IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 15, 2021, accepted July 22, 2021, date of publication July 26, 2021, date of current version August 11, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3100408

Sequence Alignment Using Machine Learning-Based Needleman–Wunsch Algorithm

AMR EZZ EL-DIN RASHED[1,2], HANAN M. AMER[1], MERVAT EL-SEDDEK[3], (Member, IEEE), AND HOSSAM EL-DIN MOUSTAFA[1]

[1]Department of Communications and Electronics, Faculty of Engineering, Mansoura University, Mansoura 35516, Egypt
[2]College of Computers and Information Technology, Taif University, Taif 26571, Saudi Arabia
[3]Higher Institute of Engineering and Technology, Mansoura 35516, Egypt

Corresponding author: Hossam El-Din Moustafa (hossam_moustafa@mans.edu.eg)

ABSTRACT Biological pairwise sequence alignment can be used as a method for arranging two biological sequence characters to identify regions of similarity. This operation has elicited considerable interest due to its significant influence on various critical aspects of life (e.g., identifying mutations in coronaviruses). Sequence alignment over large databases cannot yield results within a reasonable time, power, and cost. heuristic methods, such as FASTA, the BLAST family have been demonstrated to perform 40 times faster than DP-based (e.g., Needleman–Wunsch) techniques they cannot guarantee an optimum alignment result An optimized software platform of a widely used DNA sequence alignment algorithm called the Needleman–Wunsch (NW) algorithm based on a lookup table, is described in this study. This global alignment algorithm is the best approach for identifying similar regions between sequences. This study presents a new application of classical machine learning (ML) to global sequence alignment. Customized ML models are used to implement NW global alignment. An accuracy of 99.7% is achieved when using a multilayer perceptron with the ADAM optimizer, and up to 2912 Giga cell updates per second are realized on two real DNA sequences with a length of 4.1 M nucleotides. Our implementation is valid for RNA/DNA sequences. This study aims to parallelize the computation steps involved in the algorithm to accelerate its performance by using ML algorithms. All datasets used in this study are available from https://ieee-dataport.org/documents/dna-sequence-alignment-datasets-based-nw-algorithm.

INDEX TERMS Bioinformatics, DNA, RNA, pairwise sequence alignment (PWSA), Needlema–Wunsch (NW) algorithm, machine learning (ML) algorithms, multilayer perceptron (MLP), XGBoost algorithm.

CONTRIBUTION: This study presented six DNA/RNA sequence alignment datasets for one of the most common alignment algorithms, namely, the Needlema–Wunsch (NW) algorithm. It proposed a fast and parallel implementation of the NW algorithm by using machine learning techniques. This research is an extension and improved version of our previous work [1]. The current implementation achieved 99.7% accuracy by using a multilayer perceptron with the ADAM optimizer and up to 2912 Giga cell updates per second on two real DNA sequences with an of length 4.1 M nucleotides. Our implementation is valid for extremely long sequences by using the divide-and-conquer strategy.

I. INTRODUCTION

Bioinformatics has developed due to the need for understanding the code of life, i.e., deoxyribonucleic acid (DNA). It is an interdisciplinary research area that uses the principles of engineering, mathematics, and computer science to solve issues related to biology. Bioinformatics is an integration of biology and informatics because it includes the innovation of using computers in the measurement, recovery, control, and appropriation of information related to natural macromolecules, such as DNA, RNA, and proteins. Research endeavors in this field include genome assembly, sequence alignment, drug design, gene finding, drug discovery, protein structure alignment, and protein structure prediction [2].

Inside each cell in the human body is a complex molecule known as the ''hereditary material'' or DNA, which encodes the genetic instructions required for human development, functioning, and reproduction. DNA consists of two

The associate editor coordinating the review of this manuscript and approving it for publication was Xinyu Du.

complementary strands built from four simple units called chemical nucleotides (NTs): adenine (A), cytosine (C), guanine (G), and thymine (T). The four NTs comprise a four-letter set {A, C, G,}. Nearly all the cells in the human body contain similar DNA, and the majority of DNA is found in the cell nucleus. The human DNA has approximately 3 billion NTs. DNA is one of the most important examples of a biological sequence, and it can be represented using a long string of the letters ACGT [3].

Ribonucleic acid (RNA) is "a complex compound of high molecular weight that functions in cellular protein synthesis and replaces DNA as a carrier of genetic codes in certain viruses. It consists of four ribose NTs or nitrogenous bases: A, G, C, and uracil (U)." [4] U replaces T, which is present in DNA. Thus, the alphabet for an RNA sequence is also a four-letter set {A, C, G,}.

Pairwise sequence alignment (PWSA) is one of the most essential tasks in bioinformatics. It involves arranging a pair of genetic sequences (e.g., DNA and RNA) of characters to determine regions of similarity. It intends to identify the most optimal alignment with the highest total score, i.e., the maximum number of base-to-base matches, without altering the order of bases in either sequence. In addition, gap-to-gap matches are prohibited. Mismatches and gaps can be considered mutations and indels, respectively [5]. The computation time of an optimal PWSA increases proportionally with respect to the length of sequences. Consequently, producing timely results for large-scale problems requires more efficient algorithms and the use of parallel computing algorithms. PWSA can be used to identify the location of mutations between two viruses (e.g., coronaviruses), and this location can be used as a reference for the manufacture of coronavirus vaccines [6].

Various approaches for sequence alignment have been introduced. They include the dot-matrix technique; dynamic programming (DP) based algorithms, such as the Needleman –Wunsch (NW) algorithm, the Hirschberg algorithm for global alignment, the Smit–Waterman (SW) algorithm, the Gotoh algorithm, and the Miller–Myers algorithm for local alignment; and word techniques (i.e., heuristic methods). Although heuristic methods, such as FASTA, the BLAST family, and SIM2, have been demonstrated to perform 40 times faster than DP-based techniques, e.g., central processing unit (CPU)-based serial implementation of the SW algorithm, they cannot guarantee an optimum alignment result. That is, the output of these heuristic methods is only an approximation of the optimal solution. The NW and SW algorithms are widely known DP-based PSWA algorithms. They ensure that the most optimal alignment is found for a specified set of scoring functions from a mathematical perspective, and they require O(MN) calculations steps and runtime. Here, M denotes the length of the first sequence, and N indicates the length of the other sequence. That is, the runtime is proportional to the length of the sequences. These algorithms become time-consuming and require a huge number of calculations when aligning extremely long or multiple sequences (i.e., more than two sequences). Hence, optimization methods, such as Mille–Myers, and Hirschberg, can optimize space complexity into O(M+N), but are otherwise similar to NW and also require O(MN) runtime [7]. In general, the output of DP-based sequence alignment algorithms is classified into either global or local alignments. Table (1) provides examples of the alignment results of the SW and NW algorithms. Another important aspect is related to the alignment array of both algorithms. The text highlighted in green is the real alignment results when the SW algorithm is used. In contrast with the SW algorithm, the NW algorithm displays the complete set of input letters in the alignment result or array. As shown in Table (1), the alignment result is reshaped as a 1D array rather than a 2D array in the current study to aid the design process.

TABLE 1. Alignment result examples of the SW and NW algorithms.

Algorithm Name	SW Algorithm	NW Algorithm
Example	Sequence 1: GCCCTAGCG Sequence 2: GCGCAATG A match score of +1, a mismatch of −1, and a constant gap penalty of −1	Sequence 1: GCCCTAGCG Sequence 2: GCGCAATG A match score of +1, a mismatch of −1, and a constant gap penalty of −1
Alignment Result	GCCCTAGCG ||| GCGCAATG	GCCCTAGCG ||:| |::| GCGC−AATG
Reshaping Result (1D array)	GCG|||GCG	GCCCTAGCG||:||::|GCGC−AATG

The remainder of this paper is organized as follows. The background is provided in Section II. Related work is presented in Section III. The problem is defined in Section IV. Our proposed algorithm, computational parameters, and class reduction are described in Section V. The discussion and evaluation of the results are found in Section VI. Conclusions are drawn in Section VII, and the future directions of this research are introduced in Section VIII.

II. BACKGROUND

A. NW ALGORITHM

The NW algorithm is a type of pairwise sequence alignment algorithm based on D; it is mainly used to obtain the optimal global alignment between two biological sequences (e.g., DNA, RNA, and protein) in O(MN) time and space. The NW algorithm is divided into four phases: matrix initialization, similarity score calculation, traceback, and result generation [8]. Figure 1 shows the pseudocode of the algorithm for computing the scoring H matrix.

