Journal of Current Science and Technology, September-December 2022 Vol. 12 No. 3, pp. 462-481
Copyright ©2018-2022, Rangsit University ISSN 2630-0656 (Online)

Cite this article: Kaur, K., Sagar, A. K., & Chakraborty, S. (2022, September). Accelerating the performance of sequence alignment using machine learning with RAPIDS enabled GPU. *Journal of Current Science and Technology, 12*(3), 462-481. DOI:

Journal of Current Science and Technology
Journal homepage: https://jcst.rsu.ac.th

Accelerating the performance of sequence alignment using machine learning with RAPIDS enabled GPU

Karamjeet Kaur[1*], Anil Kumar Sagar[1], and Sudeshna Chakraborty[2]

[1]School of Engineering and Technology, Sharda University, Greater Noida, U.P., 201306, India
[1a]E-mail: karam_7378@yahoo.com; [1b]E-mail: aksagar22@gmail.com
[2]Galgotias University, Greater Noida, U.P., 203201, India
E-mail: sudeshna2529@gmail.com

*Corresponding author: E-mail: karam_7378@yahoo.com

Received 19 April 2022; Revised 3 June 2022; Accepted 12 June 2022;
Published online 26 December 2022

Abstract

In bioinformatics, sequence alignment is a useful way of identifying similarities in the DNA sequences to identify common evolutionary and structural relationships. Currently, with the number of sequences increasing in sequence databases, traditional methods take too much time to align two or more sequences simultaneously, because these methods are sequentially-based. Even when sequential algorithms are modified so that alignment can be done parallelly, this modification is unable to reduce the time proportionally, as the number of sequences in databases is increasing at an exponential rate. This limitation can be overcome by machine learning if sequences are treated as big data, and knowledge from such large-scale data can be gained. If machine learning techniques are combined with the capabilities of GPUs, processing time is reduced due to the parallel architecture of GPUs. Thus, a GPU-based approach is proposed to accelerate multiple sequence alignment, yielding significant accuracy improvement. An efficient model is proposed and implemented to predict classes of biological sequences. Many challenges are overcome by applying pre-processing to sequence data, which is necessary for machine learning techniques to work. This model uses the embedding method for the representation of DNA sequences and combines the capabilities of GPUs with a random forest algorithm. Results show that the model yields a high accuracy of 99.5%, with a reduction in time required to align sequences. Compared with other CPU-based methods, this GPU-based model takes less computation time. Fast and accurate alignment is vital in evolutionary studies, which can help in designing new drugs or modifying existing drugs for new diseases.

Keywords: *deep learning; graphical processing unit; machine learning; multiple sequence alignment; random forest.*

1. Introduction

Sequence alignment is used to identify functional, evolutionary, and structural similarities in amino acid (proteins) and nucleic acid (DNA and RNA) sequences, both important factors in understanding organisms. Proteins represent the building blocks of living organisms, while DNA stores the genetic information. DNA sequences are combination of four characters – A, C, G, T and comprise variable lengths, ranging from a few dozen characters to hundreds of megabytes. Figure 1 provides DNA sequence example. The sequence alignment process compares two or more DNA sequences to find similarities between unknown sequences and known sequences in a database.

Many DNA sequences are available in biological data centres like the National Centre for Biotechnology Information (NCBI). To explore evolutionary similarities, which inform drug design and studies of human genetics, cancer, epidemiology, and biodiversity, understanding sequence alignment and its analysis is prerequisite. Such understanding also underlies the molecular and structural subareas of biology, helping explain both the history, functional and structural roles of protein and amino acid sequences. It finds similar parts of two or more sequences, which are considered homologous if they share a similarity. Depending on sequence length, alignment can be either local or global. Local sequence alignment finds a region of higher similarity, while global sequence alignment aligns sequences along their entire length to find as many matching characters as possible. Alignment is also classified by the number of sequences involved. Pairwise alignment aligns two sequences to find the best region of similarity and multiple sequence alignment (MSA) aligns more than two sequences, providing more biological information than pairwise alignment.

```
Sequence 1: TACTAGTCCATCGTAACTTGACTACTGAGA - - - - -
Sequence 2: TCATAGTCCATAATAACGATTACACTGAGA - - - - -
Sequence 3: AACTAGTCCATTATAACTGATGTACTGAGA - - - - -
|        |         |         |         |         |        |
|        |         |         |         |         |        |
|        |         |         |         |         |        |
Sequence N: TCATAGTCCATCGCTAACATGACACTGAGA - - - - -
```

Figure 1 Multiple DNA biological sequences with nucleotide residues A, C, G and T.

1.1 Motivation and requirements

As the size of sequence data increases, so alignment takes more time. Last year alone, the GenBank division of the NCBI added 11 million new sequences to the existing database, which already contained 106 billion nucleotide bases and 108 million individual sequences. In April 2021, GenBank had 832,400,799,511 bases and 227,123,201 sequences. With new sequences added at such a high rate, finding functional, evolutionary and structural relationships requires too much search time. Computation grows quadratically as the number of sequences to align increases. Thus, execution speed must be improved so that evolutionary relationships can be discovered and gene families of new members predicted more quickly. The existing CPU-based programming approach for dealing with such problems is computationally expensive. Furthermore, reducing search time in the database requires hardware-based acceleration. Dealing with large amounts of data is the biggest challenge for traditional CPU-based methods. Sequence alignment results should be fast and accurate to help scientists analyse protein functions, develop drugs, prescribe personalised medicine, compare studies, provide gene therapy treatment, and develop tools to find similarities between new sequences and existing sequences in the database.

The core of this research exists within the bioinformatics field, in which computing techniques are used to analyse and interpret biological data. Primary contributions of this research include:

- If sequences are aligned and classified accurately, it is possible to develop more efficient drugs.
- Individuals can receive personalised medicine and genetic treatments.
- Diseases can be classified as pre-existing or novel. Pre-existing diseases require identifying which drugs are suitable for treatment; novel diseases require the development of new drugs.
- Much faster computation is required compared to the sequential approach.
- A better strategy than single-threaded CPU is needed.
- Machine learning analyses huge volumes of data and makes data-driven recommendations and decisions.
- Machine learning algorithms learn automatically without the need for explicit programming, performing complex predictions on huge datasets.
- GPUs are faster than CPUs due to parallel architecture and are perfect for deep learning.
- Multiple cores in GPUs split tasks into multiple sub-tasks and run them simultaneously.
- GPUs perform better on large tasks versus small tasks.

- GPUs are CUDA language-enabled, utilising the power of machine learning with parallel computing.
- If a GPU fully utilises its multiple cores and CUDA-enabled libraries of machine learning, the problem of accelerating sequence alignment can be solved with more accuracy.

1.2. Literature review

In multiple sequence alignment, progressive, computation-intensive methods (Naznin, Sarker, & Essam, 2012) are used for large-sized sequences, but when a single sequence needs to be compared against the whole database, heuristic-based methods are used. MSA can be obtained through several methods, such as progressive, dynamic programming, genetic and greedy algorithms. Currently, progressive MSA methods are the most popular. With these, two sequences are initially aligned by pairwise alignment, after which the third sequence is aligned to the first alignment; this process is repeated to align all the sequences. This type of alignment has limitations, however, as errors occurring at the initial stage are propagated to the final stage, decreasing alignment accuracy.

Heuristic methods (Sun, Palade, Wu, & Fang, 2014), compared to dynamic programming methods, are used to maximise similarity, but an optimal alignment cannot be obtained. When global pairwise alignment is combined with the tree-based progressive method, it provides a basis for Clustal Omega, ProbCons, and T-Coffee methods. These methods, when combined with iterative strategy, result in new methods like Muscle and MAFFT (Zhu, Li, Salah, Shi, & Li, 2015). These limitations are overcome by combining the heuristics method, dynamic programming, and parallel programming.

Another approach to achieving better alignment in less time involves combining the CPU with a graphical processing unit (GPU) for searching protein sequences, using multiple cores to parallelise the alignment process according to the architecture of the GPU. This paper optimised the basic local alignment search tool, in which the time-consuming searching phase is mapped to many threads according to the architecture of the GPU. Several operations related to bioinformatics, image processing, modelling, scientific computing, game development, robot motion planning, computational geometry, collision detection, and many numerical applications are performed by GPUs nowadays.

The GPU-based Smith-Waterman (SW) algorithm is used to search sequences from biological databases. The SW alignment algorithm (Liu, Hong, Lin, & Hung, 2015) is modified by inter-task parallelisation, which exploits the architecture of GPUs via the CUDA programming language supported by the NVIDIA Corporation. This corporation manufactures different high-end GPUs. The CPU-GPU-based parallelisation process aligns two sequences at a reduced time and gives better results than the CPU-based SW algorithm, but due to the limited internal memory of GPUs, memory bandwidth is a major bottleneck; this issue is overcome when extended for multiple sequence alignment by computing batches of alignment in parallel. The CPU-based sequential alignment algorithm is using dynamic programming approach and is reformulated in such a way that the alignment matrix is calculated in parallel by a GPU to speed up the task of alignment. Hence, the alignment score and traceback step of the SW algorithm are parallelised.

