considered and the rest were discarded. On the other hand, if the length of the vector is less than 45, then the vector was left-padded with 0s to make it 45 integers long. The embedding layer turns positive integers (indexes) into dense vectors of fixed size. Then LSTM outputs a vector of length 32 (see Chapter 3 for details). The LSTM layer is connected to a fully connected layer followed by an activation layer which gives an output vector of length 128. An activation layer applies an activation function to the output of it's initial layer. An activation function is a non-linear function which defines the output of a specific index for a given input at that index. The activation layer is then followed by a dropout layer.

A dropout layer is used to avoid over-fitting during training the model. The output vector is then passed to a fully connected layer with an activation layer which outputs a single value between 0 and 1. For training the model, curated data-set defined in Table 2 was used.

3. Filtering candidate CRISPR Arrays

   For each repeat sequence of a candidate CRISPR array, the LSTM model, in step 2, gives a score from 0 to 1. In this step, the score of a CRISPR array is calculated as the average score of each of its repeats. A threshold of 0.5 is then used to filter out invalid CRISPR candidates i.e. if a candidate CRISPR has a score >0.5 it is considered as a valid CRISPR array otherwise it is filtered out. The threshold is an input parameter and can be changed if required.

4.3 An Example

Consider a sample genomic sequence:

$NNR'NR'NR'NNRSRSRSRSRNNNNNNNNNNNNNNNNRSRSRSRNN$

where,

N : *Any nucleotide base* S : *Spacer sequence*

R : *Repeat sequence* R' : *Invalid repeat sequence*

In the sequence, let's assume, there are 2 valid CRISPR arrays one in the center of the sequence with 5 repeat sequences and one at the right end with 4 repeat sequences, and 1 invalid CRISPR array just at the beginning with 3 invalid repeats.

Now, assume the sequence is given as input to the proposed pipeline shown in Figure 4. In the first step, CRT with lenient default parameter; will output 3 CRISPR candidates, each with 3, 4 and 5 repeat sequences. In the second step, each of 12 candidate repeats will be passed through the trained LSTM model which will give a

score, between 0 and 1, to each of the repeats. Let's assume the scores to be 0.31, 0.42, 0.56, 0.83, 0,77, 0.90, 0.92, 0.69, 0.75, 0.79, 0.97, 0.87 for each of the repeat sequences starting from the 5' end.

In the final step, for each candidate CRISPR array, an average of all its repeat's score will be calculated. If that score is less than the empirically determined threshold of 0.7, the array will be considered invalid. In our case, the scores of the candidate CRISPR arrays will be 0.43, 0.79 and 0.81, respectively, starting from the 5' end. Since, the score of the first candidate array is less than 0.5, it will be filtered out and the rest of the two will be considered as valid. Thus, the output of the pipeline will be the two valid CRISPR arrays with 4 and 5 repeat sequences each.

In the next chapter, the experiments that were performed using CRISPRLstm and their results are discussed.

CHAPTER 5

Experiments and Results

5.1 Comparison with CRF

As mentioned in Chapter 4, CRISPRLstm is inspired from CRF, so it was important to compare its results with CRF. The authors of CRF claim that the random forest classifier achieved an accuracy of 94.42% and sensitivity of 93.99% [1]. They have used the dataset defined by Table 3 to train a random forest classifier. It is worth noting that the results achieved by CRF were on a 80:20 training:testing split of the dataset and not a cross validation results. So, to compare the CRISPRLstm's classifier with the random forest classifier, two sets of 10-fold cross validation tests were performed, one using the curated dataset mentioned in Table 2 and the other with the CRF's dataset mentioned in Table 3.

Table 4: Fold-wise classification accuracy of random forest and LSTM on the CRF's dataset

Fold	1	2	3	4	5	6	7	8	9	10
CRF	0.937	0.963	0.952	0.965	0.938	0.946	0.961	0.946	0.949	0.963
LSTM	0.930	0.927	0.926	0.924	0.927	0.931	0.931	0.933	0.927	0.927

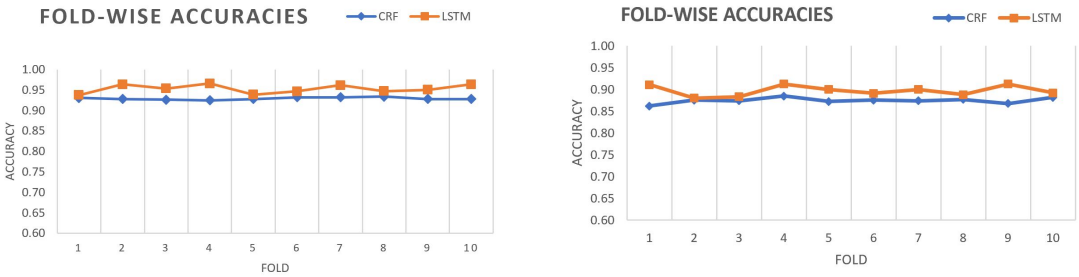
Table 5: Fold-wise classification accuracy of random forest and LSTM on the self-curated dataset