However, this algorithm requires too long running time (O(MN)) when aligning two, extremely long sequences. Our work aims to solve this problem (i.e., it accelerates the

```
AlignmentA ← ""
AlignmentB ← ""
W ← Gap penalty score
for i = 0 to length(A)
    H(i,0) ← d * i
for j = 0 to length(B)
    H(0,j) ← d * j
for i = 1 to length(A)
    for j = 1 to length(B)
    {
        Match ← H(i-1, j-1) + S(Ai, Bj)
        Delete ← H(i-1, j) + W
        Insert ← H(i, j-1) + W
        H(i,j) ← max(Match, Insert, Delete)
    }
```

FIGURE 1. Pseudocode of the NW algorithm for computing the scoring *H* matrix.

algorithm speed and improve the length of sequences) using ML algorithms. According to DP and the Divide & Conquer strategy, the sequence alignment problem (e.g., two long sequences) can be divided into smaller sub problems (two sequences with a length of 4NTs). Then, the smaller sub problems can be solved optimally, and their results can be used to construct the optimal result to the main problem. In addition, it Can be applied to problems that consist of overlapping sub-problems (e.g., two unequal length sequences).

B. MACHINE LEARNING (ML) TECHNIQUES

In the current study, the performance of 15 state-of-the-art techniques in predicting optimal sequence alignment based on the NW algorithm is investigated. These techniques include multilayer perceptron (MLP), support vector machine (SVM), and the XGBoost classifier. In the two succeeding subsections, we briefly describe the top two techniques used in the current study (i.e., MLP and XGBoost). In addition, to the best of our knowledge, this is the first time in which ML is used in sequence alignment and the first implementation of sequence alignment datasets based on global alignment techniques (NW algorithm).

C. MLP

Artificial neural networks (ANNs) comprise a group of efficient and flexible methods that are used for classification and regression tasks in various real-world problems because of their inherent learning capability.

MLP is a type of ANN that contains three primary layers: the input, hidden or intermediate, and output layers. Each layer contains a large number of processing elements, called artificial neurons. The input layer contains the input information for a task, and the output layer contains the target solution for a task. Moreover, the hidden or intermediate layer is responsible for data processing and transmission between the input and output layers. It also deals with the nonlinearity and complexity of a problem. In addition, MLP requires a small training set and can be easily implemented [9], [10].

In MLP, each neuron j in the hidden or intermediate layer obtains the sum of its input variables x_i after multiplying them by the related connection weights w_{ij} and then computes its output y as a sum of products [11]. Mathematically,

$$y_i = f\left(\sum w_{ij} \times O_i\right).$$

Scikit-learn uses three solvers or weight optimization algorithms [12]: stochastic gradient descent (SGD; limited-memory Broyde–Fletcher–Goldfarb–Shanno (L-BFGS), which is an optimizer in the family of quasi-Newton method; and ADAM, which is an effective and popular SGD-based optimizer introduced by Kingma *et al.* [13].

D. XGBoost

XGBoost stands for extreme gradient boosting. This algorithm is a type of ensemble technique that includes stacking, bagging, and boosting methods. Boosting methods comprise a group of low-accuracy classifiers used to develop a highly accurate classifier (i.e., a strong classifier provides a low error rate) through optimization steps for every new tree that attaches. These algorithms are less affected by the overfitting problem. XGBoost has been proven to push the limits of computing power for boosted tree algorithms. It is approximately 10 times faster than traditional ML methods, and it exhibits the advantage of parallel processing (i.e., it uses all the cores of the machine). In addition, XGBoost allows the use of a wide variety of computing environments, and it can be handled by a variety of programming languages (e.g., Java, Pytho, R, and C++) [14], [15].

III. RELATED WORK

Khaled *et al.* [16] demonstrated a novel implementation that accelerated the PWSA algorithm for DNA sequencing by using the general-purpose graphics processing unit (GPU) architecture apart from the DP techniques provided by the SW algorithm. Their proposed parallel computing platform and model uses CUDA®, which was created by Nvidia®. This implementation can achieve 7.064 ms for a pair of sequences with a length of 1024 NTs (at threshold k = 1).

A parameterizable implementation on CUDA-compatible GPUs [17] used the divide-and-conquer (D&C) strategy to calculate the alignment matrix by dividing the entire matrix computation into small submatrices and allocating the available number of threads and memories to each submatrix. This technique achieved up to 4.2 Giga cell updates per second (GCUPS) on the Swiss-Prot database on GeForce 8800 GTX, and it was 15 times faster than the CPU implementation. However, the performance of this implementation deteriorates as query sequence length increases.

Chaudhary [18] demonstrated a new parallel approach of the NW algorithm. This approach uses skewing transformation for the traversal and calculation of the DP matrix. The execution times of the sequential CPU-based and parallel GPU-based implementations were compared. The GPU-based implementation achieves up to six times

performance improvement compared with the sequential CPU-based implementation. This technique takes approximately 12 s to align two sequences with a length of 14336 NTs.

In [19], a novel framework for accelerating pairwise SW sequence alignment using the CUDA parallel paradigm was presented on Nvidia® GPU. The key idea is to implement a new algorithm that assigns different NT weights by using a GPU architecture and then merging the subsequences of the match by using a CPU to achieve optimum local alignment. However, this framework was tested on extremely small sequence sizes ranging from 16 base pairs (bps) to 1024 bps.

In [20], the researchers aimed to parallelize the computation involved in the SW algorithm to accelerate performance by using CUDA. They introduced a heterogeneous antidiagonal approach that benefits from the interaction between the CPU-based serial design and the GPU-based parallel design. This technique is valid for extremely long sequences and can align two sequences with a length of 3 M NTs in 29499 s (computational time). In addition, the computational time of their proposed technique increases gradually as sequence length increases, whereas that of the serial approach increases rapidly.

A new parallel method for the SW algorithm was introduced in [21]. This method utilizes the parallelism of the columns of the similarity matrix to parallelize the SW algorithm on a heterogeneous system based on CPU and GPU. This approach achieves 37 times higher speed than OSEARCH (a sequential algorithm), and it exhibits the advantages of CPU and GPU. However, it fixes the target sequence length to 361 bps.

In [22], the authors intended to improve the performance of the NW algorithm through three implementations that are a mixture of specialized software and hardware solutions on large-scale input without affecting accuracy. Their experiments showed that the GPU-based implementation is better, achieving performance that is 72.5 times faster than that of the sequential implementation. Aligning two sequences with a length of 20 k NTs took only 0.599 s (running time).

The efficiency of sequence alignment based on a parallel multithreaded model design of the NW algorithm was investigated in [23]. This model was then verified via a multithreaded parallel program implementation that utilized OpenMP. This model took 1157.830 s (i.e., 19 min) to align two sequences with a length of 50 k NTs.

The 1D pairwise convolutional neural network (CNN) algorithm described in [24] failed to align two long sequences with lengths of 24343 NTs and 42028 NTs. This finding implied that considerable memory is required to use this method in aligning a pair of extremely long DNA sequences.

By contrast, our ML model can align two long DNA or RNA sequences without requiring much memory.

Our technique is based on the D&C strategy. It places certain constraints on the length and type of a sequence, i.e., an equal-length DNA or RNA sequence used with a fixed length of four, which, in turn, can reduce the steps required to implement the NW algorithm. In accordance with the D&C strategy, the alignment process (i.e., the primary problem) can be divided into smaller subproblems. Then, the smaller subproblems can be solved optimally, and their results can be used to construct the optimum solution for the primary problem.

TABLE 2. Binary and decimal representations of DNA NTs.

DNA Sequence 1	DNA Sequence 2	Description
ACGT (Four letters)	ACTG (Four letters)	Nucleobase representation
00–01–10–11	00–01–11–10	Binary representation
0–1–2–3	0–1–3–2	Decimal representation

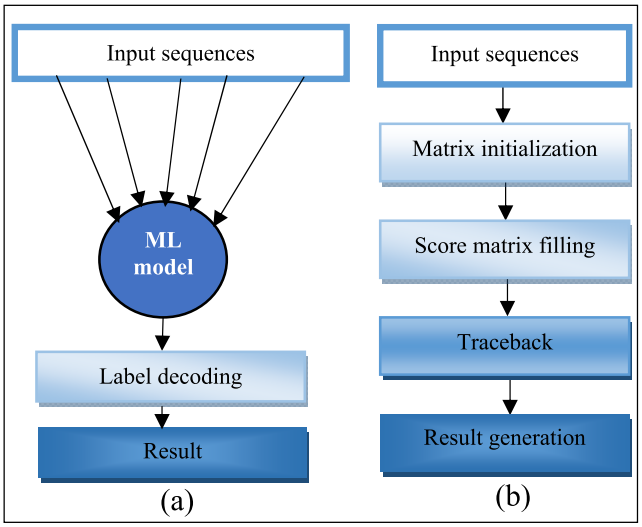


FIGURE 2. (a) Proposed parallel workflow and (b) traditional NW algorithm sequential workflow.