The dynamic programming-based Needleman-Wunsch algorithm is further reformulated (Warris, Yalcin, Jackson, & Nap, 2015) by using a GPU as a hardware accelerator. If alignment is performed by the MAFFT tool and the architecture is sequential, runtime is long. The MAFFT algorithm is accelerated for organising the sequence data via CUDA-enabled GPUs. It uses modified run-length encoding to allocate memory and the shared memory of the GPU to speed up alignment. This CUDA-based implementation improves execution time compared to CPU-based implementation. A parallel MSA algorithm (Andalon-Garcia, & Chavoya, 2017) that does not use the progressive method was also developed for global alignment. This algorithm generates alignment by finding the longest common subsequence.

For searching MSA, several processes run independently to reduce computation time but still have less accuracy. Computing exact MSA is time-consuming due to increasing sequence data, which presents a major challenge nowadays. If this sequence data were treated as big data, machine learning could help in the classification and alignment of DNA/protein sequences. Researchers design a necessary drug after identifying the function of the protein. Similarly, metric learning from biological data can be done by machine learning (Min, Lee, & Yoon, 2017), in which a framework is designed to perform alignment in feature vector space and further perform classification.

The main challenge facing computational biology is the transformation of heterogeneous data. Once such data is transformed, biological knowledge can help in constructing a model and obtaining a prediction. Machine learning methods are applied to extract knowledge from biological data. To provide a comprehensive perspective, the bioinformatics domain incorporates deep learning, which produces good results by selecting appropriate architecture. When deep learning is applied to big sequence data, alongside the help of parallel and distributed computing, it can be used in processing images and text.

Deep learning automatically generates high-level features with the help of machine learning (Ravi et al., 2017) and is useful for bioinformatics, medical imaging, and public health. In bioinformatics, neural networks are capable of memorising sequential data to analyse the stream of data, although previous computations are needed to obtain such output as the analysis of DNA sequences. When combined with GPUs, which are designed for parallel processing, deep neural networks can scale large datasets and reduce complexity.

When machine learning is applied to big data (L'heureux, Grolinger, Elyamany, & Capretz, 2017) then performance of the algorithm depends upon the machine learning architecture chosen to store large-sized data. Thus, it is necessary to rethink typical architecture, because with increasing data size, algorithms can sometimes become unstable and lead to the curse of dimensionality. This huge challenge of dealing with complex biological data can be overcome by artificial neural networks and reinforcement learning (Mahmud, Kaiser, Hussain, & Vassanelli, 2018). Such machine learning techniques can find complex patterns in biological data. A novel method (Jiang, Ganesan, & Yao, 2018) for accelerating the hidden Markov model was proposed, utilising GPU hardware resources and improving performance and scalability regardless of sequence datasets and query models. Newly introduced GPUs have more computing power in terms of hardware resources and advanced features capable of handling the increasing size of data for industry and the medical field. CUDA-enabled NVIDIA GPUs provide a development environment to the user which, if the user uses it appropriately, helps in parallelising the task related to biological data.

In bioinformatics, to manage complex workflow, a GPU-accelerated application for enhancing performance was designed (Welivita, Perera, Meedeniya, Wickramarachchi, & Mallawaarachchi, 2018) on an Amazon cloud platform. This type of cloud-based computing provides three levels of parallelism that decrease execution time. It has been noted that while deep learning can handle big data and is used in bioinformatics (Li et al., 2019), it faces problems when handling sequence data that lead to overfitting and interpretability. Performance is further enhanced by including ensemble learning and SVM (Mirzaei, Sidi, Keasar, & Crivelli, 2019; Zhu, Wang, Li, Zhu, & Du, 2020), which also deal with imbalanced datasets.

Recently, the open-source, machine learning-based benchmark iMLBench (Zhang et al., 2020) was developed by integrating CPU and GPU architecture with shared unified memory, in which CPUs and GPUs run together and eliminate unnecessary overhead by machine learning tasks. A widely used Needleman-Wunch algorithm is optimised (Rashed, Amer, El-Seddek, & Moustafa, 2021) by using a multilayer perceptron for aligning DNA/RNA sequences. This study used a divide-and-conquer strategy to parallelise computation steps for aligning long sequences.

Deep learning models are increasingly used to analyse sequences. For example, CNN with LSTM and bidirectional LSTM are used (Gunasekaran et al., 2021) to improve accuracy. Since a huge amount of sequence data is available today, deep learning architecture, when used with ANN by setting hyperparameters, proves helpful in sequence classification (Alhalem et. al., 2020). Sequence classification can be done by combining machine learning with statistical classification techniques like CRT, QUEST, CHAID and C5.0 (Gupta, Bihari, & Tripathi, 2019), yielding better results for imbalanced datasets.

When identifying functionally similar proteins and evolutionary pathways, protein-protein interactions help in alignment. A representation learning method combining topological features and biological characteristics is used for better alignment results (Gao et al., 2019). The structural features of sequences are transformed into lowdimensional vectors that help determine similarities for multiple sequences.

DNA sequences can be clustered using unsupervised deep learning, in which sequence alignment is unnecessary. Frequency chaos game representation is used (Millán Arias, Alipour, Hill, & Kari, 2022) to self-learn data patterns, generating mimics by optimising multiple neural networks. The accuracy varies from 77% to 100% by majority voting scheme in finding the final assignment of a cluster. This suggested method outperforms K-means and Gaussian mixture clustering, which completely ignores sequence homology, length of sequences, and taxonomic identifiers.

Previous literature suggests that sequence alignment should be accurate when dealing with large sequence data. When large-sized protein families are searched to find similarities, acceleration of this task is necessary. Machine learning helps in alignment and analysis tasks, but machine learning techniques cannot be applied directly to biological sequences, otherwise the dimensionality of data would be increased and acceleration of the classification task would not be possible. To reduce execution time, machine learning with hardware accelerators like GPUs, along with some specific encoding method at the time of pre-processing, is required.

2. Objectives

To overcome the above challenges, this paper has the following objectives:

- Large DNA sequences are collected and pre-processing steps applied to transform character-type data to vector form and make it suitable for machine learning methods. This transformation is the biggest challenge to align sequences, which otherwise increases the dimensionality of the data which in turn increases the time and memory space taken by the alignment process.
- Natural language processing maintains the positional information of each sequence, and a suitable classifier is applied to predict the similarity of DNA sequences with existing sequences.
- Machine learning techniques are integrated into the bioinformatics domain.
- Parallelisation is applied in such a way that all cores of the GPU are fully utilised.
- Given all these points, this machine learning model yields high accuracy with valid output and accelerates the performance of the sequence alignment process, taking less CPU time.

3. Methodology

Machine learning handles automatic learning from numeric data without the need for explicit programming and is thus widely used in generating valuable information from data. The input to our model comprises protein sequences, also called features. Machine learning techniques identify suitable features that allow the model to differentiate one type of data from another type to solve the classification problem. A group of features for one protein sequence is called a feature vector and n-dimensional space is called feature space.

Machine learning can process large data by considering DNA sequences as big data, yet machine learning methods face challenges due to the mismatch in the type of data accepted by machine learning versus the DNA data available in biological databases, as it is difficult to integrate different types of datasets. Machine learning techniques cannot be applied directly to DNA sequences, and machine learning models cannot be trained on the variety of data and are thus unable to predict the output. Machine learning techniques extract features, but DNA sequences do not have clear features that generate high-dimensional data. High dimensionality is also overcome (Gunasekaran et al., 2021) when CNN is combined with LSTM and bi-directional LSTM.

There is a need to increase efficiency when large DNA sequences are processed and ensure that appropriate similarity measures are considered when measuring similarity among sequences. As DNA sequences comprise variable lengths, it is prerequisite to convert the sequences into vectors by preserving their sequence order and key patterns.

Figure 2 represents the methodology followed to design this model. Different pre-processing steps are performed by the CPU. Pre-processing helps prepare and transform DNA sequence data into numeric data so that the machine learning model can be trained and can predict the output accurately. To accelerate sequence classification performance, the machine learning model is trained and tested using a GPU. The GPU has thousands of cores that provide a parallel computing platform, cores that are dedicated to a single task. Training the deep learning model requires hardware resources, and the GPU can handle many computations on a huge amount of data simultaneously. The sequential task is performed by the CPU while the parallel task is performed by the GPU. Pre-processing is a sequential task required for data preparation and transformation. Our model uses natural language processing (NLP) for DNA sequence coding, sequence padding to make the size of each sequence equal, and an ensemble-based random forest classifier to avoid overfitting and underfitting.