Fold	1	2	3	4	5	6	7	8	9	10
CRF	0.862	0.876	0.874	0.885	0.873	0.876	0.874	0.877	0.868	0.882
LSTM	0.911	0.880	0.883	0.912	0.900	0.891	0.900	0.888	0.912	0.892

Table 6: Performance comparison between random forest and LSTM classifier

	CRF dataset			Self-curated dataset		
	AUC	Sensitivity	Specificity	AUC	Sensitivity	Specificity
random forest classifier	0.925	0.923	0.933	0.874	0.881	0.867
LSTM classifier	0.989	0.933	0.979	0.969	0.946	0.845

Tables 4 and 5 show the fold-wise classification accuracy of both the random forest and the CRISPRLstm's LSTM model on the CRF's dataset (Table 3) and the



(a) On CRF's dataset

(b) On self-curated dataset

Figure 6: Fold-wise classification accuracy graphs

curated dataset (Table 2) respectively. Besides accuracy, average sensitivity and specificity was computed for each of the dataset. Table 6 shows the values of each of the performance metric for both the datasets. For all the metrics and both the datasets, except for the specificity of self-curated dataset, CRISPRLstm's LSTM outperforms the random forest classifier. Figure 6a and 6b are the fold-wise line charts of the accuracies. It can clearly be seen that the LSTM model performs better than the random forest classifier, with an approximate 3-4% increase on an average, for each fold. To ensure that this does not happen by chance, 100 10-fold cross validations were performed using the CRF's dataset on both the LSTM and the random forest classifier. For each fold, accuracy, sensitivity and specificity metrics were computed. Thus, there were 1000 values for each metrics. This data was then used to perform a t-test. Scipy library's [15] stat package was used to compute the tstatistic for each of the metrics. Table 7 shows the results of the test. t-value measures the size of the difference relative to the variation in the two sets of data, whereas, the p-value is the probability of finding extreme results between the two sets of data [16]. For all the metrics, the t-value is large and the p-value is either very small or 0. This suggests that the results of cross validation are indeed different and did not occur by chance.

Table 7: t-test result of 100 10-fold cross validation's on [1]'s dataset

	t-value	p-value
Sensitivity	7.8132	8.93E-15
Specificity	113.8492	0
Accuracy	66.5159	0

Table 8: No. of CRISPR arrays detected by different programs on 8 genomic sequences

Accession No.	Organism Name	CRISPR Lstm	CRT	PilerCR	CRISPR Detect	CRISPR Finder
CP006907.1	Clostridium botulinum CDC_297	3	4	2	3	4
NC_010628.1	Nostoc punctiforme PCC 73102	6	7	7	9	7
NC_019771.1	Anabaena cylindrica PCC 7122	9	9	12	12	8
NC_019682.1	Calothrix sp. PCC 7507	10	11	15	13	11
NC_003272.1	Nostoc sp. PCC 7120	11	12	11	12	11
CP009149.1	Methanocaldococcus bathoardescensJH146	11	11	15	12	12
NC_019676.1	Nostoc sp. PCC 7107	12	14	14	15	15
NC_019678.1	Rivularia sp. PCC 7116	13	14	13	18	13

5.2 rCRISPR metric on 8 known genome sequences

The performance of CRISPRLstm was compared with four other tools, namely, CRT [6], PilerCR [9], CRISPRDetect [4] and [3]. Eight complete genome sequences were used for the analysis. Table 8 shows the number of CRISPR arrays detected by each of the programs on every sequence. The results of CRISPRLstm are generally in good agreement with the results from other tools. But, these measurement do not accurately show the detection accuracy of each of the tools. A novel performance metric, rCRISPR, defined by [5] was used for this purpose. rDR is defined as the ratio of the number of DR copies detected by a given program with the number of DR copies in known the CRISPR array. rCRISPR is the average rDR of all the CRISPR arrays in a given genome. It is possible that the rCRISPR value can be greater than 1,

since a program can detect more DRs than know number of DRs in a given sequence.

