applied sciences

MDPI

Article

Optimized Feature Learning for Anti-Inflammatory Peptide Prediction Using Parallel Distributed Computing

Salman Khan [1], Muhammad Abbas Khan [1], Mukhtaj Khan [2], Nadeem Iqbal [3,*], Salman A. AlQahtani [4,*], Mabrook S. Al-Rakhami [5] and Dost Muhammad Khan [6]

1 Department of Computer Science, Abdul Wali Khan University Mardan, Mardan 23200, Pakistan; salman@awkum.edu.pk (S.K.); abbaskheljee@gmail.com (M.A.K.)
2 Department of Information Technology, The University of Haripur, Haripur 22620, Pakistan
3 Division of Computer Science, Mathematics and Science, Collins College of Professional Studies, St. John's University, New York, NY 11439, USA
4 Department of Computer Engineering, College of Computer and Information Sciences, King Saud University, P.O. Box 51178, Riyadh 11543, Saudi Arabia
5 Department of Information Systems, College of Computer and Information Sciences, King Saud University, P.O. Box 51178, Riyadh 11543, Saudi Arabia; malrakhami@ksu.edu.sa
6 Department of Statistics, Abdul Wali Khan University Mardan, Mardan 23200, Pakistan; dostmuhammad@awkum.edu.pk
* Correspondence: nikhan@awkum.edu.pk (N.I.); salmanq@ksu.edu.sa (S.A.A.)

Abstract: With recent advancements in computational biology, high throughput Next-Generation Sequencing (NGS) has become a de facto standard technology for gene expression studies, including DNAs, RNAs, and proteins; however, it generates several millions of sequences in a single run. Moreover, the raw sequencing datasets are increasing exponentially, doubling in size every 18 months, leading to a big data issue in computational biology. Moreover, inflammatory illnesses and boosting immune function have recently attracted a lot of attention, yet accurate recognition of Anti-Inflammatory Peptides (AIPs) through a biological process is time-consuming as therapeutic agents for inflammatory-related diseases. Similarly, precise classification of these AIPs is challenging for traditional technology and conventional machine learning algorithms. Parallel and distributed computing models and deep neural networks have become major computing platforms for big data analytics now required in computational biology. This study proposes an efficient high-throughput anti-inflammatory peptide predictor based on a parallel deep neural network model. The model performance is extensively evaluated regarding performance measurement parameters such as accuracy, efficiency, scalability, and speedup in sequential and distributed environments. The encoding sequence data were balanced using the SMOTETomek approach, resulting in a high-accuracy performance. The parallel deep neural network demonstrated high speed up and scalability compared to other traditional classification algorithms study's outcome could promote a parallel-based model for predicting anti-Inflammatory Peptides.

Keywords: clustering computing; deep learning; optimum features; computational biology; anti-inflammatory peptides

Citation: Khan, S.; Khan, M.A.; Khan, M.; Iqbal, N.; AlQahtani, S.A.; Al-Rakhami, M.S.; Khan, D.M. Optimized Feature Learning for Anti-Inflammatory Peptide Prediction Using Parallel Distributed Computing. *Appl. Sci.* **2023**, *13*, 7059. https://doi.org/10.3390/app13127059

Academic Editors: Flavio Licciulli and Arianna Consiglio

Received: 8 May 2023
Revised: 6 June 2023
Accepted: 7 June 2023
Published: 12 June 2023

1. Introduction

Higher organisms use inflammation as a defensive mechanism to protect themselves against harmful infections and agents. When tissues are damaged during normal conditions by trauma, toxin, illness, or heat, inflammatory reactions occur [1]. Psoriasis, cancer, asthma, neurodegenerative diseases, diabetes, rheumatoid arthritis, and multiple sclerosis are chronic conditions and autoimmune disorders that occur when these reactions happen without infection or injury [2,3]. Several inflammatory activities are required to maintain the tolerance state [4]. Anti-inflammatory medications, like endogenous peptides identified by inflammatory reaction functions, have the potential to be utilized in innovative

treatments for inflammatory and autoimmune illnesses. Immunity to AIPs two therapeutic applications: inhibition of antigen-specific T (H) 1-driven reactions and regular T cell production [5]. Furthermore, several synthetic AIPs are essential in treating autoimmune illnesses and inflammatory disorders [6]. Alzheimer's disease, for instance, is caused by human amyloid-peptide chronic adenoidal direction. The Mice model generates amyloid -β, a pathological sign of microgliosis, neurological degeneration in the brain, Alzheimer's disease, and astrocytosis [7]. Immune suppressants and nonspecific anti-inflammatories are utilized to recognize inflammatory and autoimmune illnesses. Unfortunately, they are frequently associated with several side effects, such as inflammatory and pathogenic infections, and an increased risk of ineffectiveness [8].

Several autoimmune disorders and inflammatory diseases use wet lab experiments. However, recognizing the AIPs that use such experiments is costly and time-consuming. Therefore, in terms of computational biology, fast prediction of potential AIPs demands the creation of an effective sequential-based computer model. Various experimental approaches using machine learning for analyzing and identifying peptides have been proposed in the literature [9,10]. To classify AIPs peptides, a few computational models were proposed in the literature. However, the existing method performance is insufficient that has to be improved. Conventional machine learning methods are finding it difficult to analyse and accurately classify massive numbers of AIPs. Traditional technologies are finding it difficult to process them on time.

Similarly, many researchers have developed parallel and distributed deep learning models to enhance processing time under challenging situations. For speech recognition using deep learning algorithms, for example, Chen et al. and Mass et al. [11,12] employed distributed Graphic Processing Units (GPUs) clusters.

In the same way, Strom et al. [13] developed a cloud computing-based platform for a distributed Deep Neural Network (DNN) to tackle the communication bottleneck challenge, which happens throughout data-parallel Stochastic Gradient Descent (SGD). The primary methods considerably lower computation time: nevertheless, since they do not have the fault tolerance feature, the complete training process would be disturbed if the GPU node failed during execution.

Moreover, Sinthong et al. [14] introduced the parallelized DNN model for video data analysis and provided fault tolerance using the MapReduce Hadoop framework [15,16]. The Hadoop framework, named MapReduce, famous for big data analytics, includes built-in fault tolerance. However, the performance of the computational model based on Hadoop is limited because of the Hadoop framework's high I/O latency [17]. Furthermore, due to the Hadoop framework I/O delays, its usability for the applications of iterative processing, like machine learning methods, has been limited.

In this study, we used distributed computing models to classify AIPs sequences using a proposed scalable, fault-tolerant, parallelized multilayer DNN model. Spark framework [18] is used to develop the proposed model, a prominent data-related computing method that employs cluster processing nodes. In contrast to Hadoop, Spark offers in-memory computation, which allows data to be kept and processed in distributed memory during the map and reduce phases. This functionality lowers the I/O latency Spark framework, producing it a superior computational framework for machine learning approaches. Furthermore, the proposed model used parallelizing data and implicit coding approaches, considerably lowering DNN training time. In this context, the salient features of our work are summarized below:

- We proposed a scalable distributed deep generative model to classify AIPs and non-AIP sequences from large-scale peptide sequences.
- The proposed model considers the dataset's non-linearity by utilizing multi-stack processing layers and a non-linear activation function.
- The proposed model efficiently distributes data (i.e., model code) and computation across several processing nodes utilizing Spark to obtain massive parallelism. Further-

more, the proposed model incorporates fault-tolerance and scalability characteristics to create a robust system.

- The proposed model performance is extensively measured employing various performance measurement metrics, including scalability, speedup, accuracy, and execution time.

Spark cluster computing has been used to develop the proposed parallel model. Furthermore, the proposed model employs several feature extraction approaches to obtain the best features. Lastly, different computational and statistical-based metrics are utilized to analyze the proposed scalable and parallel model. We used Sensitivity (SN), Specificity (SP), Accuracy (ACC), and Matthew's correlation coefficient (MCC) to analyze statistical-based measurement metrics. In contrast, we used efficiency in terms of scalability, speedup, and computational times to evaluate computational-based measurement metrics.

The following structure is this paper's remainder. Section 2 describes the deep neural network background. Section 3 describes the proposed model structure (i.e., architecture) and implementation. Section 4 describes the performance assessment and experimental outcomes. Section 5 includes a discussion.