3.1 Natural language processing

NLP techniques are well suited for text-based features, which can be converted into a numeric format. Sequence-to-vector encoding uses a natural language processing approach in which amino acids are represented as words and protein sequences as sentences. Our model uses word embedding and can transform individual fixed-sized words into real-valued vectors. The NLP task can be performed with deep learning through word embedding, which helps in solving a classification problem. The text should be prepared in such a way that each word is one-hot encoded. Keras supports embedding layers for text data and has a one_hot function for integer encoding. Since sequences are of variable length, the requirement for Keras is that all vectorised input comprises the same length; thus, Keras has another function named pad_sequence for meeting this requirement. After encoding, these fixed-length numeric values are used as input for classification. For classification, an ensemble learning-based random forest is used to predict the exact class as an output.

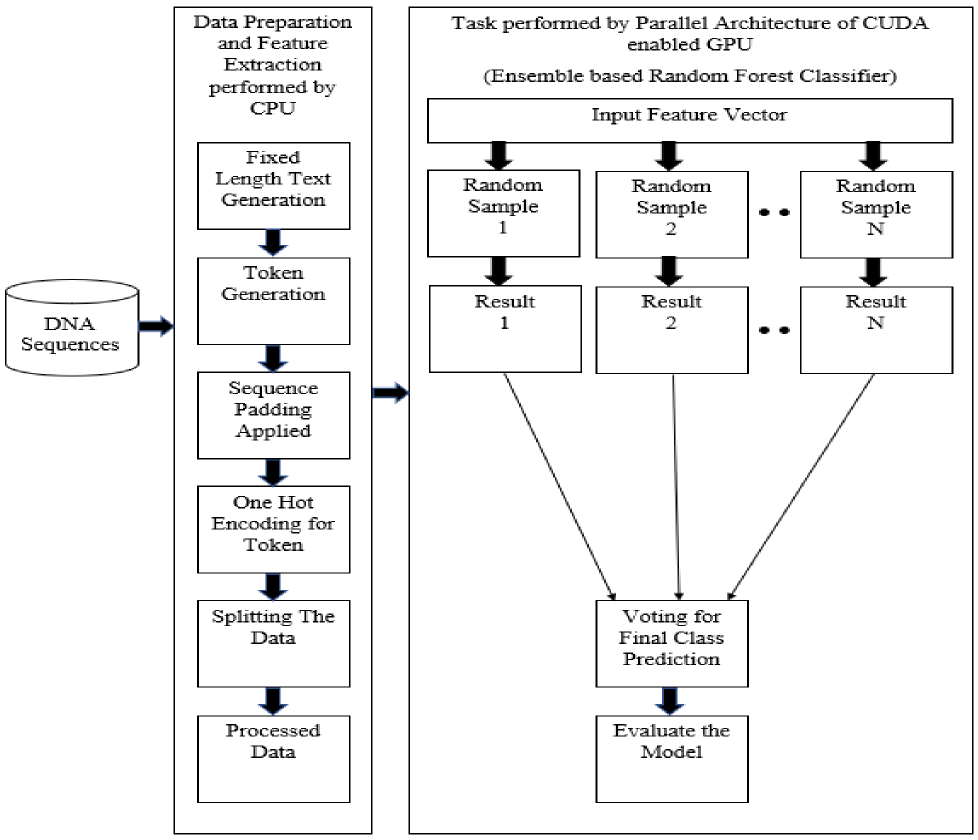
Data Preparation and Feature Extraction performed by CPU
Fixed Length Text Generation
Token Generation
Sequence Padding Applied
One Hot Encoding for Token
Splitting The Data
Processed Data
DNA Sequences
Task performed by Parallel Architecture of CUDA enabled GPU
(Ensemble based Random Forest Classifier)
Input Feature Vector
Random Sample 1
Random Sample 2
Random Sample N
Result 1
Result 2
Result N
Voting for Final Class Prediction
Evaluate the Model

Figure 2 Methodology for designing a GPU-based machine learning model

3.2 DNA sequence coding

DNA sequences are combination of four characters – A, C, G and T with no spaces in between and cannot be considered text data or numeric data. However, to develop a suitable model, machine learning techniques work on numeric data in the form of a matrix; the sequence data thus requires pre-processing to make it suitable for the training model. There are three ways to encode DNA sequences: sequential, one-hot, and k-mer encoding. The designed model accepts DNA sequences as input, but sequences should be encoded as numeric values before the classification process used by the machine learning algorithm. These numeric values are represented by 2-dimensional matrices. When DNA sequences are transformed into matrix form and fed as input to the designed model, the model then classifies the DNA sequences.

This model uses k-mer encoding and one-hot encoding. At first, k-mer encoding is applied to divide large-sized sequences into k-length overlapping segments. If the length represented by k is 5, then the biological sequence ‘AGCTGCATGTC’ is decomposed into seven k-mers: ‘AGCTG’, ‘GCTGC’, ‘CTGCA’, ‘TGCAT’, ‘GCATG’, ‘CATGT’, and ‘ATGTC’. In the second step, one-hot encoding characters [A, G, T, C] are coded as vectors [0,0,0,1], [0,0,1,0], [0,1,0,0] and [1,0,0,0]. These are most commonly used in deep learning because of its consistent performance in different datasets.

3.3 Sequence padding

The deep learning model works on the same shape input, but DNA sequences comprise different lengths. To make each sequence of equal length, a zero-padding technique (Lopez-del Rio, Martin, Perera-Lluna, & Saidi, 2020) is used, in which zeroes are appended at the end of the sequence so that all sequences become equal in length, helping enhance the performance of the model. This model uses pre-padding (shown in Figure 3), adding zeroes at the start of the sequence after all sequences are encoded. This process is part of the pre-processing phase before training the model.

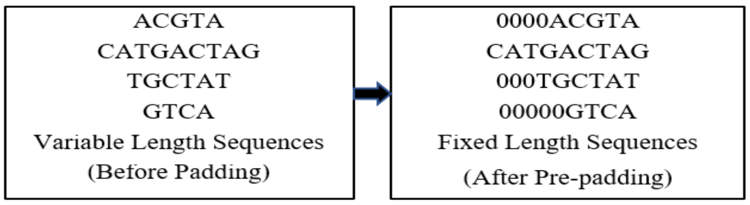

Figure 3 Process of pre-padding to make all sequences of equal length

3.4 Sequence classification using machine learning

When constructing a computation model, feature extraction is very important for predicting the classification for DNA sequence analysis. DNA sequence analysis problems can be solved by classification. If there are *n* DNA sequences (D1, D2, D3 …. Dn) that belong to *m* categories, then the purpose of designing a classifier is to predict the unseen sample’s label, a prediction based on labelled samples upon which this classifier is trained. There are many classifiers for predicting the output, such as support vector machine (Rangwala, & Karypis, 2005), random forest (Liu, Long, & Chou, 2016), and k-nearest neighbors (Chou, & Shen, 2006). Among these algorithms, random forest solves tasks related to bioinformatics. Random forest is a tree-structured classifier that uses random feature selection (Jiang et al., 2007) and overcomes the overfitting problem (Chen, Wang, & Zhang, 2011). Further studies show (Liu, Wang, Dong, Li, & Liu, 2016; Liu, Yang, & Chou, 2017) that if an ensemble predictor is used for classification along with random forest, better performance can be achieved, because this type of predictor uses a voting strategy by combining an array of the individual predictors.

In mining tasks, machine learning helps classify DNA sequences. The classification model learns from training samples and then predicts the class of unknown DNA sequences. DNA sequences are a special type of data with non-numeric attributes, thus creating a problem for data mining tasks. Deep learning solves big data problems and helps enhance computational performance. It is furthermore capable of extracting features from input. A convolutional deep learning model with an LSTM layer was designed (Di Gangi, Lo Bosco, & Rizzo, 2018), in which features are extracted from sequences and LSTM finds the features of the previous layer of sequential data with 50 hidden layers. LSTM helps identify long-term relationships between sequential data (Gunasekaran et al., 2021).

Machine learning techniques transform given input into desired output. For the classification task, input is mapped to output, which should be discrete. The classifier can be trained using unsupervised, supervised, or semi-supervised learning. In unsupervised learning, there is input data but no output variables. Examples are the Apriori algorithm for association rule learning and k-means for clustering.

In supervised learning, input and corresponding output variables map input to output. Examples are support vector machines, linear regression and random forest. Semi-supervised machine learning combines supervised and unsupervised learning, in which some data out of the large volume of input data is labelled and the rest of the data is unlabelled. Our model uses supervised machine learning, in which random forest uses many decision trees and employs the voting technique for more accuracy. Ensemble methods maintain accuracy because many tree-based algorithms are combined rather than a single tree-based algorithm being used. This helps reduce bias and variance factors, which can lead to a difference between actual and predicted values. These two factors can also lead to the underfitting or overfitting of the model. Ensemble methods are classified into three types: sequential, parallel and stacking. In sequential ensemble learning, base learners are generated sequentially, and many machine learning algorithms are combined to convert weak learners to strong learners, decreasing bias and variance (e.g., stochastic gradient boosting and Adaboost). In parallel ensemble learning, base learners are generated parallelly, improving the accuracy of the classification and reducing bias and variance. In stacking, multiple models are combined by splitting training datasets into two disjoint datasets that train and test several base learners. Predictions are then used to train another higher-level learner, after which blending yields a weighted average for the final result (e.g., voting classifier).