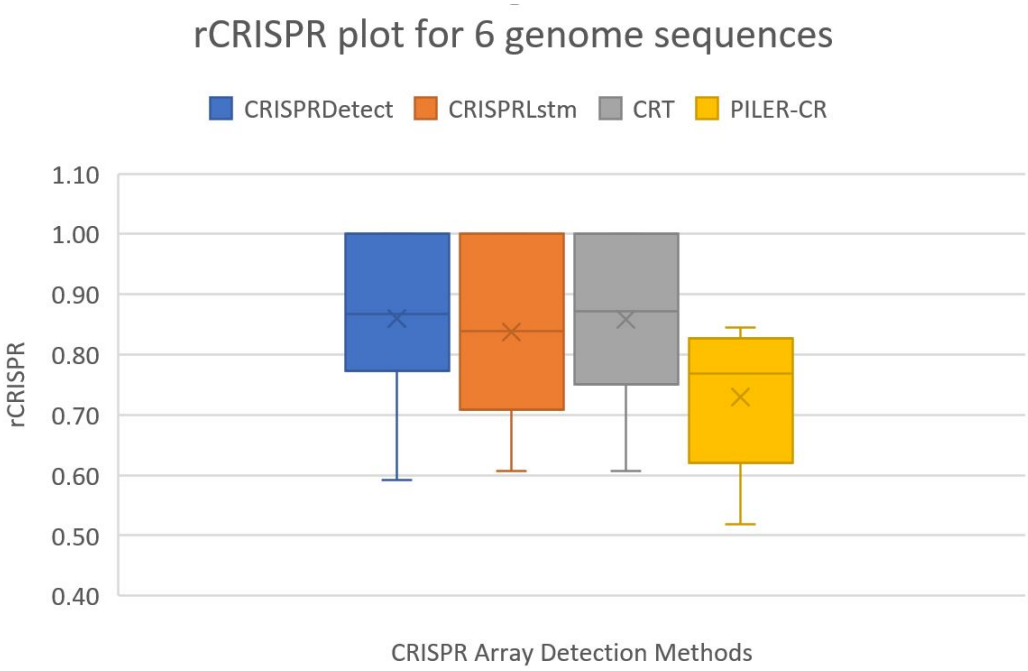


Figure 7: Performance measurement of CRISPR Detection programs

For all the programs and genome sequence in Table 8, rCRISPR value was computed. Figure 7 shows the box and whisker plot of their rCRISPR values. As it can be seen, none of the tools has a mean greater than 1, suggesting that for those 6 genome sequences none of the programs detected more DRs than the known number of DRs. CRISPRDetect, CRISPRLstm and CRT has a mean close to one, inferring that they usually detect equal to or slightly less than the actual repeat sequences, although CRISPRLstm does not have a tight distribution. PilerCR, on the other hand, has a mean well below 1, suggesting that it detects fewer DRs than the known ones.

5.3 Comparison with CRT, PILER-CR and CRISPRDetect

Stand-alone versions of CRT, PILER-CR and CRISPRDetect with their default parameters were executed on 81 fasta files of the organisms from the Godde &

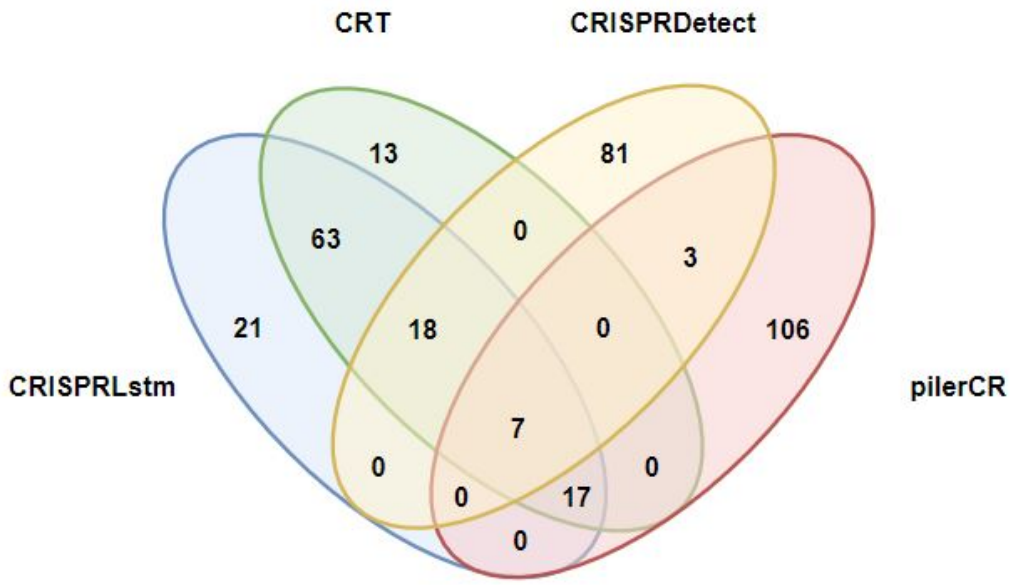


Figure 8: Visualizing the number of CRISPR arrays detected by CRT, PILER-CR, CRISPRDetect and CRISPRLstm on Godde and Bickerton's dataset