2. Recent Work

In the literature, few models for predicting AIPs have been proposed until now [19–21]. For example, Gupta et al. [19] developed a Support Vector Machine (SVM) based method in 2017 under the name of AntiInlflam predictor by using a hybrid with motif feature. In 2018 Manavalan et al. [20] used a Random Forest (RF) classifier with a primary sequence encoding feature to develop AIPpred. In 2019 Khatun et al. [21] used integrated multiple complementary features to establish a random forest-based classification method called PreAIP (See, Table 1).

Table 1. Recent work's feature extraction methods and classifiers.

Prediction Model	Feature Extraction Method	Classification
Gupta et al., 2017 [19]	TPC, motif-based features	SVM
Manavalan et al., 2018 [20]	AAC, AAI, DPC, PCP	RF, KNN, SVM
Khatun et al., 2019 [21]	AAI, KSAAP, pKSAAP, structural features	SVM, RF, NB

Feature abbreviations: TPC (tripeptide composition), DPC (dipeptide composition), AAindex (amino acid index), KSAAP (k-spaced amino acid pairs), pKSAAP (k-spaced amino acid pairs from the position-specific scoring matrix), AAC (amino acid contact), physicochemical descriptors (PCP).

Model abbreviations: SVM (support vector machine), RF (random forest), NB (Nave Bayes), k-Nearest Neighbor (KNN).

3. Architecture of Deep Neural Network

Artificial Neural Networks (ANNs) are fast gaining traction as a powerful machine learning method that enables it to solve complex bioinformatics problems with high efficiency in terms of accuracy. The ANNs primary goal was to create a hierarchical network capable of intellectual activity and perception like the human neural system. Previously, linear models were used to train learning models such as Perceptron [22] and Adeline [23] for data input. The linear approach has several drawbacks, including its inability to address complex problems. Later, when AI and bio-inspired models improved, the term " deep neural network" provided in-depth training and handled nonlinear problems.

Deep learning methods allow learning non-linear and complex functions employing computational models with multi-stack processing layers. In a variety of fields, deep

learning approaches have proven they are the most efficient method, including bioinformatics [24], image recognition [25], natural language processing [26], and speech recognition [27]. Furthermore, numerous publications have stated that deep learning approaches outperform conventional machine learning approaches in several complex learning problems [28–32].

The DNN, which evolved from ANNs with extraordinary learning abilities, is inspired by human brain activity and can represent big data in hierarchical representations. In recent years, the model has had numerous levels (i.e., numerous hidden layers), with an input layer and output layers connected by free learning parameters, including weights that considerably increase the learning capabilities of the DNN model. Furthermore, numerous researchers have found that as the dimension of the data and the variety of processing layers increases, the DNN structure makes computation more complex [33–38]. As a result, this research proposes a DNN model based on the Spark computing platform for parallel processing to reduce computational complexity. The DNN model in this paper is set up with five hidden layers, including an input and an output layer, as indicated in Figure 1. Multiple neurons in each layer produce the output by processing the input feature vector through Equation (1). The Xavier function is used to initialize the weight matrix on each neuron [39], which can maintain the same variance across all layers. The back-propagation procedure also updates the weight matrix to minimize errors between the target and output classes. Equation (2) is used to apply a non-linear activation function, like Relu, to input and hidden layers. In the dataset, the method learns the complex patterns and non-linearity with the assistance of the activation function. Furthermore, depending on the output produced by that neuron determines whether that neuron can be removed or ignored [40]. In addition, the sigmoid activation function generates some value between [0, 1] on the output layer, indicating the probability of data points belonging to a specific class.

$$y_i = g(b_i + \sum_{j-1}^{m} x_j{}^i w_j^i) \tag{1}$$

where y_i represents output on the layer i, b represents bias value, x is the input feature, w_j^i is the weight utilized by neuron j at layer i, and g is the non-linear sigmoid activation function, which can be derived by using Equation (2).

$$g(Z) = \frac{1}{1 + e^{-z}} \tag{2}$$

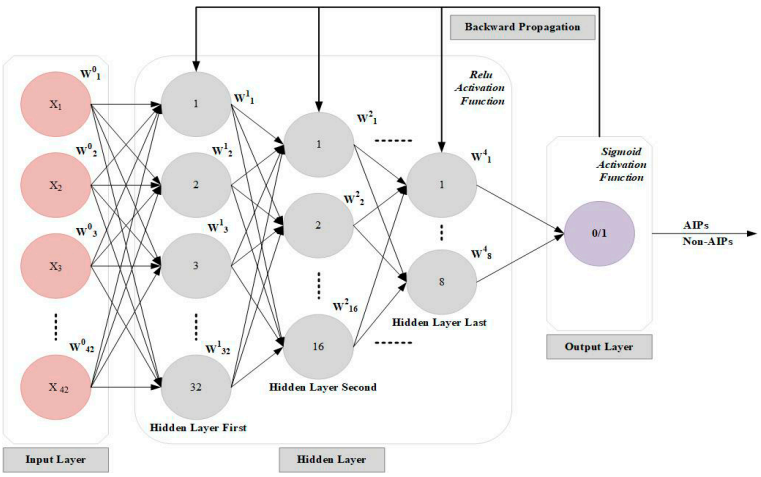

Figure 1. Deep Neural Network Architecture.

With the increase in the input size, the DNN model complexity and computational cost increased. Due to this, we calculated the algorithm's computational complexity to evaluate

the DNN model performance—the model input size (i.e., n) determines the computational complexity. When computational complexity equals $\Theta(1)$, it reaches its lowest value. The computational complexity of the model rises as the number of inputs increases. The proposed DNN model computational complexity can be separated into two main categories: forward and backward propagation [41]. We consider a fully connected DNN model of the number of the same layers (i.e., n), and neurons have $\Theta(n^4)$ Time complexity to compute the big O notation for forward propagation [41]. Moreover, because the proposed network employed the gradient descent for back-propagation for n iterations and has each layer (i.e., n) with neurons, the overall back-propagation run-time is given as $\Theta\left(n^5\right)$. As a result, the overall DNN algorithm computational complexity can be defined as.

$$\Theta(n^5 + n^4) \cong (n^5) \tag{3}$$

Equation (3) indicates that deep neural networks have high computational complexity compared to other learning methods. Therefore, distributed, and parallel computational approaches can reduce the DNN model computational cost and training time.

4. Proposed Model Design

The proposed model design is introduced in this section. Figure 2 illustrates the proposed model architecture that includes several mechanisms, which are discussed in more detail.

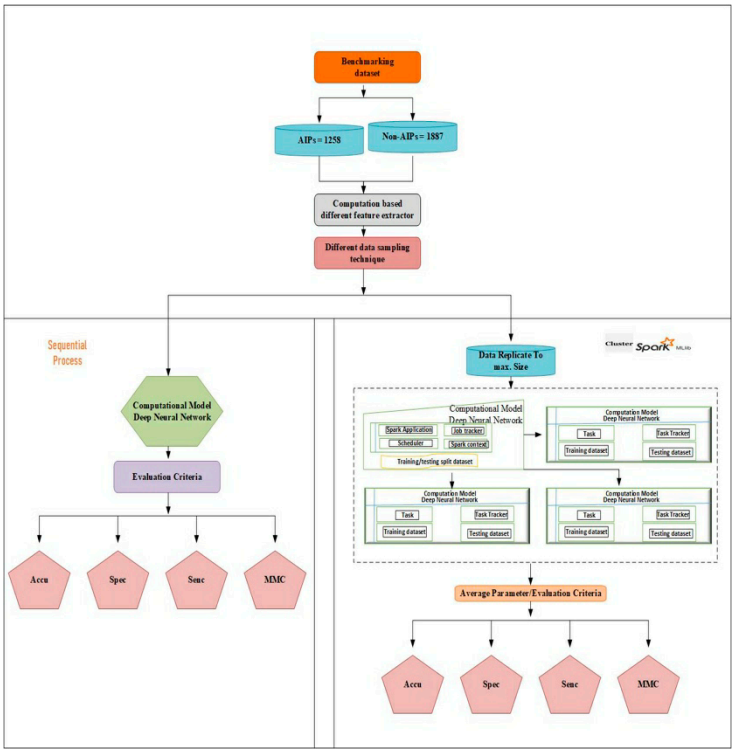

Figure 2. Proposed Sequential Model And Spark Framework's Architecture Mechanism.

4.1. Benchmark Dataset

There were AIPs and non-AIP peptides in the Benchmark dataset. The IEDB database [42] and a newly published paper [21] were utilized to develop this paper's benchmark dataset. The benchmark dataset is mathematically represented using Equation (4).

$$D = D^+ \cup D^- \tag{4}$$

where AIPs are denoted with D^+, while non-AIPs are denoted with D^-. The D represents the AIPs and non-AIPs sequences union. We got 3145 peptide sequences from the [21,42], with 1258 AIPs sequences (i.e., D^+) and 1887 non-AIPs sequences (i.e., D^-).