3.5 Random Forest classifier

The random forest (RF) classifier produces accurate results without overfitting problems and also works for missing values. Datasets are split randomly, random samples are selected, and decision trees that predict results are generated for each sample. The random forest classifier increases the accuracy of this model. Random forest is used for classification problems; a voting mechanism is employed for each predicted result, and the most voted class is predicted as the final result. When random forest is employed for regression problems, it finds the average of all predicted outputs to predict the final result. The random forest classifier uses a decision tree algorithm, in which features are selected randomly whenever nodes are split during random forest construction. In biological sequences, there is randomness in character sequences and any of the characters can appear multiple times at any position or time. This uncertainty and the way the random forest classifier works make it suitable for the type of datasets we have used during experimentation.

4. Experimental analysis

This model was implemented in the Google Colab environment with Python 3.7. The RAPIDS compatible package was installed with Miniconda 3. RAPIDS has open-source libraries designed especially for GPUs. This software uses CUDA primitives through a Python interface for low-level compute optimisation. Colab assigns different types of GPUs, but only P4, T4, P100, and V100 GPUs are compatible with RAPIDS. It is necessary to configure the appropriate environment with CUDA toolkit 11.2, RAPIDS 0.21, and cuML version 21.06.02. If an incompatible GPU is assigned at runtime (e.g., Tesla K80), the runtime must be set to default to wait for the right type of GPU allocation. Tesla T4, chosen as the GPU for experiments, has 2,560 cores and 16 GB of GPU memory with driver version 460.32.03.

4.1 Training and test data partitioning

DNA sequences were obtained from the NCBI. These datasets were divided into training and test data using a cross-validation parameter with a value of 10. Thus, complete datasets were divided into 10 groups and random forest automatically select test versus training data individually. One group was unannotated and nine groups were annotated. The output of the model was predicted by an unannotated group. We were able to use 70%, 60%, or any percentage of datasets for training the model. We used a random forest classifier to choose datasets randomly. We tested both approaches (i.e., dividing the datasets into training and testing datasets, as well as not splitting datasets by using a cross-validation of 10) and found that random splitting of the data did not affect the accuracy score of this model.

4.2 Evaluation metrics

The model was evaluated by a confusion matrix according to the actual and predicted class based on correctly identified/rejected and incorrectly identified/rejected as a member of class. There are four terms for this:

1. True Positive (TP): When both classes are true.
2. True Negative (TN): When both classes are false.
3. False Positive (FP): When the actual class is false and the predicted class is true.
4. False Negative (FN): When the actual class is true and the predicted class is false.

Based on these cases, this model was evaluated using the following metrics:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

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

$$\text{F1 Score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

$$\text{Specificity (TNR)} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

5. Results analysis

There are three different settings in Google Colab: CPU, GPU and TPU. All experiments were performed under the GPU setting to determine the runtime for sequence classification. The GPU was CUDA-enabled, and including CUDA-enabled libraries. This designed model helped reduce the training time, leading to less execution time for sequence classification. Table 1 shows the runtime taken by CPU with and without GPU.

Table 1 Runtime taken by CPU with and without GPU

No. of Sequences	CPU Time with GPU (ms)	CPU Time without GPU (ms)
200	32.2	43.2
400	33.6	50.5
600	35.0	53.9
800	35.3	64.9
1000	36.0	75.4
1200	36.3	79.6
1400	36.8	81.1
1600	37.1	107
1800	38.2	122
2000	38.9	139
2200	39.3	153
2400	39.8	162
2600	40.2	168
2800	42.5	172
3000	43.4	179
3200	43.8	185
3400	45.3	193
3600	47.1	205
3800	48.4	223
4000	50.8	230

The runtime analysis is shown in Figure 4(a) and Figure 4(b) using different types of graphs. At present, we have used 4,000 sequences varying in length from 200 to 4,000. The average accuracy remains the same irrespective of sequence length. The only difference is that CPU takes more time to search the large-sized sequences. Pre-padding was used to make all sequences of equal length because the shape of all datasets should be the same for machine learning.

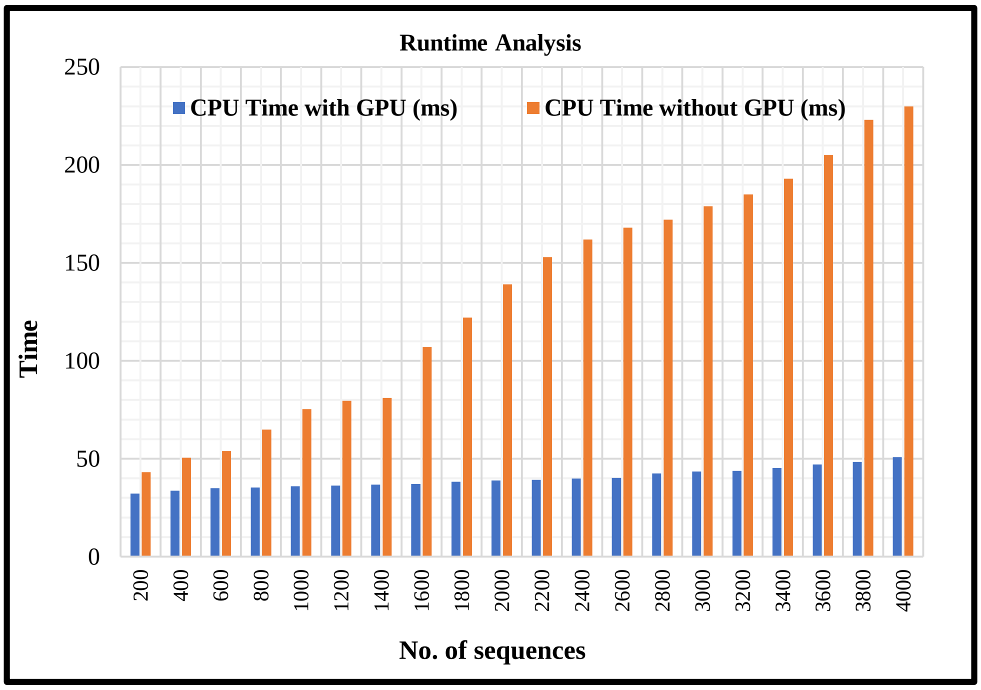

Figure 4 (a) Runtime analysis according to no. of sequences

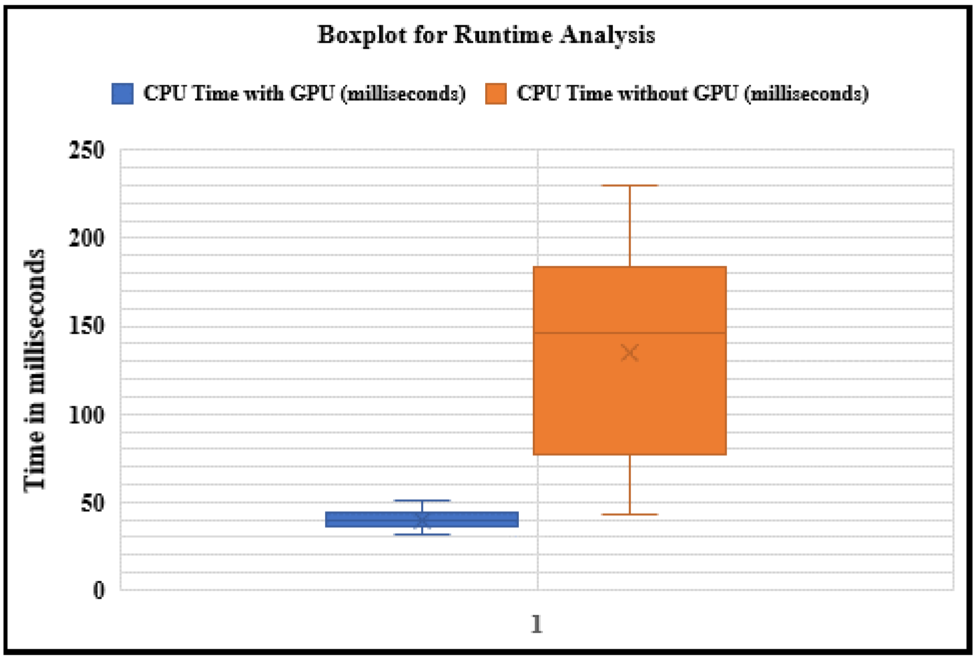

Figure 4 (b) Time analysis using boxplot for CPU and GPU according to Table 1

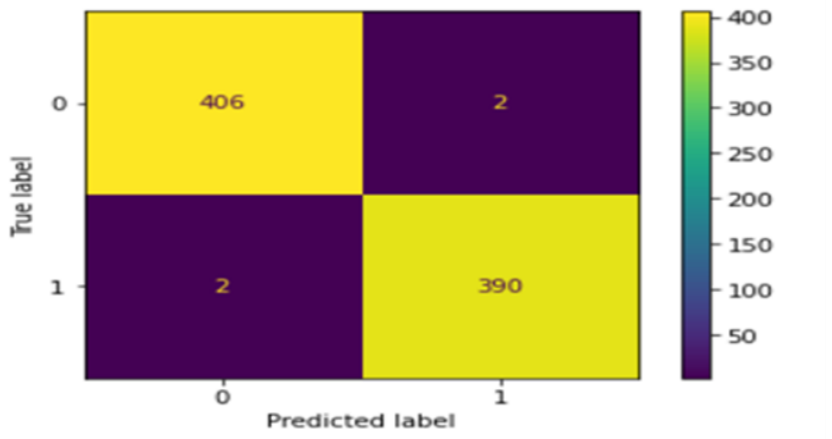

Figure 5 Confusion matrix according to the proposed model

When the same number of sequences was classified, the time taken by CPU with GPU was less compared to without GPU, as represented by Figure 4(a). Our model accelerated the performance of sequence classification with an accuracy of 99.5%. When this model was evaluated using precision, sensitivity, f1 score and specificity metrics, then these were also 99.5%. With GPU, there was a very small change in runtime, but without GPU, runtime consistently increased as the number of sequences increased. After classification, this model was evaluated by the confusion matrix; the output is shown in Figure 5.