As shown in Figure 2, the general workflow of the NW algorithm has four major steps: initializing, calculating the scoring matrix, performing traceback, and generating results. These steps are unnecessary in our proposed algorithm. Thus, our technique can decrease the runtime and the required number of calculations of the alignment algorithm. Our proposed algorithm regards DNA sequence alignment as a text classification problem. This study uses classical ML algorithms to implement the NW algorithm based on a limited number of output classes or alignment patterns. Table (2) provides the binary and decimal representations of DNA NTs. Table (3) presents an example of two DNA sequences and their corresponding binary and decimal representations. Each letter is converted into 2 bits in accordance with the binary representation and 1 digit in accordance with the decimal representation.

IV. PROBLEM DEFINITION

Biological sequence alignment algorithms are time-consuming even when their implementation uses accelerating

TABLE 3. Examples of two DNA sequences and their corresponding binary and decimal representations.

Amino Acid Letters	Decimal	Binary	NT Names
A	0	00	Adenine
C	1	01	Cytosine
G	2	10	Guanine
T	3	11	Thymine

hardware platforms because of the following reasons. (1) The number of biological sequences (i.e., DNA sequences) and the sequence length are increasing with time (i.e., the big data problem). (2) Common algorithms (i.e., NW and SW algorithms) that are used to align the sequences require O(MN) calculation steps (i.e., initialization, matrix filling, traceback, and result generation) and consume O(MN) time (M and N are the lengths of the two input sequences). (3) Basic sequence alignment algorithms are internally dependent on the sequential process that consists of four steps, and each step is dependent on previous steps. (4) The hardware implementation of sequence alignment algorithms that uses multi-core processors, GPU, or field-programmable gate array (FPGA) does not provide an effective solution to sequential process problems, which affect system speed and memory requirements. (5) DP algorithms guarantee optimal alignment results, although they are slower than FASTA and BLAST. Moreover, they require extensive computation time and memory due to the sequential processes, and such time and memory are proportional to the length of the sequences. Although FASTA and BLAST are fast, they do not guarantee optimum alignment. (6) Most of the studies discussed in the previous section (RELATED WORK) cannot deal with extremely long sequences by using parallel or sequential implementations.

Our proposed algorithms depend on the parallelization of NW for DNA/RNA sequences by using fast and efficient ML algorithms under certain constraints to overcome most of the issues that arise because of the sequential process problem. This technique is also valid for extremely long sequences.

V. PROPOSED ALGORITHM

In the current study, we propose the use of equal-length sequences (i.e., multiples of four N = 4, 8, 12, . . .) that can be applied to DNA or RNA sequences because DNA and RNA sequences consist of four letters of the alphabet that represent the four NTs. Meanwhile, protein sequences consist of 20 letters. The alignment type used in the present study is a pairwise sequence, and our proposed technique is applied to one of the commonly used global alignment algorithms (i.e., the NW algorithm). In addition, our implementation is based on a lookup table or dataset construction. The input of the dataset (features or attributes) is two DNA sequences in binary or decimal representation, and the target is the alignment array category. The NW algorithm presents the complete set of input letters in the alignment array.

TABLE 4. Dataset of computational parameters.

Parameters	Settings
Alignment Algorithm	Global
Type of Gap Penalty	Linear
Gap Opening	−5
Gap Extension	−5
Substitution Matrix	BLOSUM 50

Our implementation constructs a dataset of all feasible combinations of the two DNA input sequences after converting the DNA sequence from alphabets into binary or decimal representations. The dataset presents each feasible DNA input sequence combination with the alignment result or alignment array (target) depending on the combination of DNA input sequences and computational parameters.

Two input DNA sequences exist, and each sequence consists of four letters, implying that each sequence will require 8 bits for binary representation or four digits for decimal representation. The total number of bits required for binary representation is 16 bits or eight digits for decimal representation. The total number of rows (possibilities) in the dataset or truth table is 216, which is equal to 65536 rows. A classical MATLAB built-in function is used to obtain the alignment for each sequence pair for the NW algorithm. The alignment array produced from the MATLAB function in the character array consists of three rows. The two sequences appear in the first and third rows, and the symbols representing the optimal global alignment for these sequences appear in the second row. For the proposed algorithm, the alignment arrays are reshaped into one row. Therefore, we have a dataset that consists of two DNA sequences (16 bits) as the input features and global alignment arrays as the output target, instead of applying all the sequential steps and equations to the global alignment algorithms to obtain the scoring matrix and alignment array. Table (4) lists the computational parameters used in the current study.

Table (5) presents a portion of the truth table for the NW algorithm. The information in this table is arranged as follows. The first to fifth columns list the row number, binary representations, decimal representations, DNA sequences related to the binary data, and global alignment arrays, respectively, after they are reshaped into a single row. Notably, all the sequence letters appear in the global alignment array, but only the matched letters appear in the local alignment array.

The alignment arrays for global alignment are analyzed. Table (6) lists the numbers of unique and repeated alignment arrays. Evidently, no repeated alignment arrays are found for global alignment because all the input DNA letters are fully represented in the alignment array. These alignment arrays are then reduced into a specific number of classes. Table (7) provides the symbols used in the alignment array and their corresponding description and function.

TABLE 5. Portion of the NW algorithm.

Row Number	Truth Table (Binary Data)	Truth Table (Decimal Data)	Related DNA Sequences	Alignment Arrays (Target)
	Sequence1–Sequence2	Sequence1–Sequence2	Sequence1–Sequence2	NW Algorithm
0	00000000–00000000	0000–0000	AAAA–AAAA	'AAAA||||AAAA'
1 ….	00000000–00000001	0000–0001	AAAA–AAAC	'AAAA||| AAAC'
95 ….	00000000–01011111	0000–1133	AAAA–CCTT	'AAAA:: CCTT' (Full Mismatch)
65535	11111111–11111111	3333–3333	TTTT–TTTT	'TTTT||||TTTT'

TABLE 6. Number of unique and repeated alignment arrays for the NW algorithm.

Number of Alignment Arrays	NW Algorithm Count (%)
Unique Alignment Arrays	65536 (100%)
Remaining Alignment Arrays	0 (0%)
Total Number	65536 (100%)

TABLE 7. Symbols, description, and function.

Symbols	Descriptions	Functions
'|'	Vertical Bar	Match
':'	Colon	Mismatch
'-'	Hyphen	Gap
' '	Space	Mismatch

A. DETERMINING THE SUBSTITUTION MATRIX AND GAP PENALTY SCHEME

A substitution matrix assigns a score for match or mismatch to each pair of bases or amino acids. Many substitution matrices, such as the BLOSUM series (50, 62) or the PAM series (80, 250), are available for sequence alignment. In this study, BLOSUM 50 is configured as the scoring matrix, i.e., s(a_i, b_j), to calculate the alignment matrix score H(i, j) [25].

A gap penalty function determines the score cost for opening or extending gaps. Gap penalties have two types: linear and affine.

A linear gap penalty scores all opening and extending gaps (i.e., indels) equally. This study adopts a linear gap penalty cost of five $W_k = kW_1$ where W_1 is the cost of a single gap, and k indicates gap length.

Gap penalty is directly proportional to gap length. When a linear gap penalty is imposed, the scoring matrix of the SW algorithm can be presented as follows:

$$H_{ij} = \max \begin{cases} H_{i-1,j-1} + s(a_i, b_j), \\ Hi - 1, j - W_1, \\ H_{i,j-1} - W_1, \\ 0, \end{cases}$$

where $H_{i-1,j-1}+s(a_i, b_j)$ is the score of aligning a_i and b_j, and 0 indicates no similarity up to a_i, and b_j. The scoring matrix of the NW algorithm can also be presented as follows:

$$H_{ij} = \max \begin{cases} H_{i-1,j-1} + s(a_i, b_j), \\ Hi - 1, j - W_1, \\ H_{i,j-1} - W_1, \end{cases}$$

where $H_{i-1,j-1} + s(a_i, b_j)$ is the score of aligning a_i, and b_j.

B. CLASS REDUCTION FOR THE NW ALGORITHM

Three class reduction techniques are used in this study. The first technique reduces the number of classes from 65536 to 254. The second technique reduces the number of classes from 254 to 239, and the third technique reduces the number of classes from 239 to 221. As shown in Table (8), the final column contains an alignment array after replacing each letter with an asterisk. This replacement reduces the number of classes to 254 instead of 65536 as shown in Table (9). The final column in Table (10) lists the proposed indices for each pattern. These indices are subsequently used as the target output for the ML algorithms.

