

# Deep learning optimization for drug-target interaction prediction in COVID-19 using graphic processing unit

Refianto Damai Darmawan<sup>1</sup>, Wisnu Ananta Kusuma<sup>1,2</sup>, Hendra Rahmawan<sup>1</sup>

<sup>1</sup>Department of Computer Science, Faculty of Mathematics and Natural Science, IPB University, Bogor, Indonesia

<sup>2</sup>Tropical Biopharmaca Research Center, IPB University, Bogor, Indonesia

## Article Info

### Article history:

Received Jul 13, 2022

Revised Sep 22, 2022

Accepted Oct 1, 2022

### Keywords:

Bioinformatics

Deep learning

Drug-target interaction

Graphic processing unit

High performance computing

## ABSTRACT

The exponentially increasing bioinformatics data raised a new problem: the computation time length. The amount of data that needs to be processed is not matched by an increase in hardware performance, so it burdens researchers on computation time, especially on drug-target interaction prediction, where the computational complexity is exponential. One of the focuses of high-performance computing research is the utilization of the graphics processing unit (GPU) to perform multiple computations in parallel. This study aims to see how well the GPU performs when used for deep learning problems to predict drug-target interactions. This study used the gold-standard data in drug-target interaction (DTI) and the coronavirus disease (COVID-19) dataset. The stages of this research are data acquisition, data preprocessing, model building, hyperparameter tuning, performance evaluation and COVID-19 dataset testing. The results of this study indicate that the use of GPU in deep learning models can speed up the training process by 100 times. In addition, the hyperparameter tuning process is also greatly helped by the presence of the GPU because it can make the process up to 55 times faster. When tested using the COVID-19 dataset, the model showed good performance with 76% accuracy, 74% F-measure and a speed-up value of 179.

*This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.*



## Corresponding Author:

Wisnu Ananta Kusuma

Department of Computer Science, Faculty of Mathematics and Natural Science, IPB University

Wing 20 Level 5 Meranti Road, Dramaga District, Bogor 16680, Indonesia

Email: ananta@apps.ipb.ac.id

## 1. INTRODUCTION

Nowadays, the demand of efficient computing is increasing due to the abundance of data, especially in the field of bioinformatics. High performance computing (HPC) is a collection of integrated computing environments and programming techniques that can help solve large-scale problems [1]. It is something that is currently applied in chemoinformatics; Heikamp and Bajaroth [2] have conducted an in-depth review of the field of chemoinformatics [3], compound screening [4] and the challenges that will be faced in the future. One of their concerns is computational efficiency. This happens because the amount of available data is increasing exponentially, so standard computational techniques will take a long time to process the data. Several solutions are also proposed in this article, including the use of a graphics processing unit (GPU) [5]–[7], parallel programming such as MapReduce [8] and a particular computer for molecular dynamics simulation [9].

One of the large-scale problems in the computing world is in bioinformatics or, more precisely, in the prediction of drug-target interactions [10]. Prediction of drug-target interaction (DTI) is the first step that is usually done in the drug development process or commonly called drug discovery. The use of conventional

methods is considered less able to accommodate the trend in biological data which is increasing yearly. A topic often discussed in HPC is GPU utilization as a parallel computing tool [11]–[14]. DTI is an interaction between the drug and the target (protein). DTI prediction is carried out to find new indications for existing drugs which can be achieved by identifying new interactions between drugs and target proteins. The *in silico* (computational) prediction of DTI has several challenges in two aspects. The first aspect is that the volume of available data on drugs and proteins is growing exponentially. The second aspect is that drug-target interaction pairs are pretty rare. In addition, it is quite challenging to select negative samples because no negative interactions of the drug-target interactions have been verified experimentally. These conditions need some unique approaches to integrate the computational power of GPU to predict a DTI efficiently.