Due to errors in the models, no model predicted 100% correct results, resulting in incorrect classification. False positives and negatives should be minimised. Results obtained by this designed model yielded 406 true positive and 390 true negative cases, as shown in Figure 5. The total number of false-positive and false-negative cases are 4, far fewer than the number of true cases, which clearly shows that our model was predicting correctly. If we divide the sum of correct predictions (406 + 390 = 796) by the total number of predictions (800), the model is then verified with an accuracy of 99.5%, as shown in Figure 5.

Table 2 Performance metrics of our proposed GPU based model

Performance Metrics of Our Proposed Model	**Value**
Accuracy	99.5%
Precision (Positive Predictive Value)	99.5%
Sensitivity/Recall (True Positive Rate)	99.5%
Quality Measure/F1 Score	99.5%
Specificity (True Negative Rate)	99.5%
False Positive Rate (FPR)	0.50%
False Negative Rate (FNR)	0.50%
Mathew Correlation (MC)	0.99%
Kappa Value (K)	0.99%

Table 2 shows the different performance metrics that were evaluated according to our proposed model. Precision gives a positive predictive value, whereas recall gives a true positive rate. Our main aim was to reduce false negatives and false positives, so we used both recall and precision, yielding a value of 99.5%. The F1 score, which maintains the balance if there are more actual negatives, was 99.5%. The false-positive and false-negative rates were 0.5%. The Mathew correlation was 0.99, which is nearly equal to 1 and clearly shows that classes were predicted well and there was a perfect positive correlation. Kappa takes the values from the main diagonal of the confusion matrix and adjusts these for agreement. Our proposed model had a Kappa value of 0.99, which shows very good agreement. Another parameter for evaluating this model is precision-recall curve as shown in Figure 6.

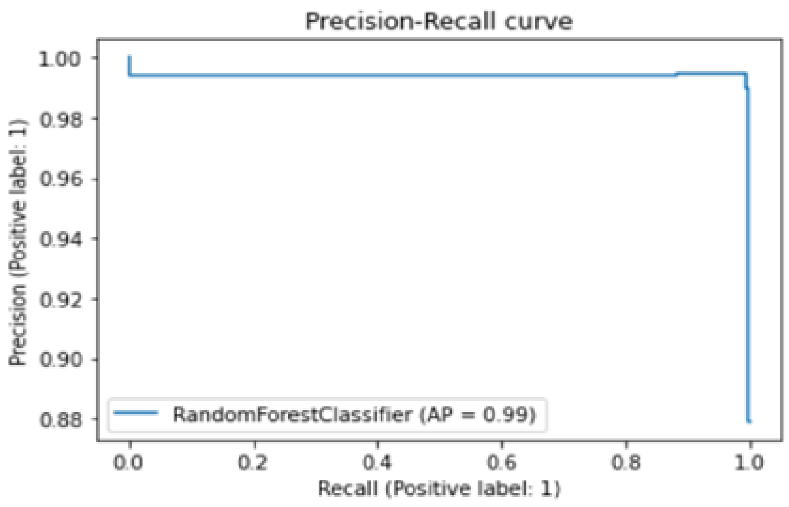

Figure 6 Precision-recall curve

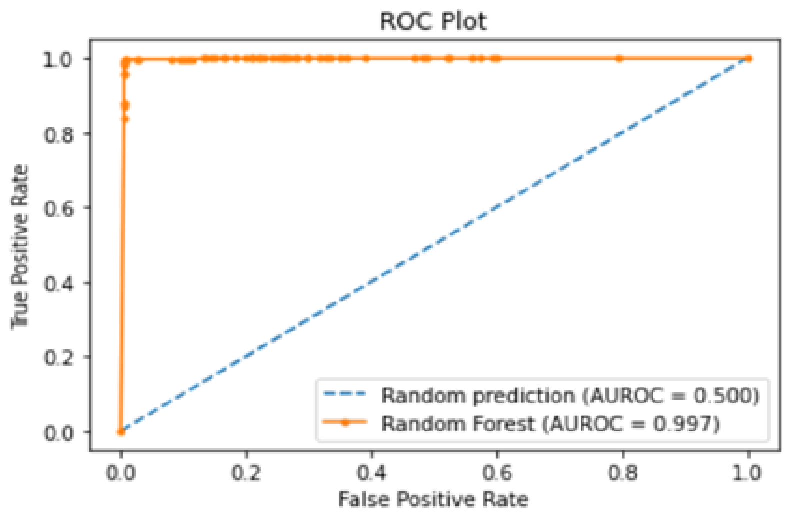

Figure 7 ROC

Our model also produced a good Receiver Operating Characteristic curve for predicting a positive class if the actual outcome was positive, as shown in Figure 7. An ROC curve shows a true positive rate and false-positive rate after the classification. A line is then plotted by taking the FPR value on x-axis and the TPR value on y-axis. The TPR can be determined by the recall metric. The FPR can be obtained by dividing the number of false positives by the total number of false positives and true negatives. The false-positive rate is also called 'inverted specificity'. The shape of the curve shows the desirable balance between false negatives values and false positives values. The line represents a model with perfect skill as it travels from left bottom to left top and then from left top to right top.

Our forest-based model fit the training data and was generalisable for new input data. The effectiveness of the model was measured by a validation curve that depends on hyperparameters, as shown in Figure 8. This validation curve was plotted to determine the effect of a single parameter on test versus training data. When plotted based on a hyperparameter that determines how many times each tree splits and stops, this curve showed that the model was neither underfitted nor overfitted.

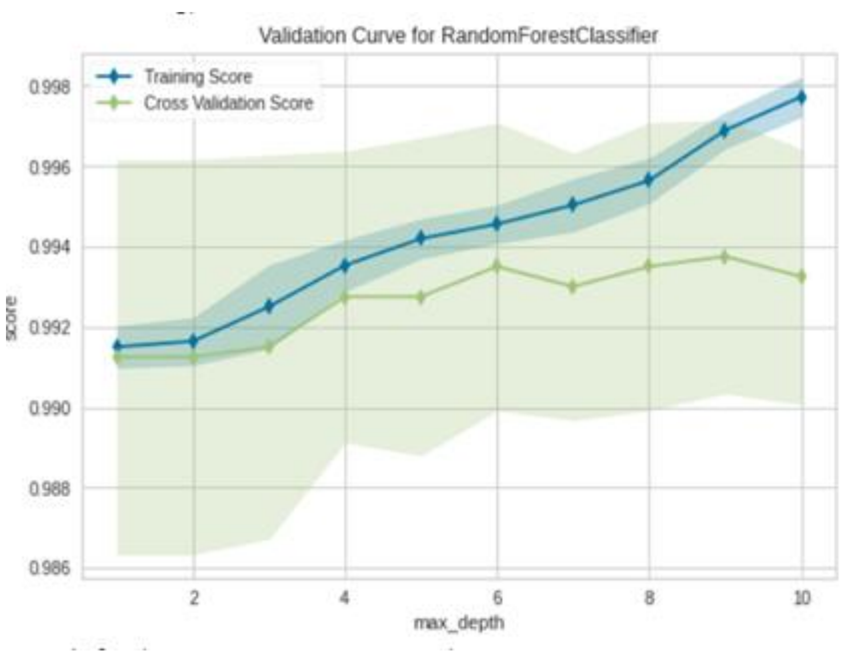

Figure 8 Validation curve

The best accuracy value was decided by varying the value of maximum depth from 1 to 10. When this curve was plotted, we used n_jobs = -1, meaning all cores of the GPU were used to parallelise computation when performing cross-validation. In cross-validation, the number of folds was chosen as cv = 10. The validation curve was generated by input X and y without splitting the datasets. In the case of cross-validation, however, splitting was performed using a number of folds chosen by us, guaranteeing that the random splitting of data did not affect the accuracy score of this model.