The second reduction technique depends on merging all the mismatch conditions into one, e.g., ('****::::****'). This technique reduces the number of classes to 239 instead of 254. The final reduction technique is based on removing all the classes represented in less than 10 instances in the dataset. That is, we have six datasets, and three of them have a binary input (16 bit) and the other three have a decimal input (eight digits). In addition, the first two datasets have 254 output classes, the next two datasets have 239 output classes, and the last two datasets have 221 output classes.

VI. DISCUSSION AND RESULTS

A Dell G3 laptop with Intel® Core i7-9750H six-core 2.60 GHz CPU, 16 GB RAM, Nvidia GeForce GTX 1660 Ti with Max-Q design, and 500 GB SSD with MATLAB 2020a 64 bit, Anaconda with Orange3 (version 3.26), and PyChar community 2020 running on Windows 10 64-bit operating system is used in this study. Dataset implementations of the NW sequence alignment algorithms are written in MATLAB, and the other ML algorithms are implemented using Python with Orange library and scikit-learn library.

TABLE 8. NW alignment array after using an asterisk for alignment array and label encoding.

Row Number	Truth Table (Binary Data)	Truth Table (Decimal Data)	Related DNA Sequences	Alignment Arrays (Target)	Alignment Arrays after Replacing Each Letter with an Asterisk
	Sequence1–Sequence2	Sequence1–Sequence2	Sequence1–Sequence2	NW Algorithm	
0	00000000–00000000	0000–0000	AAAA–AAAA	'AAAA||||AAAA'	'****||||****'
1 ….	00000000–00000001	0000–0001	AAAA–AAAC	'AAAA||| AAAC'	'****||| ****'
95 ….	00000000–01011111	0000–1133	AAAA–CCTT	'AAAA:: CCTT' (Full Mismatch)	'****::::****'

TABLE 9. NW alignment arrays after replacing each letter with an asterisk.

Alignment Arrays	Count
Number of Unique Instances	254
Number of Repeated Instances	65282
Total Number of Alignments	65536

TABLE 10. Label encoding for 254 alignment patterns.

Unique Alignment (254) Patterns	Count	Percentage	Encoded Indices
'****||||****'	256	0.3906	130
'****||| ****'	512	0.7813	128
'****|||:****'	256	0.3906	129
'****|| |****'	512	0.7813	124
'****|| ****'	640	0.9766	122
'****|| :****'	480	0.7324	123
'****||:|****'	256	0.3906	127
'****||: ****'	480	0.7324	125
'****||::****'	256	0.3906	126

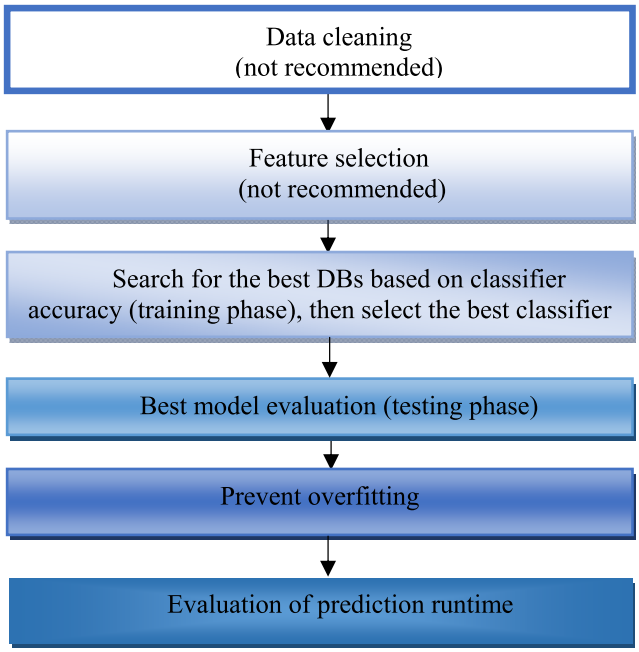


FIGURE 3. Block diagram of our proposed workflow.

Fifteen ML classifiers, including MLP, SVM, decision tree, and SGD, are selected. In addition, several ensemble methods are used as averaging techniques. (i.e., bagging method and random forest classifier) and boosting strategies (i.e., Adaboost and XGBoost classifier). These methods are trained and evaluated on six datasets, as shown in Table (11). We adopt 10-fold cross-validation for the training phase. The datasets are tested with and without shuffling during the training phase but only with shuffling (class randomization) during the evaluation phase.

The original dataset is the third dataset. It has a binary input of 16 bits and 254 classes. The secondary datasets are generated from the original dataset by first converting the 16-bit binary input into an input with eight decimal digits and then merging all the fully mismatched cases into one. This process reduces the number of classes from 254 to 239. The first two datasets have a decimal input of eight digits and different numbers of classes (254 or 239) as the target or output data. Moreover, the next two datasets have a binary input of 16 bits and different numbers of classes (254 or 239) as the target or output data. The last two datasets have the lowest number of instances with 221 classes as the target. Figure 3 illustrates our proposed workflow in this study. Notably, some steps are not recommended, as indicated in the figure.

Five common tasks are typically performed during data preparation for ML application. These tasks are data cleaning (i.e., detecting and correcting errors, such as missing values, in the dataset); data transformation (e.g., discretization, one-hot encoding, normalization, and standardization; feature selection; feature engineering (i.e., deriving new input variables from the available variables in the dataset); and dimensionality reduction, such as principal component analysis (PCA) and linear discriminant analysis [26].

A. DATA CLEANING, TRANSFORMATION, AND REDUCTION

In our case, we do not recommend data cleaning because no data are missing. Moreover, no duplicated data (i.e.,

TABLE 11. Description of datasets.

	Dataset 1	Dataset 2	Dataset 3	Dataset 4	Dataset 5	Dataset 6
Number of Instances (Size)	65536	65536	65536	65536	65483	65483
Input Attributes (Features)	8 (Decimal)	8 (Decimal)	16 (Binary)	16 (Binary)	8 (Decimal)	16 (Binary)
Output Classes	254	239	254	239	221	221

TABLE 12. Feature selection by using a filter method (i.e., chi-squared) for the best datasets.

Feature Number	χ^2 (Dataset 3)	χ^2 (Dataset 6)	Feature Number	χ^2 (Dataset 3)	χ^2 (Dataset 6)
fr8	2928.6962	2335.2803	fr3	514.3212	497.2075
f16	2912.0966	2318.7507	fr11	495.2717	478.7286
fr4	2749.3369	2168.6677	fr5	472.0676	455.0509
fr12	2742.2791	2161.9829	f13	468.8299	451.2413
fr10	2735.2433	2140.9491	fr9	430.3850	407.9344
fr2	2727.2088	2130.2249	fr1	425.5596	402.7023
fr6	2700.1408	2120.4733	fr7	374.5339	355.1349
fr14	2692.3486	2112.1734	fr15	375.7883	356.3789

in any attribute or instance) or incorrect or corrupted data exist. Thus, removing any data (i.e., attribute or instance), is unnecessary. In addition, the input data are discrete and contain two values (0, 1) for binary input datasets or four values (0, 1, 2, 3) for decimal input datasets. Furthermore, normalization does not affect the accuracy or performance of the ML algorithm. We also believe that data transformation will not achieve any reasonable improvement in ML model performance. Lastly, feature reduction by using PCA also does not help because the number of input features is small and independent) [27], [28].

B. FEATURE SELECTION

Feature selection can be classified into two categories: label information (i.e., supervised, unsupervised, and semi-supervised) and search strategy. Filter selection based on search strategy can be further classified into four techniques: intrinsic method (e.g., decision trees), filter method (e.g., chi-square, Pearson's correlation, and ANOVA F-value), wrapper method (i.e., forward selection, backward elimination, and recursive feature elimination), and embedded method (e.g., lasso L1 regularization, and ridge L2 regularization). Filter methods are extremely fast, and they select the most discriminative features on the basis of data behavior. In general, filter methods perform feature selection before classification (i.e., the first step in any feature selection pipeline) and are typically a two-step technique. First, all the features are ranked in accordance with certain criteria. Then, the features with the highest rankings are selected. Wrapper methods use the intended learning algorithm itself (e.g., XGBoost classifier, random forest, and recursive feature elimination) to evaluate features [29], [30].