To date, two approaches have been used to predict new interactions of drugs and known targets, namely network-based and learning-based approaches [15]. Many kinds of research have been done to develop the DTI model. Bahi and Batouche [15] proposed using a deep semi-supervised learning model called DeepSS-DTI to accurately predict potential novel drug-target interactions using large-scale drug-target data. This model uses a stacked autoencoder to initialize the weights in the deep learning model. The proposed deep learning model managed to get better performance than the support vector machine (SVM) [16], nearest neighbor [17], ensemble classifier [18], random forest (RF) [19] and decision tree (DT) [20]. Ezzat *et al.* [21] tried to use ensemble learning and dimensionality reduction methods to predict similarity-based DTI. They reduced the large dimension of the drug-target data using singular value decomposition (SVD). Then the learning model uses the same type of positive data but different negative data. The resulting area under ROC curve (AUC) value outperforms the DT, RF, and SVM models.

Using parallelization schemes to speed-up large-scale data processing has become a common practice. We used the hardware devices of the GPU. Although initially, this device was devoted to improving graphics rendering performance. This study uses GPU parallelization to generate a DTI prediction model using a deep learning method and existing drug-target interaction data. This approach is based on several reasons. The first reason is to see how much performance gain when using the GPU during the model training process. The second reason is to determine the performance of deep learning methods in processing and predicting chemical compound data. The deep learning model was chosen because it is considered to have an easy parallelization scheme so that it can maximize the role of the GPU. The development of the deep learning model in this study was built from scratch and did not use existing libraries. The reason for the manual creation of deep learning models without using a library is the versatility of the code, so that if we want to apply it to other devices such as supercomputers, it can be quickly done.

This kind of study is unique because, to date, no research precisely measures the performance of graphics processing unit (GPU) in a deep learning model using available drug-target interaction data. Several studies that have been carried out using deep learning models or compound molecular data are the development of new molecular fingerprints [22], prediction of drug-target interactions [23], new quantitative structure analysis relationship methods [24] and development of the stacked auto-encoder deep neural network model [25]. Efficient parallelization schemes are needed to be developed and applied to the current DTI data. To better understand the role of GPU parallelization in deep learning of drug-target interaction prediction, this study aims to measure the level of effectiveness of GPU in its integration with deep learning models suitable for HPC development. The result of this research can be used by researchers, chemists and the general public to conduct an initial screening of their chemical compounds before further testing on drug-target interaction.

## 2. METHOD

This research was divided into several stages. This division of stages aimed to make it easier for researchers and readers to carry out research work so that other researchers can duplicate the same thing if they want to do similar research. The research began with preparation of research data, followed by preprocess of the data, perform hyperparameter tuning, perform deep learning modeling with and without GPU and the last was model performance evaluation.

### 2.1. Research data

This study used the gold-standard data in DTI from Yamanishi *et al.* [26]. The information contained in this dataset is taken from several databases such as KEGG BRITE [27], BRENDA [28], SuperTarget [29] and DrugBank [30]. This dataset was further divided into four sections that represent the interaction of ligands and proteins, namely enzymes (E), ion channels (IC), G protein-coupled receptors (GPCR) and nuclear receptors (NR). Table 1 shows the statistics of each dataset. We can see the difference in the number of drugs, target proteins and interactions involved in each dataset.

Table 1. Statistics for drug-target interaction datasets

Statistics	E	IC	GPCR	NR
Number of drugs	445	210	223	54
Number of target protein (Total in human genome)	664 (2,741)	204 (292)	95 (757)	26 (49)
Number of drug-target interactions	2,926	1,476	635	90

## 2.2. Data preprocessing

The objective of data preprocessing is to produce numerical feature vectors of drug-target pairs so that they can be used as input data for machine learning models [23]. Sulistiawan *et al.* [25] state that the best pair of numeric vectors was the PubChem fingerprint paired with dipeptide composition (DC). The data preprocessing flow can be seen in Figure 1, which briefly describe how the data was handled from the data acquisition step until it was ready to be used as an input to the deep learning model.