Table 3 Comparison of our model with MetaBinG

Number of Sequences	MetaBinG (ms)	Accuracy of MetaBinG (%)	Accuracy of CPU version (%)	CPU Time with GPU (ms)	Accuracy of GPU version (%)
100	4000	50.61	95	24.5	99.5
200	5000	60.82	95	27.2	95.0
300	6000	67.66	92	32.3	93.0
400	6000	71.56	96	33.6	98.8
500	8000	74.48	98	35.3	98.0
600	8000	77.24	93	36.0	95.7
700	9000	79.21	95	38.7	95.9
800	10000	80.25	98	39.0	99.4
900	12000	80.77	94	40.2	98.8
1000	13000	82.35	97	42.2	97.5

We compared our designed model with MetaBinG (Jia, Xuan, Liu, & Wei, 2011), which is the Markov model. Table 3 clearly shows that our model had an accuracy of 99.5% and outperformed in terms of accuracy and time taken by CPU and GPU when the same number of sequences was considered as an input. Figure 9 shows that if the number of sequences increases, then time taken by MetaBinG increases more abruptly than with the GPU-based model.

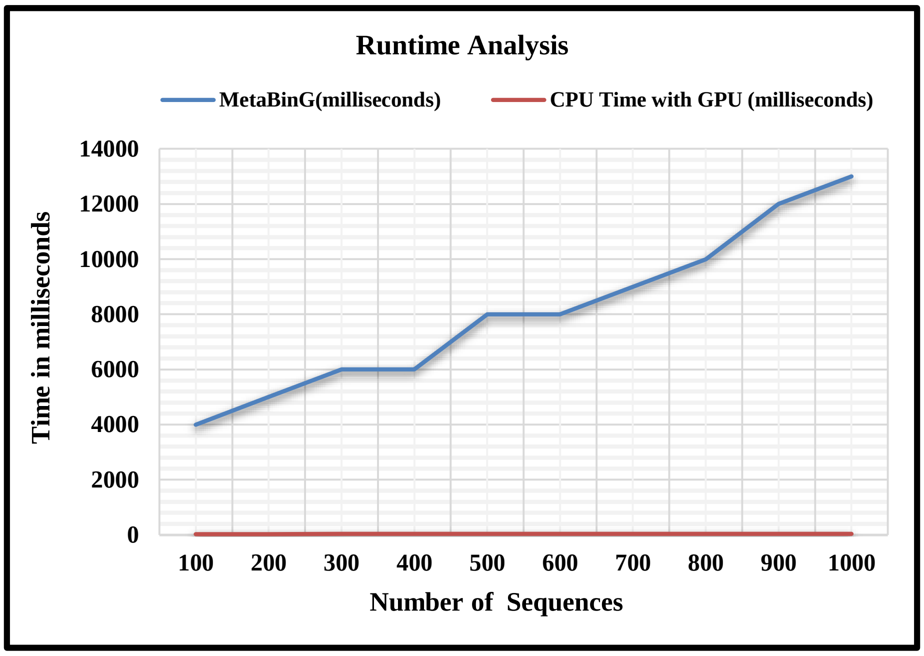

Figure 9 Runtime comparison of MetaBinG and our GPU-based model

Table 4 Comparison of our model with other models

Author	Model	Accuracy (%)
Gupta, Khare, and Aggarwal (2016)	AdaBoost	89
Al-Hyari and Al-Taee (2013)	Decision Tree	92
Park and Ryu (2016)	Bayesian	83
Jia et al. (2011)	Markov	82
Nguyen et al. (2016)	CNN	98
Lu, Hong and Wang (2020)	ResPPI	97
Zhang and Zong (2015)	CS-SVM	94
You et al. (2014)	ELM	85
Haque et al. (2018)	RF	80
Haque et al. (2018)	ANN	85
Gunasekaran et al. (2021)	CNN with LSTM	93
Our Proposed GPU based model	Random Forest	99.5

Figure 10 shows that when the number of sequences was 1,000, the maximum accuracy of MetaBinG was 80.77%, but in our CPU and GPU-based models, maximum accuracy was 98% and 99.5%, respectively. There was little variation in terms of accuracy for both models as number of sequences increased. Table 4 compares the accuracy of machine learning models designed by different researchers; this comparison is visualised in Figure 11.

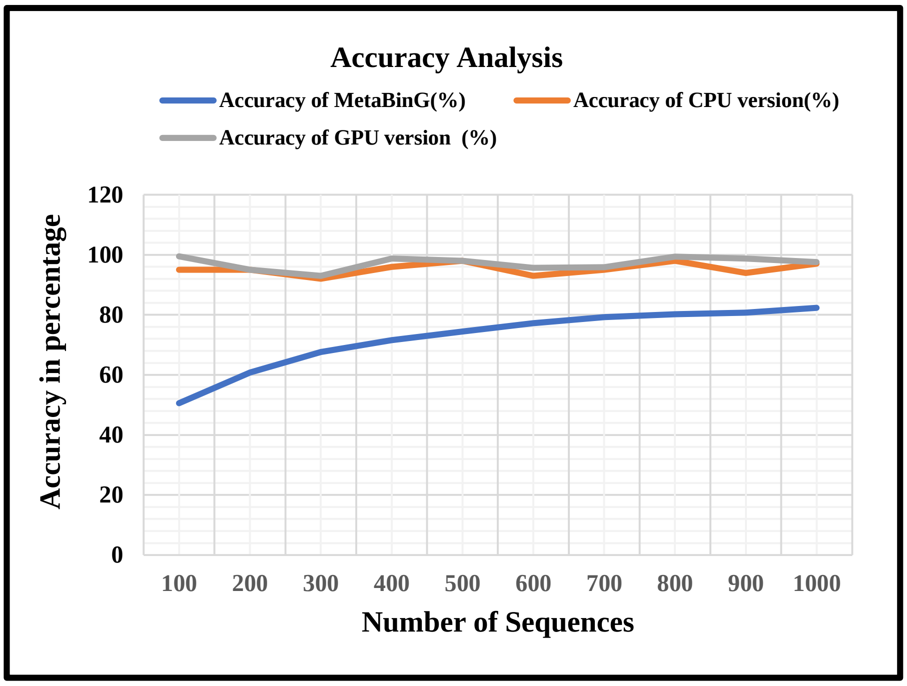

Figure 10 Comparison of accuracy between MetaBinG and our CPU and GPU-based model

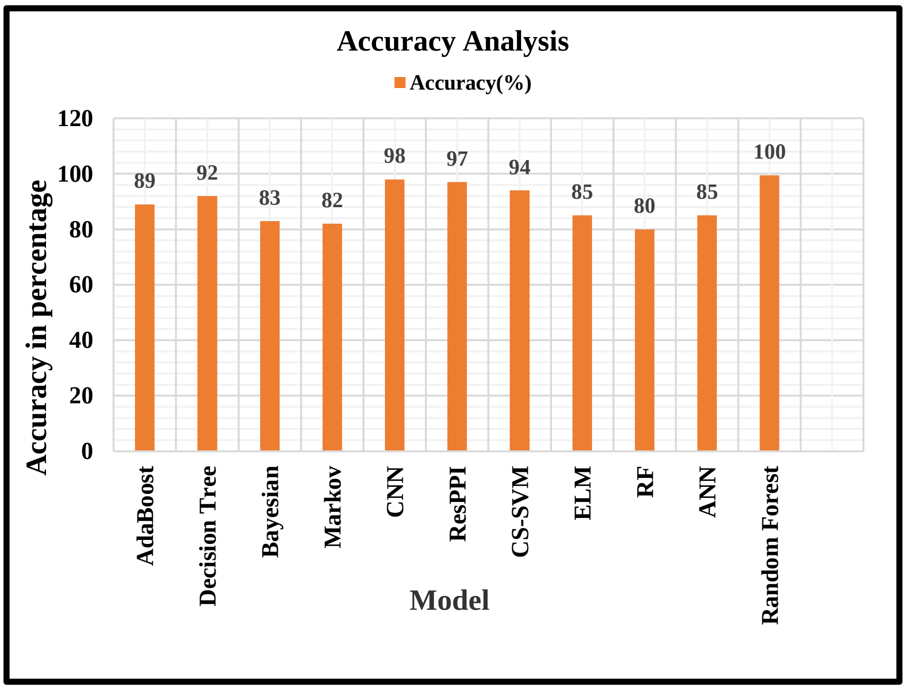

Figure 11 Comparison of accuracy between different machine learning models and our GPU-based model

Our GPU-based model ranks highest among previously developed models. When this model was compared with others metrics like precision, accuracy, and F1 scores, it demonstrated the highest accuracy, F1 score, and precision values, all of which were the same, as shown in Table 5 and Figure 12.