4.2. Technique for Feature Formulation

Biological sequences are provided in FASTA format with various sequence lengths in bioinformatics. Biological sequences cannot be processed naturally using statistical machine-learning methods. These methods work with data that is either discrete or numeric [31,43]. As a result, before using any machine learning method to learn biological sequences, they must be converted into a numeric-valued feature vector. Conversely, the sequence information order and pattern may significantly change during the formulation process. Computational biology has implemented many methodologies to transform peptide sequences of varying lengths into feature vectors while preserving the sequence information order and pattern [44].

4.2.1. Amphiphilic Pseudo-Amino Acid Composition (APAAC)

The APAAC features set is distinctly similar to PAAC descriptors.

$$H_1\,(1) \;=\; \frac{H_1^0\,(i) \;-\; \sum_{i-1}^{20} \frac{H_1^0\,(i)}{20}}{\sqrt{\frac{\sum_{i-1}^{20} [H_1^0\,(i) \;-\; \sum_{i-1}^{20} \frac{H_1^0\,(i)^2}{20}]}{20}}} \tag{5}$$

$$H_2\,(2) \;=\; \frac{H_2^0\,(i) \;-\; \sum_{i-1}^{20} \frac{H_2^0\,(i)}{20}}{\sqrt{\frac{\sum_{i-1}^{20} [H_2^0\,(i) \;-\; \sum_{i-1}^{20} \frac{H_2^0\,(i)^2}{20}]}{20}}} \tag{6}$$

where,

- $H_1^0\,(i)$ represents the hydrophobicity
- $H_2^0\,(i)$ represents the hydrophilicity.

The hydrophobicity and hydrophilicity correlation functions are mathematically expressed as

$$\begin{aligned} H_{i,j}^1 \;&=\; H_1(i)\,H_1\,(j), \\ H_{i,j}^2 \;&=\; H_2(i)\,H_2\,(j) \end{aligned}$$

where $H_{i,j}^1$ and $H_{i,j}^2$ stands for the correlation functions of hydrophobicity and hydrophilicity. As a result, sequence order factors can be defined as follows:

$$\left.\begin{aligned} \tau_1 \;&=\; \tfrac{1}{N-1} \sum_{I-1}^{N-1} H_i^1,\; i \;+\; 1 \\ \tau_2 \;&=\; \tfrac{1}{N-1} \sum_{I-1}^{N-1} H_i^2,\; i \;+\; 1 \\ \tau_3 \;&=\; \tfrac{1}{N-2} \sum_{I-1}^{N-2} H_i^1,\; i \;+\; 2 \\ \tau_4 \;&=\; \tfrac{1}{N-2} \sum_{I-1}^{N-2} H_i^2,\; i \;+\; 2 \\ &\cdots \\ \tau_{2\lambda-1} \;&=\; \tfrac{1}{N-\lambda} \sum_{I-1}^{N-\lambda} H_i^1,\; i \;+\; \lambda \\ \tau_{2\lambda} \;&=\; \tfrac{1}{N-\lambda} \sum_{I-1}^{N-\lambda} H_i^2,\; i \;+\; \lambda \end{aligned}\right\} \tag{7}$$

where λ an integer parameter and APAAC are defined as:

$$\left.\begin{array}{rcl} p_u & = & \frac{f_u}{\sum_{i-1}^{20} f_i + w \sum_{j-1}^{2\lambda} \tau_j}, \; where \; 1 \leq u \leq 20 \\ p_u & = & \frac{w\tau_u}{\sum_{i-1}^{20} f_i + w \sum_{j-1}^{2\lambda} \tau_j}, \; where \; 20+1 \leq u \leq 20+\lambda \end{array}\right\} \quad (8)$$

4.2.2. PseAAC of Distance-Pair and Reduced Alphabet

The descriptor combines information on amino acid distance pair coupling, and the amino acid reduced alphabet profile in the generic pseudo amino acid composition vector. There are three of them: Cp (13), Cp (14), and Cp (15), respectively, for the reduced alphabet profile:

$$\begin{array}{rcl} \text{Cp}\ (13) & = & \{\text{MF ; IL; V; A ; C; WYQHP ; G ; T; S; N ; RK ; D ; E}\} \\ \text{Cp}\ (14) & = & \{\text{EIMV ; L; F; WY ; G ; P ; C ; A ; S ; T ; N ;HRKQ ;E ; D}\} \\ \text{Cp}\ (15) & = & \{\text{P ; G ; E ; K ; R; Q ; D ; S; N; T; H ; C ; I ; V ; W ; YF ; A ; L ;M}\} \end{array} \quad (9)$$

where,

- Cp represent the cluster profile.
- The numbers (13) (15) represent the dimension of the feature vector of these cluster profiles.
- A semicolon (;) separates each letter, showing they are part of the same cluster.

Our research employed a hybrid distance Pair and APAAC technique that outperforms all other feature extraction techniques.

4.3. Data-Balancing Techniques

One of machine learning's most complex challenges is working with unbalanced datasets. There is an imbalanced dataset when the dominant class membership vastly outnumbers the minority class membership, which may considerably affect the machine learning method performance (i.e., unreliable biased and results). Several resampling methodologies, including under-sampling, over-sampling, and hybrid sampling techniques, balance the original imbalance dataset [45]. Prior research has shown that dealing with the problem of imbalanced data leads to better classification models [46–49]. The most basic strategy is random under-sampling, where most data samples are randomly eliminated to balance the dataset. But, in the majority class, random data point removal may outcome in the loss of unwanted information. The synthetic minority oversampling technique, Tomek link (SMOTETomek), is a hybrid technique for dealing with data imbalances [50]. SMOTETomek has been used in several studies with positive data balance and model performance outcomes. Goel et al. [51] used eight datasets from the UCI Repository website to examine five ways to fix imbalanced data. According to the results, SMOTETomek enhances most of the dataset's model accuracy. Chen et al. [52] employed resampling ways to solve unbalanced challenges and an RF approach to estimate lane-changing risk levels to create a line-changing behavior framework. The results indicate that SMOTETomek significantly improves the model's accuracy to 80.3%.

The synthetic samples are created by SMOTE as follows:

$$X_{new} = X_i + (X_i' - X_i) * a \quad (10)$$

where X_{new} denotes synthetic data, X_i denotes minority samples, X_i' denotes the X_i k nearest neighbors, and a denotes a random value between 0 and 1.

The synthetic minority oversampling technique edit nearest neighbors (SmoteENN) and combines under-sampling (i.e., ENN) [53] and oversampling (i.e., SMOTE). SMOTE uses the KNN technique to find and choose the closest k neighbours, connect them, and

produce new samples in the space. The ENN approach is utilized to clean up the oversampled data.

Previous research has demonstrated that using SmoteENN to deal with unbalanced data improves model accuracy. Furthermore, none of the earlier studies used the SmoteENN technique. Therefore, in this paper, we offer machine learning approaches and the SmoteENN method to balance datasets to accurately predict AIPs and non-AIP sequences (shown in Table 2).

Table 2. Comparison of different data balancing techniques.

Data-Balancing Techniques	SN (%)	SP (%)	ACC (%)	MCC
Simple data	46.617	67.403	59.24	14.234
Under-sampling	68.595	51.538	60.09	20.394
SmoteTomek	79.29	61.257	68.51	46.968
SmoteENN	89.011	82.352	83.40	71.066

4.4. Apache Spark

Spark is a distributed computational model and an open-source framework used to analyze and process massive amounts of data on cluster computing platforms [54,55]. It supports in-memory computing that allows data to maintain and processes in shared physical memory. When the entire memory is occupied, and the system cannot store more data, the data is split by Spark into secondary storage. Because of this feature, Spark provides a better computational environment for iterative methods like machine learning methods. The Spark framework comprises a cluster manager, worker node, SparkContext, and driver program. With the support of a cluster manager, a user job can use SparkContext to allocate resources to it and access the recourses of the cluster. The primary method that creates SparkContext is in the driver program, a Java process. Cluster Manager is a resource management module that schedules, allocates, and shares resources across multiple jobs. Various cluster managers, including Mesos, Hadoop Yarn (yet another resource negotiator), and Standalone Cluster Manager, can be used to deploy the Spark Framework. The master/slave architecture is used in the Spark framework. The driver is the master node, which serves as the central coordinator, while the slave nodes serve as the distributed workers (executives). Spark can access data from various sources, including Cassandra, HBase, S3, and HDFS. In addition, the Spark framework comes with various built-in libraries like MLib, Spark Streaming, GraphX, and Spark SQL to facilitate application developers across multiple domains. MLib is a scalable library that supports machine learning methods like random forests, K-nearest neighbors, SVM, etc.