Bickerton's dataset [8]. Figure 8 shows the number of CRISPR arrays detected by each of the programs. Overall, 329 CRISPR arrays were detected of which 7 CRISPR arrays were detected by all the tools. CRT and CRISPRLstm have most of the overlap since they share 105 CRISPR arrays between them. This was expected since CRISPRLstm has CRT with lenient parameters as an intermediate step. CRISPRDetect and PILER-CR shared 10 CRISPR arrays between them. Besides the 7 common CRISPR arrays, CRISPRLstm and CRT shared 0 CRISPR arrays with CRISPRDetect and PILER-CR. CRISPRLstm, CRT, CRISPRDetect and PILER-CR detected 21, 13, 81 and 106 CRISPR arrays, respectively, that none of the other tools detected. Furthermore, each of these CRISPR arrays were compared for validity with the results of Godde & Bickerton's dataset [8]. None of the 21 CRISPR arrays detected only by CRISPRLstm were valid, 1 out of 13 CRISPR arrays detected only by CRT were valid, 19 out of 81 CRISPR arrays detected only by CRISPRDetect

were valid and 38 out of 106 CRISPR arrays detected only by pilerCr were valid. CRISPRLstm's performance on this dataset was not as good compared to the other programs.

Besides this visualization, rCRISPR values were computed using the detected CRISPR arrays and the known arrays available in the dataset. Figure 9 shows the box and whisker of their rCRISPR values. pilerCr as the earlier observations, in Figure 7, detects less of the known direct repeats. CRISPRDetect, has slightly higher rCRISPR mean value. Performance of CRT and CRISPRLstm in this case were very similar.

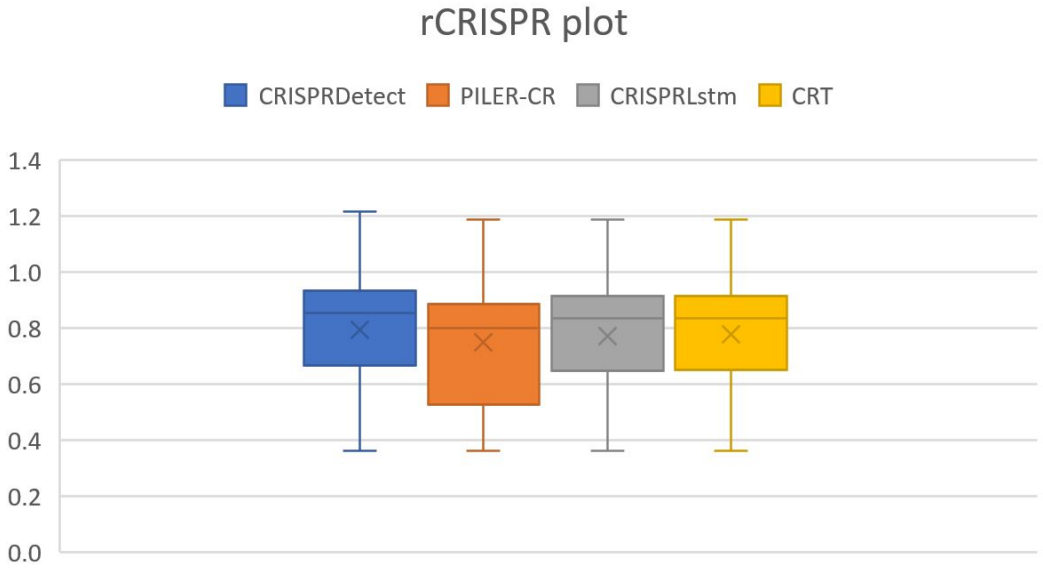
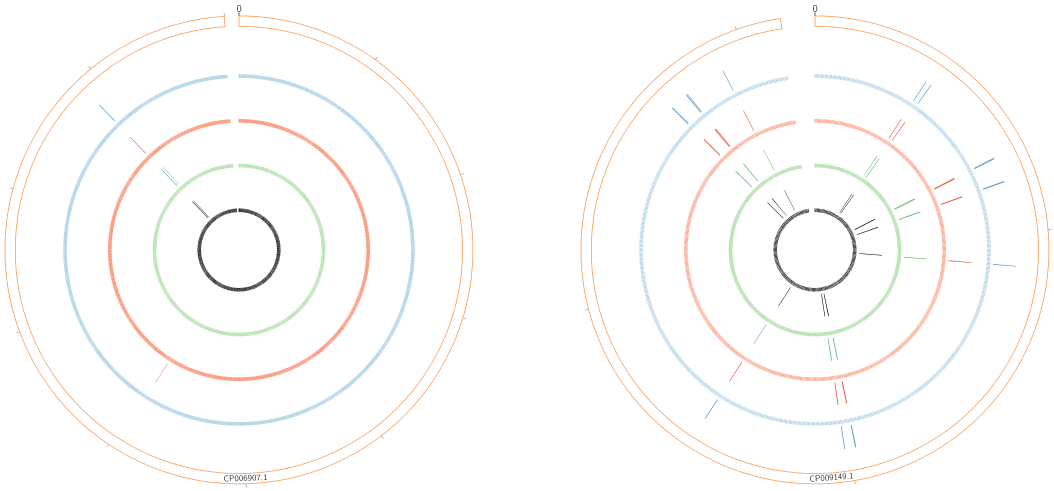