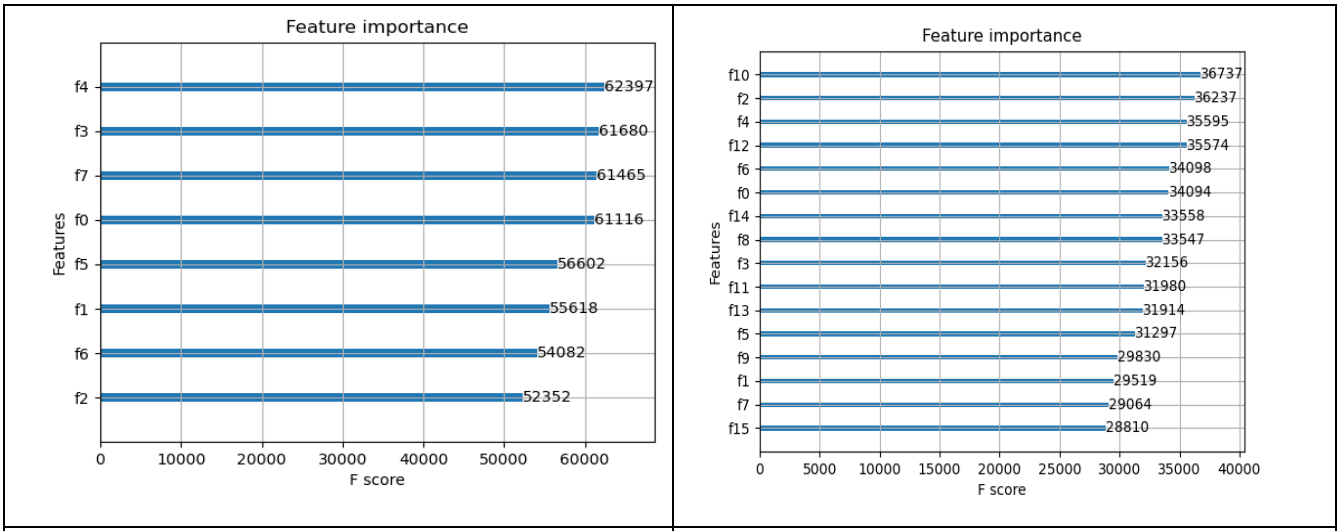
This step does not achieve any reasonable improvement in ML model performance. Table (12) provides the chi-squared rank order for Datasets 3 and 6, which are the best datasets in terms of training phase accuracy. This table is arranged in accordance with feature importance, and the eight best features for the best datasets are Fr2, Fr4, Fr6, Fr8, Fr10, Fr12, Fr14, and Fr16. In addition, our ML algorithms are trained on these highest-ranking features (i.e., eight features) and no reasonable improvement is achieved (i.e., best model accuracy is 6.4% on Dataset 3 among the fast-training models).

For the slow-training models, the XGBoost classifier is used to calculate the F-scores or feature importance scores of all the member features in all the training datasets (e.g., DB5, and DB6), as shown in Figure 4. Then, the model is fitted using each feature as the threshold. The performance (i.e., test set accuracy) of the XGBoost model generally declines with the number of selected features, as indicated in Tables (13) and (14). In this problem, a trade-off among features occurs to test set accuracy. In general, feature selection performance depends on the selected hyperparameters, and it is an open problem [29].

In accordance with the preceding results, we train our models with all the input features without performing any data preprocessing on the input datasets (as discussed later).

C. SEARCH FOR THE BEST DATASET

The search for the best model and datasets involves training 15 ML classifiers by implementing 10-fold cross-validation on all the training datasets. We start with the fast-training classifiers and then proceed to the slow-training ones. Notably, some ensemble methods (e.g., voting classifier and histogram gradient boosting classifiers) are not tested here either because of memory crashes or these methods require an extremely long computational time to train. Table (15) provides the training accuracy summary results of all the datasets. This table also includes the average training time for each model. Our best model is the MLP classifier, and it



(a) Feature importance for Dataset 5 | (b) Feature importance for Dataset 6

FIGURE 4. Trained XGBoost model calculates feature importance scores for all the member variables in our predictive modeling problem.

TABLE 13. Accuracy for all the datasets by using selected features (i.e., the wrapper method) for the XGBoost classifier.

Dataset 1			Dataset 2			Dataset 3		
Number of Features	Threshold	Accuracy	Number of Features	Threshold	Accuracy	Number of Features	Threshold	Accuracy
16	--	--	16	--	--	16	0.052	0.9641
15	--	--	15	--	--	15	0.056	0.2990
8	0.117	0.9683	8	0.028	0.9564	14	0.057	0.3517
7	0.119	0.3323	7	0.029	0.3830	13	0.057	0.1807
6	0.123	0.2048	6	0.031	0.3097	12	0.059	0.0984
5	0.124	0.1532	5	0.120	0.2158	10	0.061	0.0973
4	0.125	0.0825	4	0.130	0.0258	9	0.063	0.1145
3	0.130	0.0606	3	0.141	0.1131	8	0.065	0.0737
2	0.130	0.0294	2	0.240	0.0322	2	0.069	0.0232
1	0.132	0.0220	1	0.282	0.1332	1	0.070	0.0216

TABLE 14. Accuracy for all the datasets by using selected features (i.e., the wrapper method) for the XGBoost classifier.

Dataset 4			Dataset 5			Dataset 6		
Number of Features	Threshold	Accuracy	Number of Features	Threshold	Accuracy	Number of Features	Threshold	Accuracy
16	0.057	0.9632	16	--	--	16	0.050	0.9613
15	0.057	0.3380	15	--	--	15	0.052	0.3295
14	0.057	0.3620	8	0.023	0.9186	14	0.056	0.1810
13	0.058	0.2024	7	0.027	0.3783	13	0.056	0.2377
12	0.058	0.1529	6	0.028	0.2571	12	0.063	0.2015
11	0.059	0.1558	5	0.035	0.2207	11	0.063	0.2407
10	0.061	0.1765	4	0.166	0.0146	10	0.064	0.2152
9	0.061	0.1612	3	0.225	0.1348	9	0.064	0.1660
8	0.061	0.1579	2	0.233	0.1389	2	0.067	0.1348
7	0.064	0.1544	1	0.263	0.1348	1	0.071	0.1348

achieves the best performance on Datasets 3 and 6. The next best model is XGBoost. The top three models for each dataset are printed in bold. The top three models for the datasets with decimal input are MLP, XGBoost, and AdaBoost. The top three models for the datasets with binary input are MLP, XGBoost, and SVM–radial basis function (RBF). In addition,

TABLE 15. Training accuracy summary results on all the datasets.

Model	Training Time (s)	Dataset 1 Accuracy	Dataset 2 Accuracy	Dataset 3 Accuracy	Dataset 4 Accuracy	Dataset 5 Accuracy	Dataset 6 Accuracy
1-Neural Network (MLP)	24560.971	0.9650	0.9570	0.9847	0.9818	0.9584	0.9838
2-XGBoost	70999	0.9758	0.9168	0.9641	0.9694	0.8929	0.9689
3-AdaBoost	12.294	0.5217	0.5214	0.2912	0.3788	0.5231	0.3802
4-Random Forest	24.251	0.4362	0.4681	0.3449	0.3927	0.4649	0.3877
5-Decision Tree	447.251	0.4025	0.4000	0.2188	0.2493	0.4014	0.2480
6-SVM–RBF	12967.743	0.3486	0.3667	0.8272	0.8415	0.3670	0.8414
7-*k*-NN	78.96	0.2247	0.2770	0.2565	0.3070	0.2745	0.3089
8-CN2 Rule Inducer	150303.332	0.1946	0.2405	0.1699	0.2110	0.2362	0.2108
9-Extra Trees	67.152	0.0678	0.1391	0.0713	0.1391	0.1392	0.1392
10-NB	0.573	0.0218	0.0878	0.0090	0.1066	0.1460	0.1403
11-SGD	139.872	0.0147	0.0393	0.0225	0.0484	0.0279	0.0453
12-SVM-Linear	4718.23	0.0095	0.0131	0.0232	0.0241	0.0130	0.0247
13-LR-Ridge	36780.96	0.0069	0.1391	0.0224	0.1398	0.1392	0.1399
14-LR-Lasso	1300.71	0.0068	0.1391	0.0226	0.1395	0.1392	0.1396
15- Bagging Classifier	224.31	0.2481	0.1511	0.1947	0.1402	0.1518	0.1405

TABLE 16. Summary of the experiment results of a neural network (i.e., MLP classifier with different optimizers) on dataset 3.