Spark Streaming is an API that is language-integrated, which helps developers to build quickly fault-tolerant, scalable, real-time, and streaming data processing systems. GraphX is a built-in library that is built for graphic iterative parallel computations. Application developers can use SparkSQL to query structured data employing the Spark system. In Spark, a significant memory abstraction is provided in the form of an in-memory Resilient Distributed Dataset (RDD) that enables fault tolerance and capabilities of high scalability [56] in addition to the built-in libraries. RDD is the set of read-only objects split over several cluster processing nodes to enable parallel processing. Furthermore, by implementing RDD Lineage [57–59] service, the RDD acquires fault-tolerance capability. If a node fails during execution, RDD automatically recalculates RDD distribution using parent RDD.

4.5. Parallel Deep Neural Network

This section introduces using the Spark Framework to create parallel deep neural networks. The proposed parallel model is illustrated in Figure 3, in which the Spark framework splits massive training data in RDDs samples labeled $D1$, $D2$, $D3$, ..., Dn and shares them over a nodes cluster (i.e., utilizing data parallelization). Additionally, a DNN model copy is distributed across several workers by the Spark framework (i.e., using

parallelization of the model). The training begins with the DNN model being executed across all worker nodes simultaneously (i.e., at the same time), and several models have been given parallel training using the provided hyperparameters indicated in Table 2. After the training process (shown in Figure 3 with an orange line), each worker node in the trained models updates the master node after receiving a trained model (i.e., in globally trained mode, every worker assigns a different test dataset as an RDD sample). Worker nodes use a globally trained model to be applied to specific test datasets and produce local output (i.e., classification metrics). Finally, the average parameter function combines *n* local outcomes via the master node (i.e., parameter server) to produce the final result.

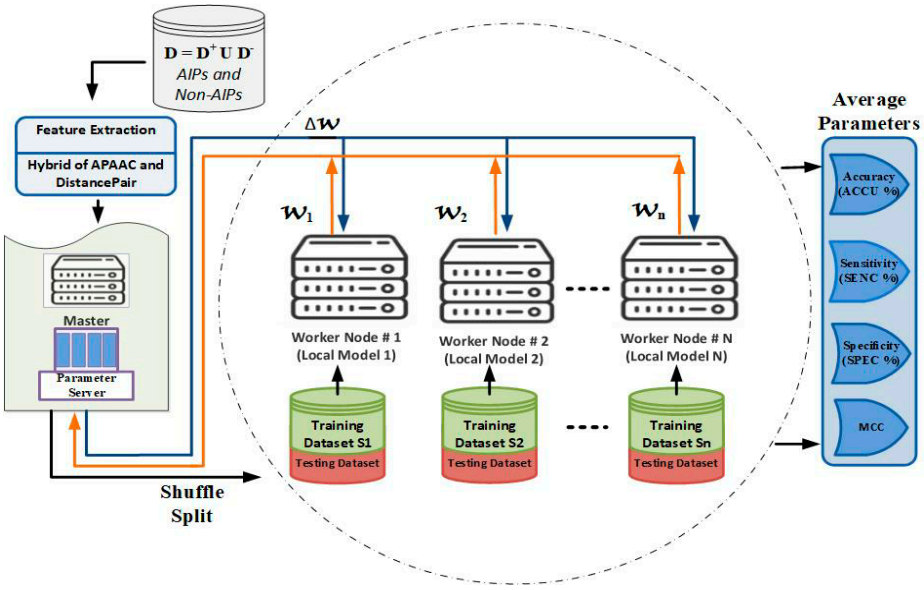

Figure 3. Proposed Parallel Deep Model Framework.

To improve the proposed model, the back-propagation mechanism is used on every model and is distributed over the worker nodes. The DNN model iterates to a lower loss function in each iteration. Every single worker node calculates Stochastic Gradient Descent (SGD) on a provided subset to reduce prediction error. Every worker node reports the computed SGD to a centralized parameter server. The worker node and parameter server communicate using a synchronous communication mechanism. The parameter server gathers partial gradients of the worker nodes to calculate the new weights set. After this, the parameter server distributes updated weights to the worker nodes that recalculate the gradients (illustrated with the blue line in Figure 3). The server parameter in the proposed model handles the distributed scaling model and functions as a coordinating agent among the worker nodes and central server [60].

Compared to asynchronous SGD optimization, synchronous SGD optimization achieves better scalability and efficiency [61,62]. However, because of the occasional slowdowns of a worker node, the synchronous SGD process can decrease model performance. We suppose that every worker nodes in the proposed model are homogeneous and there are no slowdown worker nodes. The proposed model, at a higher level, trains and tests in the following ways:

1. The Spark framework breaks down many sequences into smaller parts and divides them among all worker nodes.
2. The DNN model is replicated on each worker node.
3. Each worker node runs a model on a particular partition for local model training.
4. The worker node communicates the local model parameters to the parameter server. The parameter server uses the parameter average overall local models to integrate all parameters.
5. The parameter server updates the weights and distributes them to the worker nodes.

6. Steps 3, 4, and 5 are repeated several times in the proposed model to generate a globally trained model with the best parameters.
7. The final average classification metrics are created by applying the globally trained model to the testing dataset.

Using a parallel approach, the proposed method dramatically reduces computing time while maintaining high scalability. The proposed model can scale up (i.e., extended) by adding more nodes, which increases the model performance in terms of speedup and computation times (Section 4 discussed it in detail). Furthermore, a proposed model attained fault tolerance by distributing copies of data samples as RDDs among multiple worker nodes, effectively overcoming node failure problems. If the node fails during the execution of a job, the spark automatically detects it. It allocates the failed node workload to an additional obtainable node to ensure that it does not affect job completion.

4.6. Performance Evaluation Metrics

Computational-based and Accuracy-based matrices were used to assess the proposed model performance. We employed commonly used accuracy-based metrics, including (I) ACCU, which represents the model's overall accuracy, (II) SPEC, which represents the precision of the model, (III) SENC, which represents the sensitivity of the model; and (IV) MCC, indicates Mathew's correlation coefficient [63]. Equations (11)–(14) can be used to calculate these metrics.

$$ACCU = 1 - \frac{N_{-}^{+} + N_{+}^{-}}{N^{+} + N^{-}}, 0 \leq Accu \leq 1 \tag{11}$$

$$SPEC = 1 - \frac{N_{+}^{-}}{N^{-}}, 0 \leq Spec \leq 1 \tag{12}$$

$$SENC = 1 - \frac{N_{-}^{+}}{N^{+}}, 0 \leq Senc \leq 1 \tag{13}$$

$$MCC = 1 - \frac{1 - \left(\frac{N_{-}^{+} + N_{+}^{-}}{N^{+} + N^{-}}\right)}{\sqrt{\left(1 + \frac{N_{+}^{-} + N_{-}^{+}}{N^{+}}\right)\left(1 + \frac{N_{-}^{+} - N_{+}^{-}}{N^{-}}\right)}}, -1 \leq Mcc \leq 1 \tag{14}$$

where,

- N^{+} denotes the AIPs sequences' total number.
- N^{-} denotes the of non-AIPs sequences total number.
- N_{-}^{+} denotes the AIPs sequences' total number that the proposed model incorrectly predicts as non-AIPs sequences.
- N_{+}^{-} denotes non-AIPs sequences' total number that the proposed model incorrectly predicted as AIPs sequences.

Computational metrics like scalability, computation time, and speedup are analytical metrics that can be computed using simulation results.

5. Results and Discussion

The proposed model's efficiency and performance are discussed and measured using computational and accuracy-based metrics on the benchmark dataset.

5.1. Experimental Setup

We used four physical processing nodes to configure a Spark cluster with default configurations. Table 3 describes the basic hardware and software specifications used in the experiments. All processing nodes were set up utilizing the Ubuntu 18.04 LTS operating system, Hadoop 2.7.3, and Spark 2.0. One of the processing nodes was designated as a

master, while the other three were designated as worker nodes. Furthermore, the master node served as both a worker and a master node.

Table 3. Apache Spark cluster configuration details.