Figure 9: rCRISPR plot on Godde & Bickerton's dataset

5.4 Visualizing CRISPR array distribution detected by various programs

Two complete genomes, *Clostridum botulinum CDC297* bacterium and *Methanocaldococcus sp. JH146* archaea, were used to compare the performance of four CRISPR detection programs: PILER-CR, CRISPRDetect, CRT and the CRISPRLstm. Figure 10 shows the circos visualizations with the position of each of the detected CRISPR array by these programs. In general, the results of CRISPRLstm were in good agreement with all the progrmas. For the bacterium, 3 CRISPR arrays



(a) *Clostridum botulinum CDC297* (b) *Methanocaldococcus sp. JH146*

Figure 10: Circos plots of detected CRISPR arrays in two genomes. The outermost circle represents the genome sequence. The gap on the upper-side of each circle is the start and end positions of the sequence. Each solid circle represents CRISPR arrays detected by PILER-CR (blue), CRISPRDetect (red), CRT (green) and CRISPRLstm (black) from the innermost to the outermost circle respectively.

were found by CRISPRLstm. PILER-CR missed 1 CRISPR array. CIRPSRDetect was the only package to find a CRISPR array at a completely different position which none other programs found. CRT found 1 extra CRISPR array than CRISPRLstm. For the archaea, 11 CRISPR arrays were found by CRISPRLstm. PILER-CR found 16. It basically split long CRISPR arrays into smaller ones because of undetected internal repeats. CRISPRDetect detected 1 extra CRISPR. CRT was in exact agreement with CRISPRLstm, detecting 11 CRISPRs.

5.5 Specific Examples

Grissa et al. [3] has specific examples of genome sequences with short and long CRISPR arrays such as *Shigella sp.* and *Pseudomonas aerginosa* respectively. Godde and colleagues [8] did not find any CRISPR arrays [3]. CRISPRLstm found one questionable CRISPR array in *Shigella sp.* and two CRISPR arrays in *Pseudomonas*

aerginosa. 6 spacers out of 36 spacers in *Pseudomonas aerginosa*'s two CRISPR arrays correspond to a bacteriophage sequence. This could signify that those two CRISPR arrays are indeed valid. Same was reported by [3].

CRISPRDetect [4] states that mutations at the center of an internal repeat sequence may cause the program to split a CRISPR array in two or more short CRISPR sequences. PilerCR for example, splits a CRISPR array in *Carboxydothermus hydrogenoformanas* in two short CRISPRS (of 12 and 68 spacers). CRISPRLstm corrects this error, resulting in a long CRISPR array with 83 spacers (identifying 3 extra spacers), as reported by CRISPRDetect [4].

5.6 User Interface

The user interface [17] of the proposed pipeline has a lot of features besides just searching for a CRISPR array. It allows to create the web sequence logo of the repeat sequences, blast any of the repeat or spacer sequence, create secondary structure of the repeat or spacer sequence and show the structure of the CRISPR array.

Figure 11 shows the result page. It shows the organism name at the top, followed by 4 statistics, namely, the number of bases in the input sequence, number of valid CRISPR arrays found by the pipeline, number of questionable CRISPR arrays i.e. arrays with scores below the threshold at step 3 from the Figure 4 and the time taken to find the CRISPR arrays (this is just the processing time and excludes any network delay). These statics are then followed by two dropdown sections, one dropdown shows valid CRISPR arrays and the other shows the questionable ones. By default, the valid CRISPR array's dropdown is expanded. Each dropdown, when expanded, has a list of all the CRISPR arrays visualized in detail.

Figures 12 and 13 show the repeat web logo and the CRISPR Structure, respectively, in a pop-up. Both pop-ups are triggered by clicking on the buttons at the