Table 5 Comparison of our GPU based model with others regarding accuracy and sensitivity

Methods	Accuracy (%)	Sensitivity (%)
Gaussian Processes	73.9	77.1
Linear Logistic Regression	69.6	74.7
Multilayer Perceptron	68.8	69.9
Neural Net	73.9	79.5
Support Vector Machine	69.2	75.0
Random Forest	80.0	68.8
ANN	85.3	80.0
Proposed Paper (RF)	99.5	99.5

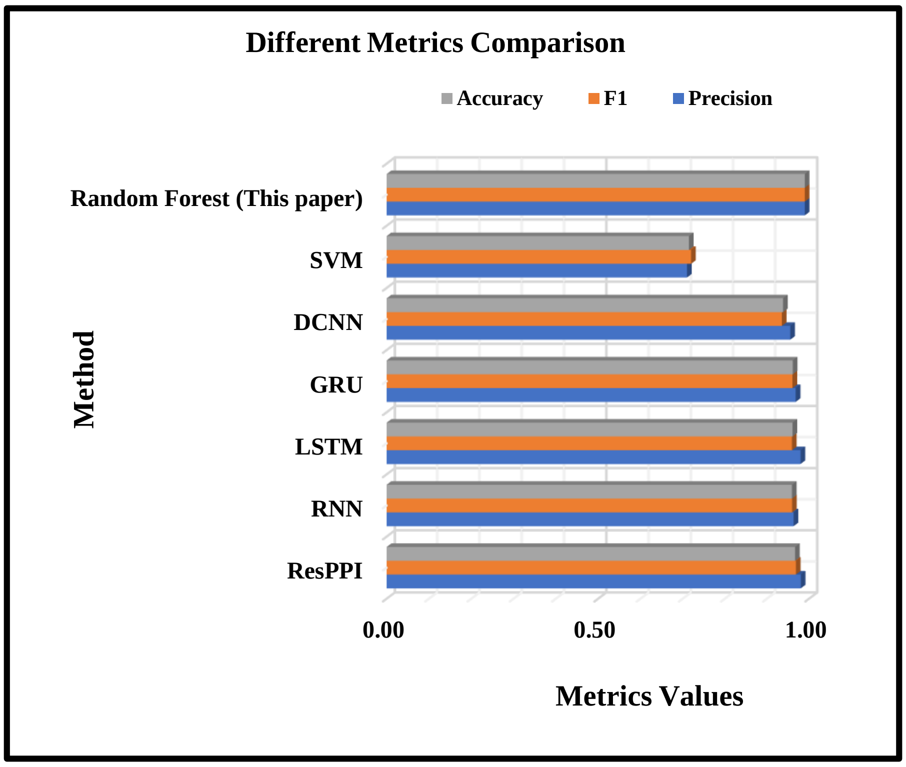

Figure 12 Different metrics comparison with our proposed model

Table 6 Comparison of our model with others for precision, F1 score, and accuracy

Methods	Precision	F1 Score	Accuracy
ResPPI	0.98	0.97	0.97
RNN	0.96	0.96	0.96
LSTM	0.98	0.96	0.96
GRU	0.97	0.96	0.96
DCNN	0.96	0.94	0.94
SVM	0.71	0.72	0.71
Random Forest (Proposed Paper)	0.99	0.99	0.99

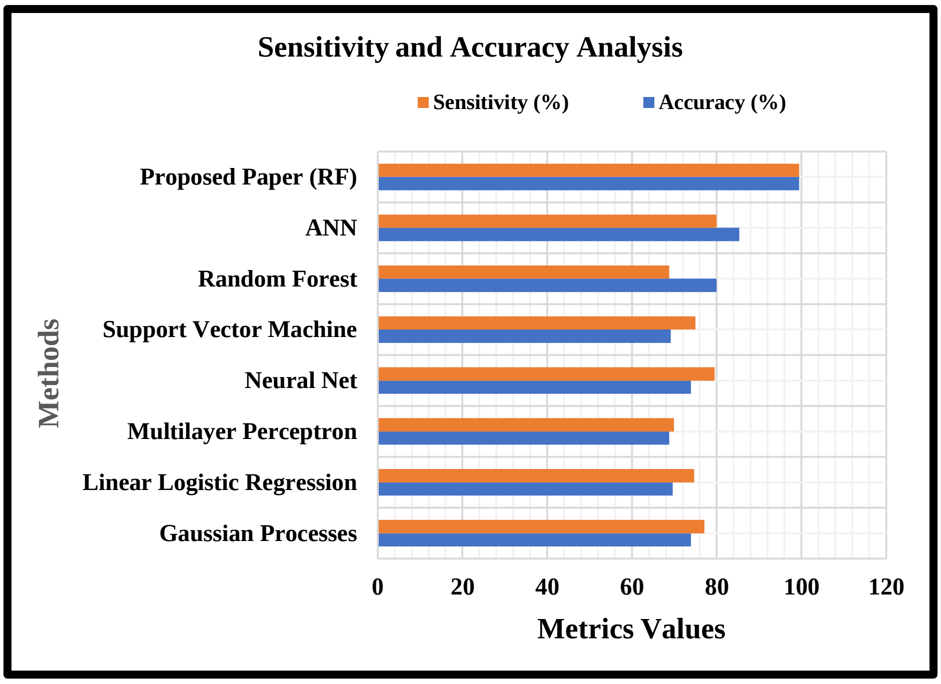

Figure 13 Comparison of sensitivity versus accuracy between different machine learning models and our GPU-based model

The comparison of this model with others models on the basis of accuracy and sensitivity is shown in Table 6. In addition to having the highest accuracy, our model also had the highest sensitivity of 99.5%. This is represented in Figure 13.

6. Conclusion and future scope

DNA sequences were classified by an ensemble-based random forest model. DNA sequences cannot be classified by machine learning methods directly, as DNA comprises a continuous sequence of characters without any space in between; thus, we applied different pre-processing steps to make DNA sequences compatible with machine learning methods, which helped in preparing numeric representation corresponding to sequence data. This proposed model utilises the full capabilities of GPUs and is suitable for handling large sequences as datasets. If the same sequences are classified by the CPU without the GPU, more time is required. With the GPU, all compute-intensive tasks are handled directly by the GPU, helping accelerate the classification task by reducing the overall execution time. For smaller-sized sequences, there is little difference in terms of execution time, but as sequence size increases, the GPU-based model reduces execution time by four times compared to CPU-based methods. The accuracy, precision, recall, F1 score, and specificity of the proposed model are all 99.5%. Fast, accurate alignment can help scientists discover new drugs quickly and accurately. In the future, this model could also be used for sequence clustering and text-based pattern matching. It can furthermore be employed with AI-based methods.

7. References

Abd–Alhalem, S. M., El-Rabaie, E. S. M., Soliman, N., Abdulrahman, S. E. S., Ismail, N. A., El-samie, A., & Fathi, E. (2021). DNA Sequences Classification with Deep Learning: A Survey. *Menoufia Journal of Electronic Engineering Research*, *30*(1), 41-51.DOI: 10.21608/mjeer.2021.146090

Al-Hyari, A. Y., Al-Taee, A. M., & Al-Taee, M. A. (2013). Clinical decision support system for diagnosis and management of chronic renal failure. In *2013 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT)*. IEEE. https://doi.org/10.1109/AEECT.2013.6716440

Andalon-Garcia, I. R., & Chavoya, A. (2017). PaMSA: A Parallel Algorithm for the

Global Alignment of Multiple Protein Sequences. *International Journal of Advanced Computer Science and Applications*, *8*(4). 513-522. https://doi.org/10.14569/ijacsa.2017.080468

Chen, X., Wang, M., & Zhang, H. (2011). The use of classification trees for bioinformatics. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *1*(1), 55-63. https://doi.org/10.1002/widm.14.

Chou, K. C., & Shen, H. B. (2006). Predicting eukaryotic protein subcellular location by fusing optimized evidence-theoretic K-nearest neighbor classifiers. *Journal of Proteome Research*, *5*(8), 1888-1897. https://doi.org/10.1021/pr060167c

Di Gangi, M., Lo Bosco, G., & Rizzo, R. (2018). Deep learning architectures for prediction of nucleosome positioning from sequences data. *BMC bioinformatics*, *19*(14), 127-135. https://doi.org/10.1186/ s12859-018-2386-9

Gao, J., Tian, L., Lv, T., Wang, J., Song, B., & Hu, X. (2019). Protein2vec: Aligning multiple ppi networks with representation learning. *IEEE/ACM transactions on computational biology and bioinformatics*, *18*(1), 240-249. DOI: 10.1109/TCBB.2019.2937771

Gunasekaran, H., Ramalakshmi, K., Rex Macedo Arokiaraj, A., Deepa Kanmani, S., Venkatesan, C., & Suresh Gnana Dhas, C. (2021). Analysis of DNA sequence classification using CNN and hybrid models. *Computational and Mathematical Methods in Medicine*, *2021*. https://doi.org/10.1155/ 2021/1835056

Gupta, C. P., Bihari, A, Tripathi, S. (2019). Human Protein Sequence Classification using Machine Learning and Statistical Classification Techniques. *International Journal of Recent Technology and Engineering*, 8(2), 3591-3599. 10.35940/ijrte.B3224.078219