	Processor	**3.20 GHz × 4**
Specification of Single node	CPU	Intel Core™
	Connectivity	100 Mbps Ethernet LAN
	Hard disk	480.4 GB
	Memory	16 GB
Software	Operating system	Ubuntu18.04LTS
	Hadoop	2.7.3
	Spark	2.0
	OS Type	64-bit
	JDK	1.8

5.2. DNN Parameters Optimization

The DNN model usually involves a large number of parameters that significantly impact the performance of a model. The commonly used parameters that determine the design of a DNN model are the number of hidden layers, activation functions, learning rate, weight initialization, model optimization methods, etc. These parameters are referred to as hyper-parameters and are listed in Table 4. Table 4 shows the effects of only two hyperparameters that substantially impact the learning rate and activation function.

Table 4. DNN model hyper-parameter list with optimum values.

Parameters	**Optimized Values**
Learning rate	0.001
Iteration	128
Seed	1234 L
Number of hidden layers	12-10-2-1
Number of Neurons	5
Activation Function	ReLu
Regularization.l2	0.001
Weight initialization	XAVIER function
Optimizer	0.1
Dropout	SGD Method
Updater	ADAGRAD function

Different learning rates are utilized to produce the best accuracy, as shown in Table 5. At a 0.001 learning rate and using the Relu activation function, the DNN model achieved the maximum accuracy of 82.0% and 83.40%, respectively, using the Tanh activation function. As a result, Relu and 0.001 are the optimal activation function and learning rate configuration parameters, respectively.

5.3. Learning Algorithms Performance Comparison

The proposed DNN model is compared to that of other famous ML algorithms such as Probabilistic Neural Network (PNN) [64], k-Nearest Neighbor (KNN) [65], Random Forest (RF) [66], Support Vector Machine (SVM) [67], MLP (Multilayer Perceptron) [68], Nave Bayes (NB) [69], and logistic regression [70]. Table 6 shows the conventional machine learning methods with optimized values of their parameters. Table 7 compares the performance of the several learning methods. From Table 7, the DNN model beats other machine learning techniques. The DNN model, for example, had the best accuracy of 83.40%, while the RF model had the second-best accuracy of 82.21%. With an accuracy of 64.35%, the Logistic regression model is the least accurate.

Table 5. Impact of different learning rates and activation functions.

LR	Tanh ACC (%)	Relu ACC (%)
0.009	78.83%	80.24%
0.008	81.78%	81.85%
0.007	80.45%	81.22%
0.006	81.29%	81.36%
0.005	81.64%	81.44%
0.004	81.86%	81.36%
0.003	82.00%	82.41%
0.002	81.16%	82.56%
0.001	**82.00%**	**83.40%**
0.01	77.07%	81.63%
0.02	64.63%	73.21%

Table 6. Conventional Machine Learning Methods Parameters with Optimized Values.

Classifier	Parameters	Optimized Values
SVM	Cost	10
	Kernel Type	Linear
	Kernel Degree	2
	Coef0	9
	Gamma	0.01
	Shrinkage	TRUE
RF	Mtry	8
	No. of trees	350
KNN	Weighting	10
	k-neighbors	Distance
Logistic Regression	solver	1×10^4
	C	Sag
MLP	Solver	Adam
	Hidden layer size	100
	Learning rate	Constant
	Activation	Relu
	Shuffle	True
	Batch size	auto
NB	Binarize	0.0
	Class_prior	None
	alpha	1.0
	Fit_prior	True

Table 7. Performance Comparison of Machine Learning Method.

Method	MCC	SN (%)	SP (%)	ACC (%)
DNN	0.711	89.01	82.35	83.40
RF	0.626	74.65	87.24	82.21
SVM	0.522	61.44	88.17	77.50
MLP	0.607	88.75	70.79	81.37
KNN	0.621	96.61	58.67	81.44
Logistic Regression	0.229	77.38	44.80	64.35
NB	0.438	78.10	63.39	72.23
PNN	0.293	13.55	100	65.47

The DNN model can effectively deal with a dataset of a complex nature with high non-linearity because it utilizes hidden layers (multi-stack processing layers) with implied

back-propagation (weight optimization). In contrast, other machine learning approaches (conventional machine learning approach) use single-stack processing layers.

5.4. Comparison with Existing Models

In this section, we compare the performance of the proposed model with the existing classification models recently published in [19–21] using the original benchmark dataset. For this comparison, all the models were implemented using a single machine. Listed in Table 8 are the results obtained by the proposed model and existing classification models. The outcomes indicated that our proposed model outperformed three existing predictors. Compared to the existing predictor (PreAIP), our proposed model exhibited the highest accuracy, 83.40%.

Table 8. Comparison of proposed model performance to existing models.

Method	SN (%)	SP (%)	MCC	ACC (%)
AIPpred	75.80	71.10	0.460	73.00
PreAIP	63.20	90.30	0.566	79.50
Proposed Model	89.01	82.35	0.711	83.40

Furthermore, a meaningful comparison should consider other measurement metrics such as sensitivity, specificity, and Mathews Correlation Coefficient (MCC) with the combination of accuracy. A high value of these metrics for a model shows that the model is more accurate and stable. From Table 8, we can observe that the proposed model achieved high values for these metrics compared to the existing mentioned models. For example, our proposed model had the most significant MCC values of 0.71, compared to 0.566 for the PreAIP. These results indicate the significance and stability of the proposed model compared to its counterpart prediction models.

5.5. Performance Evaluation Using Replicated Dataset

We used replicated sequences in this part to analyze the proposed model performance. It's worth noting that the proposed model feature formulation module (Section 4.2) converts peptide sequences into a feature vector using a sequential technique. A considerable data situation results from continually repeating the constructed sequences (i.e., feature vector) to create a higher-dimensional feature vector. Accuracy-based measures cannot assess how well the suggested model performs since the dataset contains redundant and identical sequences. Therefore, we merely employ computational-based metrics to analyze the performance of the proposed model in this evaluation.

Several experiments were performed to evaluate the performance of the proposed model in terms of computation times on different numbers of sequences, and the results are displayed in Figure 4. Figure 4 shows that the proposed model performed better than the sequential implementation of a deep neural network in computation time using four physical processing nodes. The execution times of the sequential deep neural networks are highly increased with the increasing number of sequences. In contrast, the execution times of the parallel proposed model are slightly increased, as shown in the figure. The proposed model achieved better performance in computation efficiency due to parallel processing.

The proposed model scalability analysis is shown in Figure 5, which includes a variety of processing nodes and sequences. The proposed model execution time is significantly minimized as the number of processing nodes increases, as indicated in the figure. For example, when the proposed model is run on a single processing node, it takes 1380 s (i.e., execution time) to classify 12.32 million sequences, while when it is run on four processing nodes, it takes 410 s to classify the equal number of sequences. According to these findings, the execution time of the proposed model is lowered by 70.29% on a large number compared to a single machine. The proposed model is then tested for speed utilizing a range of processing nodes and sequences.

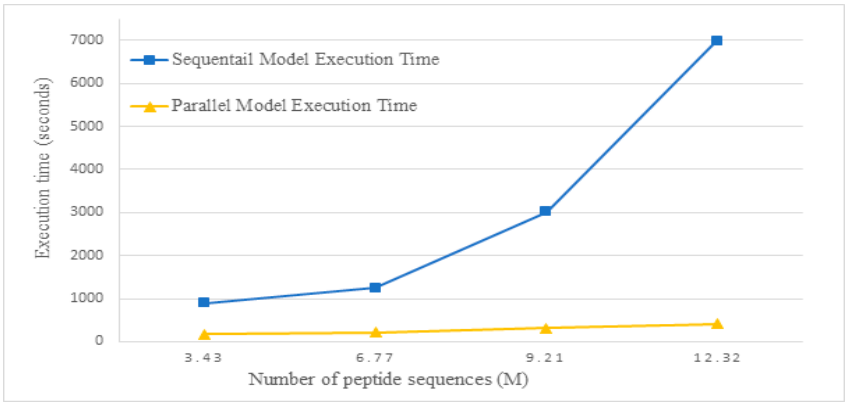

Figure 4. Proposed model efficiency analysis in terms of execution times utilizing a replicate dataset.