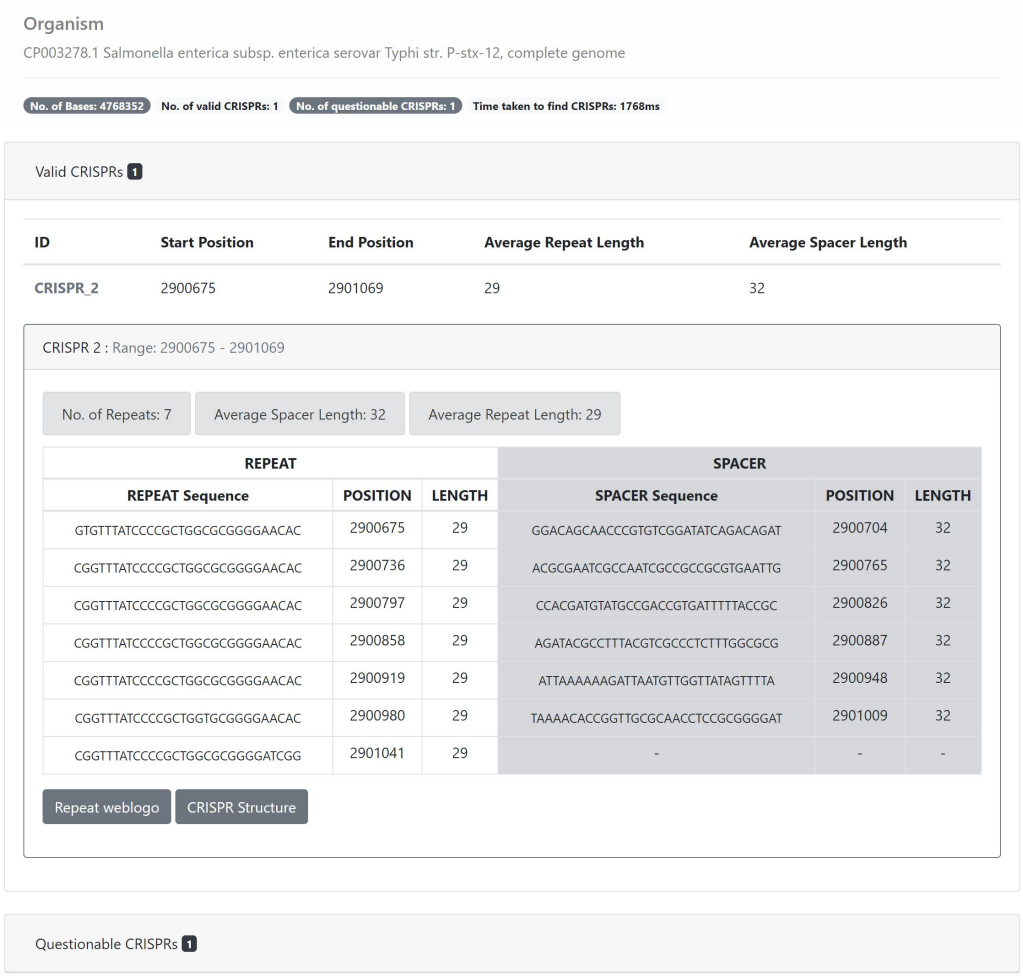


Figure 11: Results page

bottom of the CRISPR table on the result page.