Model	Optimizer	Classification Accuracy	F-measure	Precision	Recall
Neural Network (MLP)	ADAM	0.9925	0.9924	0.9924	0.9925
Neural Network (MLP)	SGD	0.9847	0.9842	0.9841	0.9847
Neural Network (MLP)	L-BFGS-B	0.9842	0.9840	0.9840	0.9842

TABLE 17. Summary of the experiment results of a neural network (i.e., MLP classifier with different optimizers) on dataset 6.

Model	Optimizer	Classification Accuracy	F-measure	Precision	Recall
Neural Network (MLP)	ADAM	0.9927	0.9927	0.9927	0.9927
Neural Network (MLP)	SGD	0.9838	0.9833	0.9833	0.9838
Neural Network (MLP)	L-BFGS-B	0.9853	0.9853	0.9854	0.9853

the experimental summary results of using MLP with three optimizers on Datasets 3 and 6 are presented in Tables (16) and (17), respectively. The ADAM optimizer achieves the best performance on the two datasets.

D. EVALUATION METRICS

We evaluate the classification results when searching for the best model and dataset on the basis of overall accuracy (all metrics are calculated but not presented in this paper). Considering the imbalanced datasets and classification sensitivity, precision, and F1 score indicators are calculated and presented for our best model (MLP) because these indicators focus on one class. These performance measures are defined and computed as follows [31], [32]:

$$\text{Accuracy} = (\text{TP} + \text{TN})/(\text{TP} + \text{FP} + \text{FN} + \text{TN}).$$

$$\text{Recall} = \text{TP}/(\text{TP} + \text{FN})$$

$$\text{Precision} = \text{TP}/(\text{TP} + \text{FP})$$

$$\text{F1 score} = 2^{*}(\text{Recall}^{*}\text{Precision})/(\text{Recall} + \text{Precision})$$

E. BEST MODEL EVALUATION

For the testing phase, we use the same best datasets (Datasets 3 and 6) after randomizing or shuffling rows or classes by 10% and then testing the datasets by using our best model (MLP with the ADAM optimizer). Table (18) provides the evaluation results of MLP on the test datasets. We observe a trade-off between test dataset accuracy and shuffling percentage, although the shuffle parameter is activated in the MLP classifier, the batch size is automatic, and L2 penalty regularization (alpha) = 0.0001. We simplify the model by decreasing the number of hidden neurons from 200 to 100.

F. OVERFITTING PREVENTION

To prevent overfitting, increasing the amount of data in datasets is an improbable technique because all the

TABLE 18. Evaluation result summary of a neural network (i.e., MLP classifier (for datasets 3T and 6T (testing phase).

Model	Optimizer	Classification Accuracy	F-measure	Precision	Recall
Neural Network (MLP)- Dataset 3T	ADAM	0.859	0.859	0.859	0.859
Neural Network (MLP)- Dataset 6T	ADAM	0.860	0.859	0.859	0.860

TABLE 19. Evaluation result summary of a neural network (i.e., MLP classifier) for datasets 3 and 6T (testing phase) when using overfitting prevention technique.

Model-Dataset	Optimizer	Classification Accuracy	F-measure	Precision	Recall
Neural Network (MLP)- Dataset 3T	ADAM	0.9889	0.9889	0.9889	0.9889
Neural Network (MLP)- Dataset 6T	ADAM	0.9879	0.9879	0.9879	0.9879
Neural Network (MLP)-Dataset 4.1 M	ADAM	0.9970	0.9970	0.9970	0.9970

TABLE 20. Proposed techniques for preventing overfitting.

Technique	Used	Technique	Used
Using more data	No	Increasing L2 penalty regularization	Yes
Shuffling data	Yes	Dropout	No
Cross-validation	Yes	Augmentation	No
Automatic batch size	Yes	Simplifying the model by using fewer hidden neurons	Yes
Adaptive learning rate	Yes	Feature selection	No
Early stopping	Yes	Ensemble methods	No

TABLE 21. Evaluation of the best model's prediction runtime.

Platform	Query Length (NT)	256	52.4 k	80 k	128 k	3.3 M	4.1 M
Single Core	Time (s)	0	0.19648	0.14369	0.21977	5.8055	6.0417
	GCUPS	--	13.9748	44.5403	74.5507	1939	2912

possibilities are already presented in our current datasets unless this process is performed on classes with fewer instances, such as in the synthetic minority oversampling technique (i.e., SMOTe for imbalanced datasets). Thus, any increase in data will be a repetition. In addition, performing augmentation or randomizing features in a dataset is impossible because the target depends on the order of the input features. Therefore, we can use the automatic batch size instead of large values. We can also adopt several regularization techniques, such as early stopping, dropout (not supported in the scikit-learn MLP classifier), and adaptive learning rates, instead of a constant value (i.e., default value) and increasing the value of L2 penalty (alpha) regularization. Increasing alpha may address high variance (a sign of overfitting) by encouraging smaller weights, resulting in a decision boundary plot with fewer curvatures. Moreover, increasing L2 penalty (alpha) regularization does not improve the performance (i.e., alpha = 90, Dataset 3, accuracy = 85.9% and alpha = 0.04, Dataset 6, accuracy = 86.9%). In addition, enhancing accuracy is possible by using hyperparameter tuning methods, such as a genetic algorithm (GA) [33], grid search, random search [34], coordinate descent, and SGD [35].

Table (19) presents the summary of the results on the test datasets after using our proposed technique to prevent overfitting [36]. This strategy is summarized in Table (20).

Figure 5 illustrates the strategy for testing new data on our best model. First, the input sequence should be merged, and then each letter is encoded into the binary data (as data in Dataset 3 or 6). Subsequently, these data are applied to our best model. The predicted label takes values from 1 to 254 (for the MLP model trained on Dataset 3) and from 1 to 221 (for the MLP model trained on Dataset 6). These values must be decoded two times to obtain the real alignment output. After shuffling its classes by 10%, Dataset 3 is referred to as Dataset 3T (the same technique is used for Dataset 6T). In addition, we assign two real DNA sequences: NC_000962.3 represents the *Mycobacterium tuberculosis* H37Rv genome with a length of 4,411,532 NTs (referred to as D4.4) and NC_000913.3 represents the *Escherichia coli* K12 MG1655 genome with a length of 4,641,652 NTs (referred to as D4.6). Query length is the number of NTs in

TABLE 22. Performance comparison with other state-of-the-art implementations.

Reference	Year	Sequence Pairs	Time (s)	GCUPS
[16]	2013	1024 NTs	7.065	1.4842×10^{-4}
[38]	2014	D4.4 vs. D4.6	203	100.7
[18]	2015	14336 NTs	12	0.0171
[22]	2019	20 k NTs	0.5995	0.6672
[23]	2019	50 k NTs	1157.8305 (19 min)	0.0022
[20]	2020	3 M NTs	29499 (492 min)	0.3051
	Proposed	52.4 k NTs (Dataset 3T)	0.19648	13.9748
	Proposed	4.1 M NTs (Dataset 4.1 M)	6.0417	2912

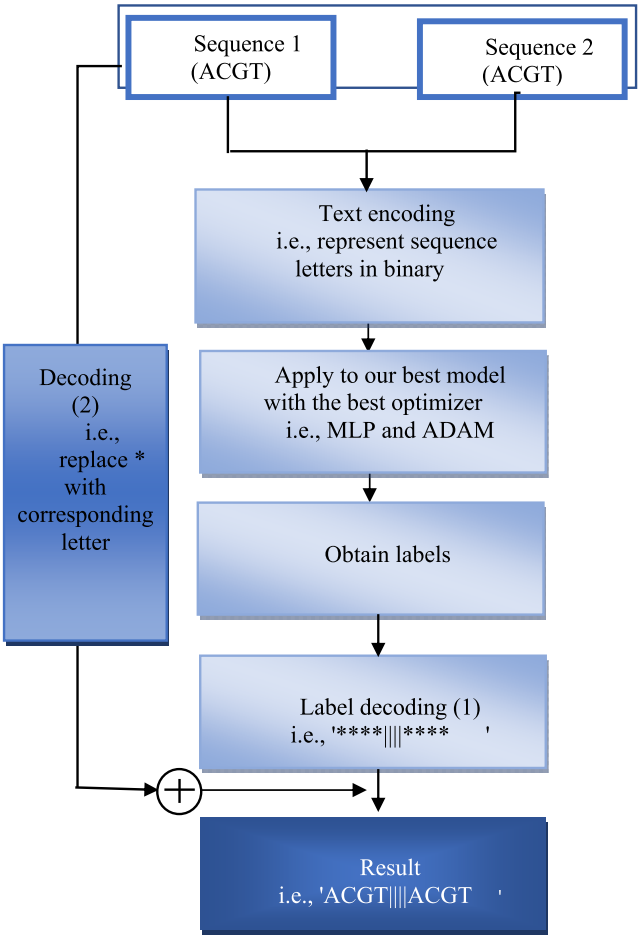


FIGURE 5. Strategy for testing new data.