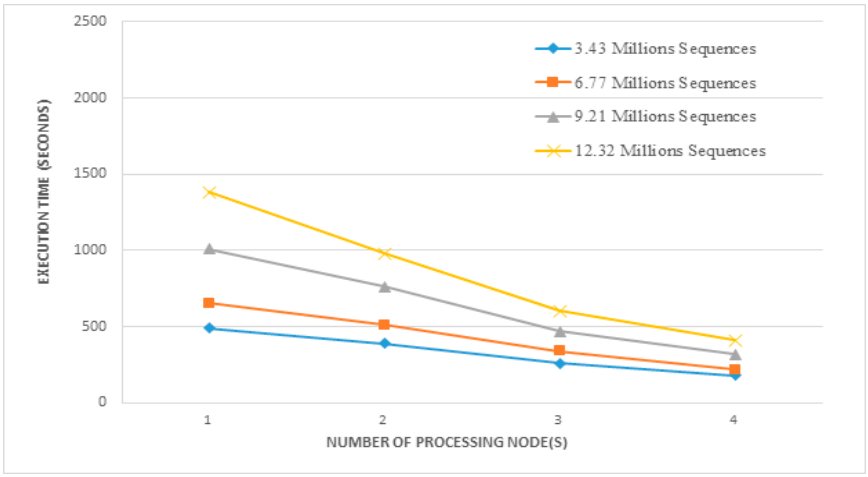

Figure 5. Proposed model Scalability analysis.

Finally, Figure 6 depicts the proposed model speedup using a replicated benchmark dataset. We assess the speed of the suggested model under various conditions. In the first scenario, the presented model achieved the maximum speed of 2.71 when processing 3.43 million sequences on four processing nodes. The proposed model achieved a speedup of 3.37 times while processing 12.32 million sequences on four processing nodes using the replicated benchmark dataset. These findings show that the suggested strategy is exceptionally scalable in terms of the number of processing nodes and the dataset size.

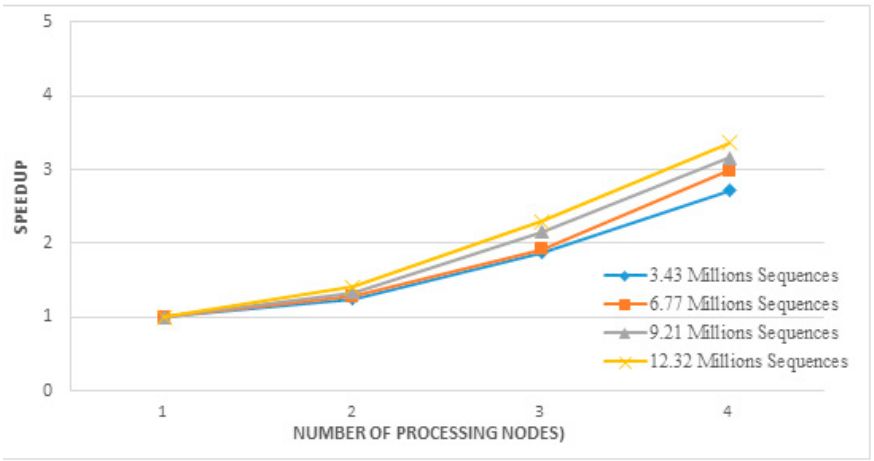

Figure 6. Speedup analysis of the proposed model.

Moreover, these results demonstrate that the proposed model outperforms others while processing many sequences (most significant speedup). This evaluation shows that the suggested strategy can handle many sequences. On the other hand, given a specific number of processing nodes, the suggested model never achieved the speedup predicted by Amdahl's law [71]. Several factors bring this on, including task launch overhead, network bandwidth, and cluster communication overhead. There has been a lot of discussion about this phenomenon [72].

6. Conclusions and Future Work

This paper presents a parallel DNN model for classifying large-scale peptide sequences as AIPs and Non-AIPs. The proposed model was built using the Spark programming model to achieve a parallel computation by partitioning and distributing sequences amongst a cluster of computer nodes. Furthermore, the proposed model formulated the sequence using APAAC and distance pair approaches. The proposed model performance was carefully examined. Due to the parallelization of both the data and the model, the experimental findings show that, compared to the sequential technique, the computation speed of the proposed model is more substantial, with no loss of overall model accuracy. Furthermore, the parallel DNN model is approvingly scalable due to the dataset size and the number of processing nodes.

The proposed model was employed using default parameter settings. The Spark framework has a lot of configurable parameters that have a substantial impact on its performance. In the future, we want to provide an approach that automatically optimizes framework configuration settings using the particle swarm optimization technique [73] and gene expression programming [74], improving the proposed model's computing speedup performance.

Author Contributions: Conceptualization, S.K. and N.I.; Methodology, S.K., M.K. and D.M.K.; Software, S.K., M.A.K., M.K. and S.A.A.; Validation, M.S.A.-R.; Formal analysis, S.K., M.K., N.I. and D.M.K.; Investigation, M.A.K. and M.S.A.-R.; Resources, M.A.K., N.I., S.A.A. and M.S.A.-R.; Data curation, M.A.K. and N.I.; Writing—original draft, S.K.; Writing—review & editing, M.K., N.I. and D.M.K.; Visualization, M.K., S.A.A., M.S.A.-R. and D.M.K.; Supervision, M.K.; Project administration, D.M.K.; Funding acquisition, S.A.A. All authors have read and agreed to the published version of the manuscript.

Funding: This Research is funded by Research Supporting Project Number (RSPD2023R585), King Saud University, Riyadh, Saudi Arabia.

Data Availability Statement: The data that we have used in the study is discussed in this paper.

Conflicts of Interest: The authors declare no conflict of interest to report regarding the present study.

References

1. Ferrero-Miliani, L.; Nielsen, O.H.; Andersen, P.S.; Girardin, S.E. Chronic Inflammation: Importance of NOD2 and NALP3 in Interleukin-1β Generation. *Clin. Exp. Immunol.* **2007**, *147*, 227–235. [CrossRef] [PubMed]
2. Asadullah, K.; Volk, H.D.; Sterry, W. Novel Immunotherapies for Psoriasis. *Trends Immunol.* **2002**, *23*, 47–53. [CrossRef] [PubMed]
3. Patterson, H.; Nibbs, R.; Mcinnes, I.; Siebert, S. Protein Kinase Inhibitors in the Treatment of Inflammatory and Autoimmune Diseases. *Clin. Exp. Immunol.* **2014**, *176*, 1–10. [CrossRef]
4. Corrigan, M.; Hirsch, G.M.; Oo, Y.H.; Adams, D.H. Autoimmune Hepatitis: An Approach to Disease Understanding and Management. *Br. Med. Bull.* **2015**, *114*, 181–191. [CrossRef]
5. Delgado, M.; Ganea, D. Anti-Inflammatory Neuropeptides: A New Class of Endogenous Immunoregulatory Agents. *Brain. Behav. Immun.* **2008**, *22*, 1146–1151. [CrossRef] [PubMed]
6. Zhao, L.; Wang, X.; Zhang, X.L.; Xie, Q.F. Purification and Identification of Anti-Inflammatory Peptides Derived from Simulated Gastrointestinal Digests of Velvet Antler Protein (*Cervus elaphus* Linnaeus). *J. Food Drug Anal.* **2016**, *24*, 376–384. [CrossRef]
7. Boismenu, R.; Chen, Y.; Chou, K.; Buelow, R. Orally Administered RDP58 Reduces the Severity of Dextran Sodium Sulphate Induced Colitis. *Ann. Rheum Dis.* **2002**, *61*, 19–25. [CrossRef]
8. Tabas, I.; Glass, C.K. Anti-Inflammatory Therapy in Chronic Disease: Challenges and Opportunities. *Science* **2013**, *339*, 166–172. [CrossRef]