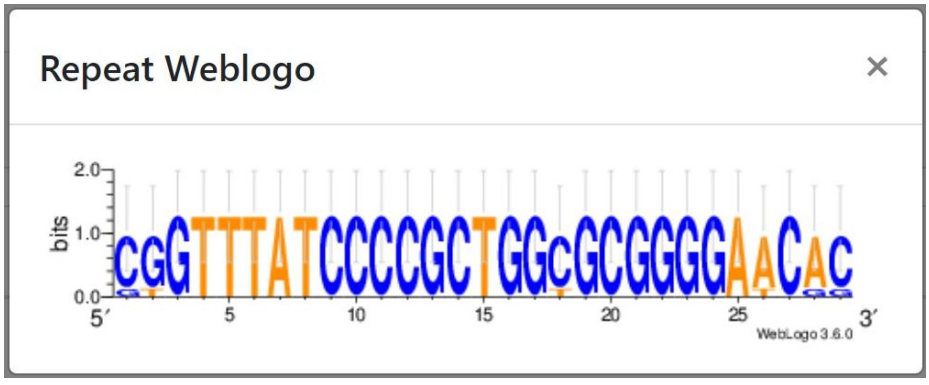


Figure 12: Repeat weblogo popup

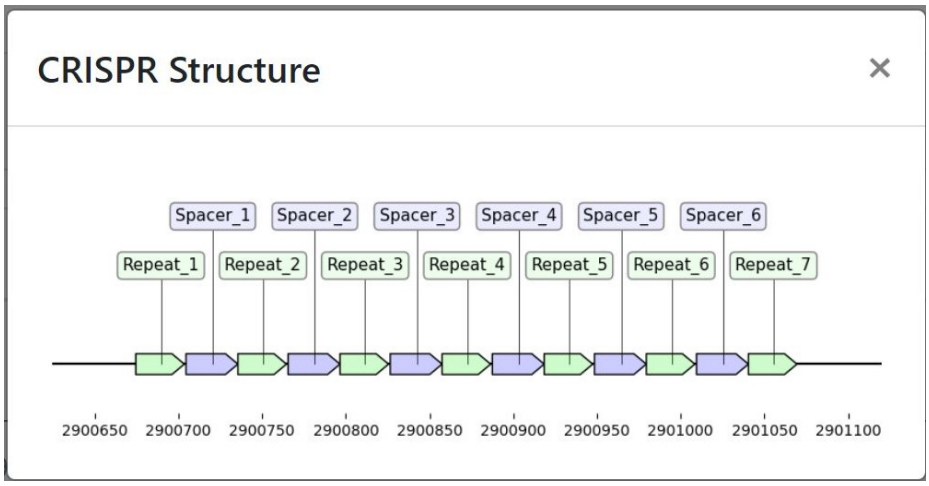


Figure 13: CRISPR Structure popup

On clicking any of the repeat or spacer sequences on the results page opens a pop-up as shown in Figure 14. It has the secondary structure of the sequence. There is a button at the bottom of the pop-up that allows to blast the sequence at NCBI. The blast page is opened in a new tab.

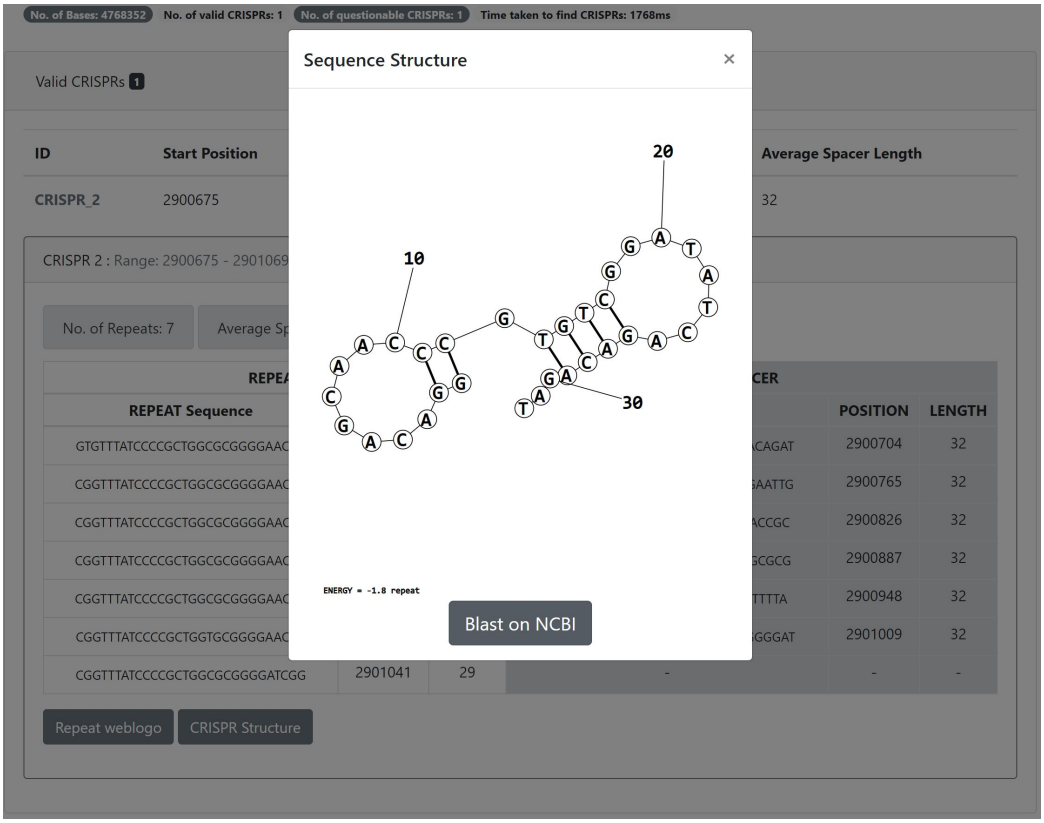


Figure 14: Secondary structure of the clicked sequence with an option to BLAST the sequence

CHAPTER 6

Conclusion

This work presents a new method to detect CRISPR arrays in a genome sequence. It leverages the power of artificial intelligence, specifically, long-short term memory network model to significantly improve the power of detecting valid repeat sequences. All packages that can be used to find CRISPR arrays have advantages and disadvantages. The proposed program, CRISPRLstm, performs at par compared to these tools. It's results are, in general, in good agreement with other packages. Using multiple tools to get the best results is recommended since the results compliment each other. The CRISPRLstm's LSTM classifier outperforms the random forest classifier. The user interface of CRISPRLstm is intuitive, highly interactive and offers varied functionality such as repeat sequence logo creation, 2D structure creation and blasting spacer/repeat sequences. Future work to the CRISPRLstm's pipeline would be to add direction to the detected CRISPR arrays and filtering out tandem repeats. The trained CRISPRLstm's LSTM model could be incorporated in any of the existing CRISPR Detection pipeline to further improve the precision value of the results, CRISPRCasFinder [18], an update version of CRISPRFinder [3], is a good fit for the same with its high recall. CRISPRLstm is available online at http://35.236.94.239/.