The existing drug data was processed using the simplified molecular input line entry system (SMILES) descriptor. SMILES, the simplified molecular input line entry system, is a chemical notation system designed for modern chemical information processing [31]. SMILES represents a compound in the form of alphabet, numeric and symbols that describe the interactions between its constituent atoms. An example of a SMILES representation for the compound acetic acid or *CH<sub>3</sub>COOH* is *CC(=O)O*. The next step was to change the SMILES descriptor to the PubChem fingerprint.

A fingerprint is a way to represent chemical compounds into numbers 1 and 0, according to the presence or absence of the criteria set on the compound. The PubChem fingerprint contains 881 structural keys. A structural key is a structural criterion assigned to each bit, indicating whether or not a structural criterion exists in the compound in question. For example, since PubChem has a bit position from 0 to 880, bit 0 addresses the structural criterion of whether a molecule has more than equal to 4 hydrogen atoms or not. The PubChem fingerprint representation is a sequence of 0 or 1 bits totaling 881 units. The SMILES descriptor and the PubChem fingerprint were obtained using the “Rcpi” library in R.

FASTA is a text-based file format representing the amino acid sequences that make up a protein. For the protein target data, FASTA files were used before being processed into feature proteins. The existing protein data was converted into the FASTA descriptor form. Proteins with UniProt entry names [32] are searched for their FASTA files using the UniProt RESTful service and a UniProtKB query using Python. In addition, proteins whose GDP ID was known got a FASTA file from the protein data bank [33] website.

After obtaining the PubChem fingerprint of drugs and the dipeptide composition of targets, the final step was to combine them into a single compound-protein feature vector. The protein feature used was DC. DC is a value with a range of 0 to 1, indicating the percentage of a dipeptide occurrence in the amino acid or protein sequence. Dipeptides are peptide or amino acid pairs calculated using the sliding window. Since the number of amino acids is 20, the number of dipeptides present is 202 or 400 pairs. The total number of features this combined feature had been 1,281.

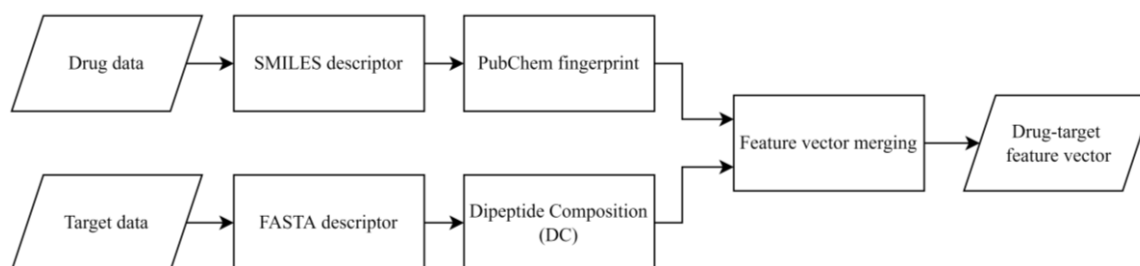


Figure 1. Data preprocessing flow to obtain drug-target feature vector

## 2.3. Build prediction model

Deep neural networks (DNN), or neural networks in general, have a parallelization concept known as embarrassingly parallel. The meaning of embarrassingly parallel is that neural network algorithms can be easily managed to run in parallel, so the researcher does not have to think about complicated techniques to achieve it. The neural network parallelization process can occur during feed forward and backpropagation. It is because each node in a layer does not need information from other nodes in the same layer, so the process can run in parallel [34].

Two types of models were created to compare their performance. The first model was a deep learning model built with central processing unit (CPU) or sequential. The second model was a deep learning model built on GPU or parallel. The meaning of parallel is simultaneously working on several computational processes at the same time in this model.