9. Zhang, Q.; Miao, R.; Liu, T.; Huang, Z.; Peng, W.; Gan, B.; Zhang, X.; Tan, H. Biochemical Characterization of a Key Laccase-like Multicopper Oxidase of Artificially Cultivable Morchella Importuna Provides Insights into Plant-Litter Decomposition. *3 Biotech* **2019**, *9*, 171. [CrossRef]
10. Tan, H.; Tang, J.; Li, X.; Liu, T.; Miao, R.; Huang, Z.; Wang, Y.; Gan, B.; Peng, W. Biochemical Characterization of a Psychrophilic Phytase from an Artificially Cultivable Morel Morchella Importuna. *J. Microbiol. Biotechnol.* **2017**, *27*, 2180–2189. [CrossRef]
11. Chen, K.; Huo, Q. Scalable Training of Deep Learning Machines by Incremental Block Training with Intra-Block Parallel Optimization and Blockwise Model-Update Filtering. In Proceedings of the ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing-Proceedings, Shanghai, China, 20–25 March 2016; Volume 2016, pp. 5880–5884.
12. Maas, A.L.; Hannun, A.Y.; Lengerich, C.T.; Qi, P.; Jurafsky, D.; Ng, A.Y. Increasing Deep Neural Network Acoustic Model Size for Large Vocabulary Continuous Speech Recognition. *arXiv* **2014**, arXiv:1406.7806.
13. Strom, N. Scalable Distributed DNN Training Using Commodity GPU Cloud Computing. In Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH, Dresden, Germany, 6–10 September 2015.
14. Sinthong, P.; Mahadik, K.; Sarkhel, S.; Mitra, S. Scaling Dnn-Based Video Analysis by Coarse-Grained and Fine-Grained Parallelism. In Proceedings of the 2020 IEEE International Conference on Multimedia and Expo (ICME), Virtual, 6–10 July 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1–6.
15. White, T. *Hadoop: The Definitive Guide*, 1st ed.; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2009; Volume 1, ISBN 9780596521974.
16. Khan, M. *Hadoop Performance Modeling and Job Optimization for Big Data Analytics*; Brunel University: London, UK, 2015.
17. Cózar, J.; Marcelloni, F.; Gámez, J.A.; de la Ossa, L. Building Efficient Fuzzy Regression Trees for Large Scale and High Dimensional Problems. *J. Big Data* **2018**, *5*, 49. [CrossRef]
18. Zaharia, M.; Chowdhury, M.; Franklin, M.J.; Shenker, S.; Stoica, I. Spark: Cluster Computing with Working Sets. In Proceedings of the USENIX Conference on Hot Topics in Cloud Computing, Boston, MA, USA, 22–25 June 2010.
19. Gupta, S.; Sharma, A.K.; Shastri, V.; Madhu, M.K.; Sharma, V.K. Prediction of Anti-Inflammatory Proteins/Peptides: An Insilico Approach. *J. Transl. Med.* **2017**, *15*, 7. [CrossRef] [PubMed]
20. Manavalan, B.; Shin, T.H.; Kim, M.O.; Lee, G. AIPpred: Sequence-Based Prediction of Anti-Inflammatory Peptides Using Random Forest. *Front. Pharmacol.* **2018**, *9*, 276. [CrossRef]
21. Khatun, M.S.; Hasan, M.M.; Kurata, H. PreAIP: Computational Prediction of Anti-Inflammatory Peptides by Integrating Multiple Complementary Features. *Front. Genet.* **2019**, *10*, 129. [CrossRef] [PubMed]
22. Chakraverty, S.; Sahoo, D.M.; Mahato, N.R.; Chakraverty, S.; Sahoo, D.M.; Mahato, N.R. Perceptron Learning Rule. In *Concepts of Soft Computing*; Springer: Singapore, 2019; pp. 183–188.
23. Chen, C.-I.I.; Chang, G.W. A Two-Stage ADALINE for Harmonics and Interharmonics Measurement. In Proceedings of the 2010 5th IEEE Conference on Industrial Electronics and Applications, Taichung, Taiwan, 15–17 June 2010; pp. 340–345.
24. Wang, K.; Hoeksema, J.; Liang, C. PiRNN: Deep Learning Algorithm for PiRNA Prediction. *PeerJ* **2018**, *2018*, e5429. [CrossRef]
25. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. *Commun. ACM* **2017**, *60*, 84–90. [CrossRef]
26. Mikolov, T.; Kombrink, S.; Burget, L.; Cernock, J.; Khudanpur, S. Extensions of Recurrent Neural Network Language Model. In Proceedings of the 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Prague, Czech Republic, 22–27 May 2011; pp. 5528–5531.
27. Hinton, G.; Deng, L.; Yu, D.; Dahl, G.E.; Mohamed, A.R.; Jaitly, N.; Senior, A.; Vanhoucke, V.; Nguyen, P.; Sainath, T.N.; et al. Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups. *IEEE Signal Process. Mag.* **2012**, *29*, 82–97. [CrossRef]
28. Khan, S.; Khan, M.; Iqbal, N.; Rahman, M.A.A.; Karim, M.K.A. Deep-PiRNA: Bi-Layered Prediction Model for PIWI-Interacting RNA Using Discriminative Features. *Comput. Mater. Contin.* **2022**, *72*, 2243–2258. [CrossRef]
29. Khan, S.; Khan, M.; Iqbal, N.; Li, M.; Khan, D.M. Spark-Based Parallel Deep Neural Network Model for Classification of Large Scale RNAs into PiRNAs and Non-PiRNAs. *IEEE Access* **2020**, *8*, 136978–136991. [CrossRef]
30. Khan, S.; Khan, M.; Iqbal, N.; Khan, S.A.; Chou, K.-C. Prediction of PiRNAs and Their Function Based on Discriminative Intelligent Model Using Hybrid Features into Chou's PseKNC. *Chemom. Intell. Lab. Syst.* **2020**, *203*, 104056. [CrossRef]
31. Khan, S.; Khan, M.; Iqbal, N.; Hussain, T.; Khan, S.A.; Chou, K.-C. A Two-Level Computation Model Based on Deep Learning Algorithm for Identification of PiRNA and Their Functions via Chou's 5-Steps Rule. *Int. J. Pept. Res. Ther.* **2020**, *26*, 795–809. [CrossRef]
32. Khan, S.; Naeem, M.; Qiyas, M. Deep Intelligent Predictive Model for the Identification of Diabetes. *AIMS Math.* **2023**, *8*, 16446–16462. [CrossRef]
33. Akbar, S.; Khan, S.; Ali, F.; Hayat, M.; Qasim, M.; Gul, S. IHBP-DeepPSSM: Identifying Hormone Binding Proteins Using PsePSSM Based Evolutionary Features and Deep Learning Approach. *Chemom. Intell. Lab. Syst.* **2020**, *204*, 104103. [CrossRef]
34. Akbar, S.; Ahmad, A.; Hayat, M.; Rehman, A.U.; Khan, S.; Ali, F. IAtbP-Hyb-EnC: Prediction of Antitubercular Peptides via Heterogeneous Feature Representation and Genetic Algorithm Based Ensemble Learning Model. *Comput. Biol. Med.* **2021**, *137*, 104778. [CrossRef]
35. Ahmad, A.; Akbar, S.; Hayat, M.; Ali, F.; Khan, S.; Sohail, M. Identification of Antioxidant Proteins Using a Discriminative Intelligent Model of K-Spaced Amino Acid Pairs Based Descriptors Incorporating with Ensemble Feature Selection. *Biocybern. Biomed. Eng.* **2020**, *42*, 727–735. [CrossRef]