LIST OF REFERENCES

[1] K. Wang and C. Liang, ''Crf: detection of crispr arrays using random forest,'' *PeerJ*, vol. 5, p. e3219, 2017.

[2] O. Christopher, ''Understanding lstm networks ,'' August 27, 2015. [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[3] I. Grissa, G. Vergnaud, and C. Pourcel, ''Crisprfinder: a web tool to identify clustered regularly interspaced short palindromic repeats,'' *Nucleic Acids Research*, vol. 35, no. Web Server issue, p. W57, Jul 1, 2007.

[4] A. Biswas, R. H. J. Staals, S. E. Morales, P. C. Fineran, and C. M. Brown, ''Crisprdetect: A flexible algorithm to define crispr arrays,'' *BMC genomics*, vol. 17, no. 344, p. 356, May 17, 2016.

[5] R. Ge, G. Mai, P. Wang, M. Zhou, Y. Luo, Y. Cai, and F. Zhou, ''Crisprdigger: detecting crisprs with better direct repeat annotations,'' *Scientific reports*, vol. 6, no. 1, p. 32942, Sep 6, 2016.

[6] C. Bland, T. L. Ramsey, F. Sabree, M. Lowe, K. Brown, N. C. Kyrpides, and P. Hugenholtz, ''Crispr recognition tool (crt): a tool for automatic detection of clustered regularly interspaced palindromic repeats,'' *BMC Bioinformatics*, vol. 8, no. 1, p. 209, Jan 1, 2007.

[7] R. Jansen, J. D. A. van Embden, W. Gaastra, and L. M. Schouls, ''Identification of a novel family of sequence repeats among prokaryotes,'' *Omics : a journal of integrative biology*, vol. 6, no. 1, pp. 23--33, 2002.

[8] J. Godde and A. Bickerton, ''The repetitive dna elements called crisprs and their associated genes: Evidence of horizontal transfer among prokaryotes,'' *Journal of Molecular Evolution*, vol. 62, no. 6, pp. 718--729, Jun 2006.

[9] R. C. Edgar and E. W. Myers, ''Piler: identification and classification of genomic repeats,'' *Bioinformatics*, vol. 21, no. S1, p. i158, Jun 1, 2005.

[10] I. Grissa, G. Vergnaud, and C. Pourcel, ''The crisprdb database and tools to display crisprs and to generate dictionaries of spacers and repeats,'' *BMC bioinformatics*, vol. 8, no. 1, p. 172, May 23, 2007.

[11] M. A. Larkin, G. Blackshields, N. P. Brown, R. Chenna, P. A. McGettigan, H. McWilliam, F. Valentin, I. M. Wallace, A. Wilm, R. Lopez, J. D. Thompson, T. J. Gibson, and D. G. Higgins, ''Clustal w and clustal x version 2.0,'' *Bioinformatics*, vol. 23, no. 21, pp. 2947--2948, Nov 1, 2007.

[12] H. R. Hassanzadeh and M. D. Wang, ‘‘Deeperbind: Enhancing prediction of sequence specificities of dna binding proteins.’’ IEEE, Dec 2016, pp. 178--183.

[13] W. Jiang, I. Maniv, F. Arain, Y. Wang, B. R. Levin, and L. A. Marraffini, ‘‘Dealing with the evolutionary downside of crispr immunity: Bacteria and beneficial plasmids,’’ *PLoS genetics*, vol. 9, no. 9, p. e1003844, 2013.

[14] F. Chollet, ‘‘keras,’’ https://github.com/fchollet/keras, 2015.

[15] E. Jones, T. Oliphant, P. Peterson, *et al.*, ‘‘SciPy: Open source scientific tools for Python,’’ 2001--. [Online]. Available: http://www.scipy.org/

[16] ‘‘What are t values and p values in statistics?’’ 04 November, 2016. [Online]. Available: https://blog.minitab.com/blog/statistics-and-quality-data-analysis/what-are-t-values-and-p-values-in-statistics

[17] S. Deshmukh, ‘‘Visualizing crispr arrays,’’ San Jose State University, Tech. Rep., spring 2019 semester Bioinformatics(CS223) project.

[18] D. Couvin, A. Bernheim, C. Toffano-Nioche, M. Touchon, J. Michalik, B. NÃľron, E. P. C. Rocha, G. Vergnaud, D. Gautheret, and C. Pourcel, ‘‘Crisprcasfinder, an update of crisrfinder, includes a portable version, enhanced performance and integrates search for cas proteins,’’ *Nucleic acids research*, vol. 46, no. W1, p. W251, Jul 2, 2018.