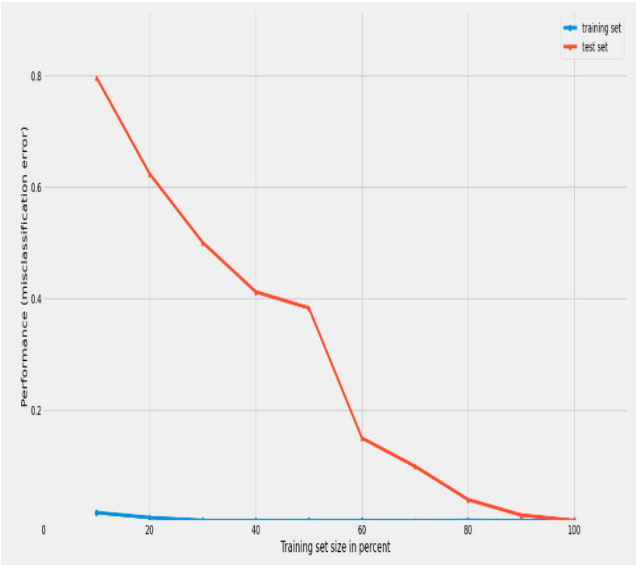


FIGURE 6. Training and testing learning curves exhibiting a good fit.

each sequence [37]. We set a fixed length for both sequences that is equal to 4.1 M NTs during our testing phase. We refer to it as Dataset 4.1 in this study. Figure 6 illustrates that the training and testing learning curves exhibit a good fit.

The evaluations of the DNA sequence databases show that our MLP method achieves an overall performance of 2912 GCUPS for the NW algorithm and runs at a speed equivalent to 6.0417 s when aligning two sequences with a length of 4.1 M NTs by using a single-core platform, as indicated in Table (21). This table provides the prediction runtime and GCUPS for global alignment when using our best model. Notably, decoding time is not included in our calculations of prediction runtime. Performance comparison with other state-of-the-art techniques is provided in Table (22).

In [1], the authors used ML and a deep learning model to implement DNA sequence alignment by applying the NW algorithm with approximately the same technique used in the current work. However, they adopted the 80/20 ratio for the training and test datasets and applied 10 ML classifiers to 4 datasets. They achieved the highest accuracy of 82.59% with an SVM classifier and obtained an accuracy of 98.37% when using a CNN model. However, the latter is time-consuming because it takes 6.5 s to align two sequences with a length of 5000 NT on GPU. The model presented in the current work uses 10-fold cross-validation for all the datasets (six datasets) and achieves an accuracy of 99.27% for the training dataset and 86% for the test dataset by using MLP with the ADAM optimizer before implementing overfitting prevention techniques. In addition, our model takes only 6 s to align two real DNA sequences with a length of 4.1 M NTs, achieving 99.7% accuracy after using overfitting prevention techniques.

VII. CONCLUSION

Most previous studies have focused on accelerating alignment algorithms by using different methods without providing any effective solution to sequential process problems. Our proposed algorithm depends on the parallelization of common alignment algorithms (i.e., NW for global alignment) for DNA or RNA sequences under certain constraints or limitations to address the major problems of DP, memory usage, computational time (i.e., O(N/4)), calculation steps (i.e., O(N/4)), and large sequences (i.e., the big data problem). The proposed method takes O(N/4) calculation steps, where N is the length of each sequence with a minimum value of four (i.e., N = 4, 8, 12, . . .).

It can also be applied to RNA by replacing the NT represented by T with another NT represented by U. This technique can be extended to any global alignment method and extremely long sequences. Our proposed algorithm accelerates large-scale sequence alignment tasks with ML algorithms and allows researchers to solve real-world problems that biologists are currently confronting.

Our proposed algorithm (i.e., MLP with the ADAM optimizer) achieves 99.70% accuracy after applying techniques for preventing overfitting to real DNA sequences with a length of 4.1 M NTs. Moreover, it takes approximately 6 s to align two sequences with the same length with 2912 GCUPS.

VIII. FUTURE WORK

Feature optimization techniques and algorithms, such as GA, the gray wolf algorithm, whale optimization, and the particle swarm optimization (PSO) algorithm can be considered in the future [39].

Moreover, testing other ensemble methods, such as a voting classifier (hard and soft) or stacking, can enhance model accuracy. Training MLP by using PSO [40] or auto ML libraries (e.g., TPOT, Auto-Keras, and H2O) can improve model performance [41].

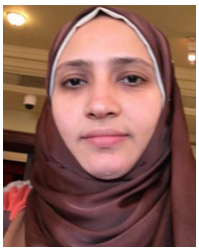
In addition, enhancing running time prediction is feasible by using CPython; PyPy, which is an alternative to CPython based on just-in-time compilation [42]; Microsoft's HummingBird-ML, which converts trained ML models into tensor computations (e.g., PyTorch) [43]; and Numba, which is a high-performance and open-source Python compiler that can reach the speeds of C and Fortran when compiling numerical algorithms [44]. Lastly, implementing MLP by using hardware platforms, such as FPGA, GPU [45], and quantum machine learning (e.g., Qiskit library) [4 6], can improve prediction runtime.

REFERENCES

[1] A. E. E. D. Rashed, M. Obaya, H. El, and D. Moustafa, "Accelerating DNA pairwise sequence alignment using FPGA and a customized convolutional neural network," *Comput. Elect. Eng.*, vol. 92, 2021, Art. no. 107112.
[2] M. Mathur and Geetika, "Multiple sequence alignment using MATLAB," *Int. J. Inf. Comput. Technol.*, vol. 3, no. 6, pp. 450–497, 2013.
[3] A. Travers and G. Muskhelishvili, "DNA structure and function," *FEBS J.*, vol. 282, no. 12, pp. 2279–2295, 2015.
[4] J. Gorodkin and W. L. Ruzzo, Eds., *RNA Sequence, Structure, and Function: Computational and Bioinformatic Methods*. New York, NY, USA: Humana Press, 2014.
[5] W. Haque, A. Aravind, and B. Reddy, "Pairwise sequence alignment algorithms: A survey," in *Proc. Conf. Inf. Sci., Technol. Appl.*, Mar. 2009, pp. 96–103.
[6] M. I. Irawan, I. Mukhlash, A. Rizky, and A. R. Dewi, "Application of Needleman–Wunch algorithm to identify mutation in DNA sequences of corona virus," *J. Phys., Conf. Ser.*, vol. 1218, no. 1, May 2019, Art. no. 012031.
[7] B. Strengholt and M. Brobbel, "Acceleration of the Smith-Waterman algorithm for DNA sequence alignment using an FPGA platform," M.S. thesis, Dept. Comput. Eng., Univ. Technol. Delft, Delft, The Netherlands, Jun. 2013.
[8] X. Xu, Y. Chan, K. Xu, J. Zhang, X. Wang, Z. Yin, and W. Liu, "SLPal: Accelerating long sequence alignment on many-core and multi-core architectures," in *Proc. IEEE Int. Conf. Bioinf. Biomed. (BIBM)*, Dec. 2020, pp. 2242–2249.
[9] A. E. Rashed, "DCT vs. DWT based license plate detection," *Int. J. Imag. Robot.*, vol. 13, no. 2, pp. 148–165, 2014.
[10] A. Subasi and E. Erçelebi, "Classification of EEG signals using neural network and logistic regression," *Comput. Methods Programs Biomed.*, vol. 78, no. 2, pp. 87–99, May 2005.
[11] S. F. Abbasi, J. Ahmad, A. Tahir, M. Awais, C. Chen, M. Irfan, H. A. Siddiqa, A. B. Waqas, X. Long, B. Yin, S. Akbarzadeh, C. Lu, L. Wang, and W. Chen, "EEG-based neonatal sleep-wake classification using multilayer perceptron neural network," *IEEE Access*, vol. 8, pp. 183025–183034, 2020.
[12] Accessed: Jun. 5, 2020. [Online]. Available: https://scikitlearn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html
[13] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980
[14] S. S. Dhaliwal, A. A. Nahid, and R. Abbas, "Effective intrusion detection system using XGBoost," *Information*, vol. 9, no. 7, p. 149, 2018.
[15] A. Ogunleye and Q.-G. Wang, "XGBoost model for chronic kidney disease diagnosis," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 17, no. 6, pp. 2131–2140, Nov. 2020.
[16] H. Khaled, R. El Gohary, N. L. Badr, and H. M. Faheem, "Accelerating pairwise DNA sequence alignment using the CUDA compatible GPU," *Int. J. Comput. Appl.*, vol. 84, no. 1, pp. 25–31, Dec. 2013.
[17] C. Ling, K. Benkrid, and T. Hamada, "A parameterisable and scalable Smith–Waterman algorithm implementation on CUDA-compatible GPUs," in *Proc. IEEE 7th Symp. Appl. Specific Processors*, Jul. 2009, pp. 94–100.
[18] A. Chaudhary, D. Kagathara, and V. Patel, "A GPU based implementation of Needleman–Wunsch algorithm using skewing transformation," in *Proc. 8th Int. Conf. Contemp. Comput. (IC)*, Aug. 2015, pp. 498–502.
[19] H. Khaled, G. R. El, N. L. Badr, and H. M. Faheem, "Hybrid framework for pairwise DNA sequence alignment using the CUDA compatible GPU," in *Proc. Int. Conf. Bioinf. Comput. Biol. (BIOCOMP)*, 2013, p. 1.
[20] Y. S. Lee, Y. S. Kim, and R. L. Uy, "Serial and parallel implementation of Needleman–Wunsch algorithm," *Int. J. Adv. Intell. Inform.*, vol. 6, no. 1, pp. 97–108, 2020.
[21] B. Chen, Y. Xu, J. Yang, and H. Jiang, "A new parallel method of Smith–Waterman algorithm on a heterogeneous platform," in *Proc. Int. Conf. Algorithms Archit. Parallel Process.*, in Lecture Notes in Computer Science, vol. 6081. Berlin, Germany: Springer-Verlag, 2010, pp. 79–90.
[22] Y. Jararweh, M. Al-Ayyoub, M. Fakirah, L. Alawneh, and B. B. Gupta, "Improving the performance of the Needleman–Wunsch algorithm using parallelization and vectorization techniques," *Multimedia Tools Appl.*, vol. 78, no. 4, pp. 3961–3977, Feb. 2019.
[23] V. Gancheva and I. Georgiev, "Multithreaded parallel sequence alignment based on Needleman–Wunsch algorithm," in *Proc. IEEE 19th Int. Conf. Bioinf. Bioeng. (BIBE)*, Oct. 2019, pp. 165–169.
[24] L. Ji, X. Pu, H. Qu, and G. Liu, "One-dimensional pairwise CNN for the global alignment of two DNA sequences," *Neurocomputing*, vol. 149, pp. 505–514, Feb. 2015.
[25] S. Henikoff and J. G. Henikoff, "Amino acid substitution matrices from protein blocks," *Proc. Nat. Acad. Sci. USA*, vol. 89, no. 22, pp. 10915–10919, 1992.
[26] J. Brownlee, *Data Preparation for Machine Learning: Data Cleaning, Feature Selection, and Data Transforms in Python*. Machine Learning Mastery, 2020. [Online]. Available: https://books.google.com.eg/books?id=uAPuDwAAQBAJ&lpg=PP1&ots=Cl2KtgeNuW&lr&pg=PP1#v=onepage&q&f=false