Gupta, D., Khare, S., & Aggarwal, A. (2016). A method to predict diagnostic codes for chronic diseases using machine learning techniques. In *2016 International Conference on Computing, Communication and Automation (ICCCA)*. IEEE. https://doi.org/10.1109/CCAA.2016.7813730

Haque, M. R., Islam, M. M., Iqbal, H., Reza, M. S., & Hasan, M. K. (2018). Performance evaluation of random forests and artificial neural networks for the classification of liver disorder. In *2018 international conference on computer, communication, chemical, material and electronic engineering (IC4ME2)*. IEEE. https://doi.org/10.1109/IC4ME2.2018.8465658

Jia, P., Xuan, L., Liu, L., & Wei, C. (2011). MetaBinG: Using GPUs to accelerate metagenomic sequence classification. *PloS one*, *6*(11), e25353. https://doi.org/10.1371/journal.pone.0025353

Jiang, H., Ganesan, N., & Yao, Y. D. (2018). CUDAMPF++: A proactive resource exhaustion scheme for accelerating homologous sequence search on CUDA-enabled GPU. *IEEE Transactions on Parallel and Distributed Systems*, *29*(10), 2206-2222. https://doi.org/10.1109/TPDS.2018.2830393

Jiang, P., Wu, H., Wang, W., Ma, W., Sun, X., & Lu, Z. (2007). MiPred: classification of real and pseudo microRNA precursors using random forest prediction model with combined features. *Nucleic acids research*, *35*(suppl_2), W339-W344. https://doi.org/10.1093/nar/gkm368

L'heureux, A., Grolinger, K., Elyamany, H. F., & Capretz, M. A. (2017). Machine learning with big data: Challenges and approaches. *IEEE Access*, *5*, 7776-7797. https://doi.org/10.1109/ACCESS.2017. 2696365

Li, Y., Huang, C., Ding, L., Li, Z., Pan, Y., & Gao, X. (2019). Deep learning in bioinformatics: Introduction, application, and perspective in the big data era. *Methods*, *166*, 4-21. https://doi.org/10.1016/ j.ymeth.2019.04.008

Liu, B., Long, R., & Chou, K. C. (2016). iDHS-EL: identifying DNase I hypersensitive sites by fusing three different modes of pseudo nucleotide composition into an ensemble learning framework. *Bioinformatics*, *32*(16), 2411-2418. https://doi.org/10.1093/bioinformatics/btw186

Liu, B., Wang, S., Dong, Q., Li, S., & Liu, X. (2016). Identification of DNA-binding proteins by combining auto-cross covariance transformation and ensemble learning. *IEEE transactions on nanobioscience*, *15*(4), 328-334. https://doi.org/10.1109/TNB.2016.2555951

Liu, B., Yang, F., & Chou, K. C. (2017). 2L-piRNA: a two-layer ensemble classifier for identifying piwi-interacting RNAs and their function. *Molecular Therapy-Nucleic Acids*, *7*, 267-277. https://doi.org/10.1016/j.omtn.2017.04.008

Liu, Y., Hong, Y., Lin, C. Y., & Hung, C. L. (2015). Accelerating smith-waterman alignment for protein database search using frequency distance filtration scheme based on cpu-gpu collaborative system. *International journal of genomics*, *2015*. https://doi.org/10.1155/2015/761063

Lopez-del Rio, A., Martin, M., Perera-Lluna, A., & Saidi, R. (2020). Effect of sequence padding on the performance of deep learning models in archaeal protein functional prediction. *Scientific reports*, *10*(1), 1-14. https://doi.org/10.1038/s41598-020-71450-8

Lu, S., Hong, Q., Wang, B., & Wang, H. (2020). Efficient resnet model to predict protein-protein interactions with gpu computing. *IEEE Access*, *8*, 127834-127844. https://doi.org/10.1109/ACCESS.2020.3005444

Mahmud, M., Kaiser, M. S., Hussain, A., & Vassanelli, S. (2018). Applications of deep learning and reinforcement learning to biological data. *IEEE transactions on neural networks and learning systems*, *29*(6), 2063-2079. DOI: 10.1109/TNNLS.2018.2790388

Millán Arias, P., Alipour, F., Hill, K. A., & Kari, L. (2022). DeLUCS: Deep learning for unsupervised clustering of DNA sequences. *PloS one*, *17*(1), e0261531. https://doi.org/10.1371/journal.pone.0261531

Min, S., Lee, B., & Yoon, S. (2017). Deep learning in bioinformatics. *Briefings in bioinformatics*, *18*(5), 851-869. https://doi.org/10.1093/bib/bbw068

Mirzaei, S., Sidi, T., Keasar, C., & Crivelli, S. (2019). Purely structural protein scoring functions using support vector machine and ensemble learning. *IEEE/ACM transactions on computational biology and bioinformatics*, *16*(5), 1515-1523. https://doi.org/10.1109/TCBB.2016.2602269

National Center for Biotechnology Information. (n.d.). Retrieved from https://www.ncbi.nlm.nih.gov

Naznin, F., Sarker, R., & Essam, D. (2012). Progressive alignment method using genetic algorithm for multiple sequence alignment. *IEEE Transactions on Evolutionary Computation*, *16*(5), 615-631.

Nguyen, N. G., Tran, V. A., Phan, D., Lumbanraja, F. R., Faisal, M. R., Abapihi, B., & Satou, K. (2016). DNA sequence classification by convolutional neural network. *Journal Biomedical Science and Engineering*, *9*(5), 280-286. https://doi.org/10.4236/jbise.2016.95021

Park, K. H., Ryu, K. S., & Ryu, K. H. (2016). Determining minimum feature number of classification on clear cell renal cell carcinoma clinical dataset. In *2016 International Conference on Machine Learning and Cybernetics (ICMLC)*. IEEE. https://doi.org/10.1109/ICMLC.2016.7873005

Rangwala, H., & Karypis, G. (2005). Profile-based direct kernels for remote homology detection and fold recognition. *Bioinformatics*, *21*(23), 4239-4247. https://doi.org/10.1093/bioinformatics/bti687

Rashed, A. E. E. D., Amer, H. M., El-Seddek, M., & Moustafa, H. E. D. (2021). Sequence

Alignment Using Machine Learning-Based Needleman–Wunsch Algorithm. *IEEE Access*, *9*, 109522-109535. https://doi.org/10.1109/ACCESS.2021.3100408

Ravì, D., Wong, C., Deligianni, F., Berthelot, M., Andreu-Perez, J., Lo, B., & Yang, G. Z. (2016). Deep learning for health informatics. *IEEE journal of biomedical and health informatics*, *21*(1), 4-21. https://doi.org/10.1109/JBHI.2016.2636665

Sun, J., Palade, V., Wu, X., & Fang, W. (2013). Multiple sequence alignment with hidden Markov models learned by random drift particle swarm optimization. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, *11*(1), 243-257. https://doi.org/10.1109/TCBB.2013.148

Warris, S., Yalcin, F., Jackson, K. J., & Nap, J. P. (2015). Flexible, fast and accurate sequence alignment profiling on GPGPU with PaSWAS. *PloS one*, *10*(4), e0122524. https://doi.org/10.1371/journal. pone.0122524

Welivita, A., Perera, I., Meedeniya, D., Wickramarachchi, A., & Mallawaarachchi, V. (2018). Managing complex workflows in bioinformatics: an interactive toolkit with gpu acceleration. *IEEE Transactions on NanoBioscience*, *17*(3), 199-208. https://doi.org/10.1109/TNB.2018.2837122

You, Z. H., Zhu, L., Zheng, C. H., Yu, H. J., Deng, S. P., & Ji, Z. (2014, December). Prediction of protein-protein interactions from amino acid sequences using a novel multi-scale continuous and discontinuous feature set. In *BMC bioinformatics*. BioMed Central. https://doi.org/10.1186/1471-2105-15-S15-S9

Zhang, C., Zhang, F., Guo, X., He, B., Zhang, X., & Du, X. (2020). imlbench: A machine learning benchmark suite for CPU-GPU integrated architectures. *IEEE Transactions on Parallel and Distributed Systems*, *32*(7), 1740-1752. https://doi.org/10.1109/TPDS.2020.3046870

Zhang, J., & Zong, C. (2015). Deep Neural Networks in Machine Translation: An Overview. *IEEE Intelligent System.*, *30*(5), 16-25. https://doi.org/10.1109/MIS.2015.69

Zhu, X., Li, K., Salah, A., Shi, L., & Li, K. (2015). Parallel implementation of MAFFT on CUDA-enabled graphics hardware. *IEEE/ACM transactions on computational biology and bioinformatics*, *12*(1), 205-218. https://doi.org/10.1109/TCBB.2014.2351801

Zhu, Z., Wang, Z., Li, D., Zhu, Y., & Du, W. (2020). Geometric structural ensemble learning for imbalanced problems. *IEEE transactions on cybernetics*, *50*(4), 1617-1629. https://doi.org/10.1109/ TCYB.2018.2877663