36. Ahmad, A.; Akbar, S.; Khan, S.; Hayat, M.; Ali, F.; Ahmed, A.; Tahir, M. Deep-AntiFP: Prediction of Antifungal Peptides Using Distanct Multi-Informative Features Incorporating with Deep Neural Networks. *Chemom. Intell. Lab. Syst.* **2021**, *208*, 104214. [CrossRef]
37. Akbar, S.; Hayat, M.; Tahir, M.; Khan, S.; Alarfaj, F.K. CACP-DeepGram: Classification of Anticancer Peptides via Deep Neural Network and Skip-Gram-Based Word Embedding Model. *Artif. Intell. Med.* **2022**, *131*, 102349. [CrossRef]
38. Akbar, S.; Ali, F.; Hayat, M.; Ahmad, A.; Khan, S.; Gul, S. Prediction of Antiviral Peptides Using Transform Evolutionary & SHAP Analysis Based Descriptors by Incorporation with Ensemble Learning Strategy. *Chemom. Intell. Lab. Syst.* **2022**, *230*, 104682. [CrossRef]
39. Khan, F.; Khan, M.; Iqbal, N.; Khan, S.; Muhammad Khan, D.; Khan, A.; Wei, D.-Q. Prediction of Recombination Spots Using Novel Hybrid Feature Extraction Method via Deep Learning Approach. *Front. Genet.* **2020**, *11*, 1052. [CrossRef]
40. Inayat, N.; Khan, M.; Iqbal, N.; Khan, S.; Raza, M.; Khan, D.M.; Khan, A.; Wei, D.Q. IEnhancer-DHF: Identification of Enhancers and Their Strengths Using Optimize Deep Neural Network with Multiple Features Extraction Methods. *IEEE Access* **2021**, *9*, 40783–40796. [CrossRef]
41. Kasper Fredenslund Computational Complexity of Neural Networks. Available online: https://kasperfred.com/series/computational-complexity/computationalcomplexity-of-neural-networks (accessed on 7 May 2023).
42. Vita, R.; Mahajan, S.; Overton, J.A.; Dhanda, S.K.; Martini, S.; Cantrell, J.R.; Wheeler, D.K.; Sette, A.; Peters, B. The Immune Epitope Database (IEDB): 2018 Update. *Nucleic Acids Res.* **2019**, *47*, D339–D343. [CrossRef] [PubMed]
43. Liu, B.; Yang, F.; Chou, K.C. 2L-PiRNA: A Two-Layer Ensemble Classifier for Identifying Piwi-Interacting RNAs and Their Function. *Mol. Ther.-Nucleic Acids* **2017**, *7*, 267–277. [CrossRef] [PubMed]
44. Liu, B.; Wu, H.; Chou, K.-C. Pse-in-One 2.0: An Improved Package of Web Servers for Generating Various Modes of Pseudo Components of DNA, RNA, and Protein Sequences. *Nat. Sci.* **2017**, *9*, 67–91. [CrossRef]
45. Gautheron, L.; Habrard, A.; Morvant, E.; Sebban, M. Metric Learning from Imbalanced Data. In Proceedings of the 31st IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2019, Portland, OR, USA, 4–6 November 2019; pp. 923–930. [CrossRef]
46. Farquad, M.A.H.; Bose, I. Preprocessing Unbalanced Data Using Support Vector Machine. *Decis. Support Syst.* **2012**, *53*, 226–233. [CrossRef]
47. Harliman, R.; Uchida, K. Data- and Algorithm-Hybrid Approach for Imbalanced Data Problems in Deep Neural Network. *Int. J. Mach. Learn. Comput.* **2018**, *8*, 208–213. [CrossRef]
48. Yousefian-Jazi, A.; Ryu, J.H.; Yoon, S.; Liu, J.J. Decision Support in Machine Vision System for Monitoring of TFT-LCD Glass Substrates Manufacturing. *J. Process Control* **2014**, *24*, 1015–1023. [CrossRef]
49. Kim, J.; Han, Y.; Lee, J. Data Imbalance Problem Solving for SMOTE Based Oversampling: Study on Fault Detection Prediction Model in Semiconductor Manufacturing Process. *Adv. Sci. Technol. Lett.* **2016**, *133*, 79–84. [CrossRef]
50. Batista, G.E.A.P.A.; Prati, R.C.; Monard, M.C. A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data. *ACM SIGKDD Explor. Newsl.* **2004**, *6*, 20–29. [CrossRef]
51. Goel, G.; Maguire, L.; Li, Y.; McLoone, S. *Evaluation of Sampling Methods for Learning from Imbalanced Data*; Lecture Notes in Computer Science, 7995 LNCS; Springer: Berlin/Heidelberg, Germany, 2013; pp. 392–401. [CrossRef]
52. Chen, T.; Shi, X.; Wong, Y.D. Key Feature Selection and Risk Prediction for Lane-Changing Behaviors Based on Vehicles' Trajectory Data. *Accid. Anal. Prev.* **2019**, *129*, 156–169. [CrossRef]
53. Wilson, D.L. Asymptotic Properties of Nearest Neighbor Rules Using Edited Data. *IEEE Trans. Syst. Man Cybern.* **1972**, *2*, 408–421. [CrossRef]
54. Guo, R.; Zhao, Y.; Zou, Q.; Fang, X.; Peng, S. Bioinformatics Applications on Apache Spark. *Gigascience* **2018**, *7*, giy098. [CrossRef]
55. Iqbal, N.; Khan, T.; Khan, M.; Hussain, T.; Hameed, T.; Bukhari, S.A.C. Neuromechanical Signal-Based Parallel and Scalable Model for Lower Limb Movement Recognition. *IEEE Sens. J.* **2021**, *21*, 16213–16221. [CrossRef]
56. Zaharia, M.; Chowdhury, M.; Das, T.; Dave, A.; Ma, J.; McCauley, M.; Franklin, M.J.; Shenker, S.; Stoica, I. Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing. In Proceedings of the NSDI 2012: 9th USENIX Symposium on Networked Systems Design and Implementation, San Jose, CA, USA, 25–27 April 2012; pp. 15–28.
57. Karau, H.; Warren, R. *High Performance Spark: Best Practices for Scaling & Optimizing Apache Spark*; Cutt, S., Ed.; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2017; Volume 15, ISBN 9781491943205.
58. Maqbool, H.F.; Husman, M.A.B.; Awad, M.I.; Abouhossein, A.; Iqbal, N.; Dehghani-Sanij, A.A. A Real-Time Gait Event Detection for Lower Limb Prosthesis Control and Evaluation. *IEEE Trans. Neural Syst. Rehabil. Eng.* **2017**, *25*, 1500–1509. [CrossRef] [PubMed]
59. Hussain, T.; Iqbal, N.; Maqbool, H.F.; Khan, M.; Awad, M.I.; Dehghani-Sanij, A.A. Intent Based Recognition of Walking and Ramp Activities for Amputee Using SEMG Based Lower Limb Prostheses. *Biocybern. Biomed. Eng.* **2020**, *40*, 1110–1123. [CrossRef]
60. Tsitsiklis, J.N.; Bertsekas, D.P.; Athans, M. Distributed Asynchronous Deterministic and Stochastic Gradient Optimization Algorithms. *IEEE Trans. Automat. Contr.* **1986**, *31*, 803–812. [CrossRef]
61. Chen, J.; Monga, R.; Bengio, S.; Józefowicz, R. Revisiting Distributed Synchronous {SGD}. *arXiv* **2016**, arXiv:1604.00981.
62. Cui, H.; Zhang, H.; Ganger, G.R.; Gibbons, P.B.; Xing, E.P. GeePS: Scalable Deep Learning on Distributed GPUs with a GPU-Specialized Parameter Server. In Proceedings of the Eleventh European Conference on Computer Systems-EuroSys '16, London, UK, 18–21 April 2016.

63. Sabooh, M.F.; Iqbal, N.; Khan, M.; Khan, M.; Maqbool, H.F. Identifying 5-Methylcytosine Sites in RNA Sequence Using Composite Encoding Feature into Chou's PseKNC. *J. Theor. Biol.* **2018**, *452*, 1–9. [CrossRef]
64. Specht, D.F. Probabilistic Neural Networks. *Neural Netw.* **1990**, *3*, 109–118. [CrossRef]
65. Guo, G.; Wang, H.; Bell, D.; Bi, Y.; Greer, K. *KNN Model-Based Approach in Classification*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2003; Volume 2888, pp. 986–996. [CrossRef]
66. Svetnik, V.; Liaw, A.; Tong, C.; Culberson, J.C.; Sheridan, R.P.; Feuston, B.P. Random Forest: A Classification and Regression Tool for Compound Classification and QSAR Modeling. *J. Chem. Inf. Comput. Sci.* **2003**, *43*, 1947–1958. [CrossRef]
67. Byun, H.; Lee, S.W. *Applications of Support Vector Machines for Pattern Recognition: A Survey*; Pattern Recognition with Support Vector Machines; Springer: Berlin/Heidelberg, Germany, 2002; pp. 213–236.
68. Taud, H.; Mas, J. Multilayer Perceptron (MLP). In *Geomatic Approaches for Modeling Land Change Scenarios*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 451–455.
69. Myaeng, S.H.; Han, K.S.; Rim, H.C. Some Effective Techniques for Naive Bayes Text Classification. *IEEE Trans. Knowl. Data Eng.* **2006**, *18*, 1457–1466. [CrossRef]
70. Wright, R.E. Logistic Regression. In *Reading and Understanding Multivariate Statistics*; American Psychological Association: Washington, DC, USA, 1995; pp. 217–244. ISBN 1-55798-273-2. (Paperback).
71. Amdahl, G.M. Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities. In Proceedings of the Spring Joint Computer Conference On-AFIPS '67 (Spring), Atlantic City, NJ, USA, 18–20 April 1967; ACM Press: New York, NY, USA, 1967; p. 483.
72. Khan, M.; Ashton, P.M.; Li, M.; Taylor, G.A.; Pisica, I.; Liu, J. Parallel Detrended Fluctuation Analysis for Fast Event Detection on Massive Pmu Data. *IEEE Trans. Smart Grid* **2015**, *6*, 360–368. [CrossRef]
73. Kennedy, J.; Eberhart, R.C.; Shi, Y. *Swarm Intelligence*; Elsevier: Amsterdam, The Netherlands, 2001; ISBN 9781558605954.
74. Ferreira, C. Gene Expression Programming: A New Adaptive Algorithm for Solving Problems. *Complex Syst.* **2001**, *13*, 87–129.