[27] P. K. Janert, *Data Analysis With Open Source Tools: A Hands-On Guide for Programmers and Data Scientists*. Sebastopol, CA, USA: O'Reilly Media, 2010.
[28] D. Conway and J. White, *Machine Learning for Hackers*. Sebastopol, CA, USA: O'Reilly Media, 2012.
[29] J. Miao and L. Niu, "A survey on feature selection," *Procedia Comput. Sci.*, vol. 91, pp. 919–926, Jan. 2016.
[30] R. Zhang, F. Nie, X. Li, and X. Wei, "Feature selection with multi-view data: A survey," *Inf. Fusion*, vol. 50, pp. 158–167, Oct. 2019.
[31] J. Wu and C. Hicks, "Breast cancer type classification using machine learning," *J. Personalized Med.*, vol. 11, no. 2, p. 61, Jan. 2021.
[32] *Tour of Evaluation Metrics for Imbalanced Classification*. Accessed: Feb. 25, 2021. [Online]. Available: https://machinelearningmastery.com/tour-of-evaluation-metrics-for-imbalanced-classification/
[33] R. G. Mantovani, T. Horváth, R. Cerri, J. Vanschoren, and A. C. P. L. F. de Carvalho, "Hyper-parameter tuning of a decision tree induction algorithm," in *Proc. 5th Brazilian Conf. Intell. Syst. (BRACIS)*, Oct. 2016, pp. 37–42.
[34] P. Liashchynskyi and P. Liashchynskyi, "Grid search, random search, genetic algorithm: A big comparison for NAS," 2019, *arXiv:1912.06059*. [Online]. Available: http://arxiv.org/abs/1912.06059
[35] W. E. I. Jiang and S. Siddiqui, "Hyper-parameter optimization for support vector machines using stochastic gradient descent and dual coordinate descent," *EURO J. Comput. Optim.*, vol. 8, no. 1, pp. 85–101, Mar. 2020.
[36] X. Ying, "An overview of overfitting and its solutions," *J. Phys., Conf. Ser.*, vol. 1168, no. 2, Feb. 2019, Art. no. 022022.
[37] *SWAPHI-LS: Alignment on Xeon Phi Cluster: Smith-Waterman Long DNA Sequence Alignment on Xeon Phi Clusters*. Accessed: Jan. 10, 2021. [Online]. Available: https://sourceforge.net/projects/swaphi-ls/files/data/
[38] Y. Liu, T.-T. Tran, F. Lauenroth, and B. Schmidt, "SWAPHI-LS: Smith–Waterman algorithm on Xeon Phi coprocessors for long DNA sequences," in *Proc. IEEE Int. Conf. Cluster Comput. (CLUSTER)*, Sep. 2014, pp. 257–265.
[39] Y. Chen, F. He, H. Li, D. Zhang, and Y. Wu, "A full migration BBO algorithm with enhanced population quality bounds for multimodal biomedical image registration," *Appl. Soft Comput.*, vol. 93, Aug. 2020, Art. no. 106335.
[40] M. Sheikhan and N. Mohammadi, "Time series prediction using PSO-optimized neural network and hybrid feature selection algorithm for IEEE load data," *Neural Comput. Appl.*, vol. 23, nos. 3–4, pp. 1185–1194, Sep. 2013.
[41] Accessed: Jun. 5, 2020. [Online]. Available: https://medium.com/swlh/8-automl-libraries-to-automate-machine-learning-pipeline-3da0af08f636
[42] Accessed: Jun. 5, 2020. [Online]. Available: https://towardsdatascience.com/getting-started-with-pypy-ef4ba5cb431c
[43] Accessed: Jun. 5, 2020. [Online]. Available: https://pypi.org/project/hummingbird-ml/
[44] Accessed: Jun. 5, 2020. [Online]. Available: https://numba.pydata.org/
[45] N. Hou, F. He, Y. Zhou, and Y. Chen, "An efficient GPU-based parallel tabu search algorithm for hardware/software co-design," *Frontiers Comput. Sci.*, vol. 14, no. 5, pp. 1–18, 2020.
[46] Accessed: Jun. 5, 2020. [Online]. Available: https://qiskit.org/

AMR EZZ EL-DIN RASHED received the M.Sc. degree from Mansoura University, Egypt, in 2010, where he is currently pursuing the Ph.D. degree with the Electronics and Communications Engineering Department, Faculty of Engineering. He is currently a Lecturer with the Computer Engineering Department, Faculty of Computers and Information Technology, Taif University, Saudi Arabia. His main research interests include bioinformatics, biomedical image processing, speaker recognition, computer vision, machine learning, deep learning applications, and embedded systems, including FPGA and VHDL.



HANAN M. AMER received the Ph.D. degree from Mansoura University, Mansoura, Egypt, in 2018. She is currently an Assistant Professor with the Electronics and Communications Engineering Department, Faculty of Engineering, Mansoura University. She has authored or coauthored more than six technical articles. Her research interests include signal and image processing, medical image analysis, and lung cancer.

MERVAT EL-SEDDEK (Member, IEEE) is currently an Assistant Professor with the Department of Electronics and Communications Engineering, Higher Institute of Engineering and Technology, Mansoura. Her research interests include image processing and machine learning.

HOSSAM EL-DIN MOUSTAFA is currently an Associate Professor with the Department of Electronics and Communications Engineering and the Founder and an Executive Manager of the Biomedical Engineering Program (BME), Faculty of Engineering, Mansoura University. His main research interests include biomedical image and signal processing and deep learning applications.