The first model only used the CPU in the modeling process because the processes were sequential. The processing unit processed each node in a layer one by one so that node  $n$  could not perform computations before the node  $(n-1)$  was finished and produced output. Figure 2 provides a visualization of the CPU model created. In this model, there are input layers, hidden layers and output layers. In Figures 2(a)-(d), we can see that to process a single layer, the sequential process can only run one node at a time, and if we consider a time in the node calculation process as  $t$ , then we needed  $5t$  to process a single layer that consists of 5 nodes.

The visualization given in Figure 2 illustrates the processes that occur in the CPU model. At the input layer, all data in each feature was spread to all nodes in the first hidden layer. At the input layer, there was no computation process because the task of this layer was only to spread data to all nodes in the next layer. In the first hidden layer, nodes in each layer received data from the previous layer within the feed-forward process. Figures 2(a)-2(d) show that a single thread could only do computation for one node at a time.

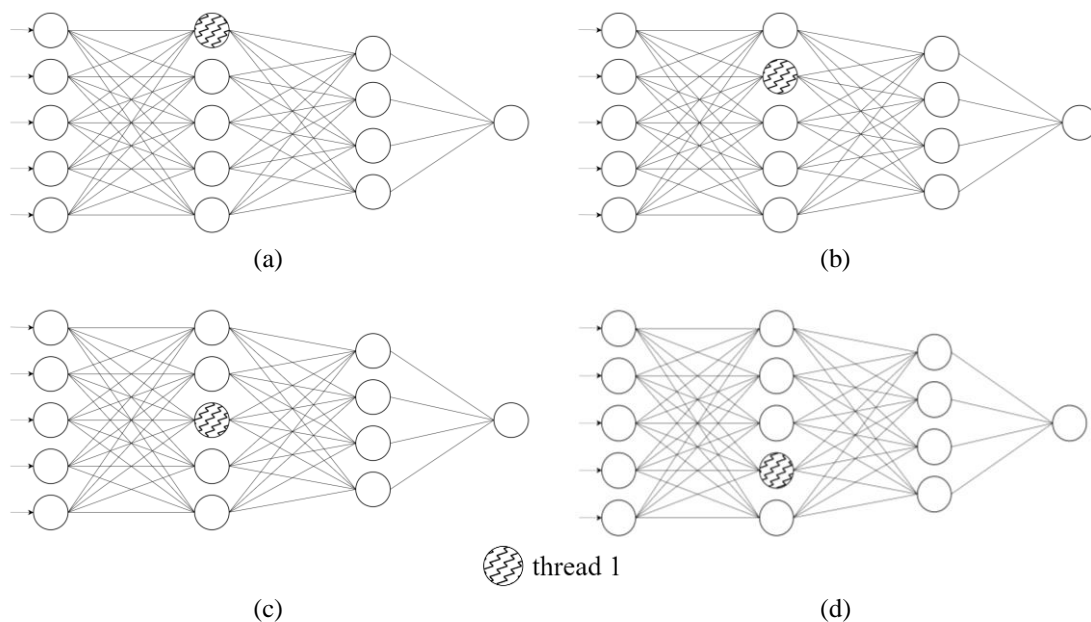


Figure 2. CPU model visualization with single thread sequential computation on (a) processing the first node in the first hidden layer, (b) processing the second node in the first hidden layer, (c) processing third node in the first hidden layer, and (d) processing fourth node in the first hidden layer

The role of the CPU was to receive data and perform computations for each node in each hidden layer and output layer. It started from the first node in the first hidden layer to the last node in the last hidden layer. All nodes in the hidden layer only have one node with pattern because there was only one thread on the CPU that did all the computing. It means that the CPU did the computations on each node. Logically, this process took a long time because the number of nodes in each layer reached hundreds. Deep learning models, both in the feed-forward and backtracking processes, used simple mathematical calculations, so it is hoped that CPU threads could do the computations faster.

The second model used a GPU in the modeling process. Here, the CPU only played a role in sending data between the hard drive and the GPU. The process that occurred in this GPU model was a parallel process, doing several computations simultaneously at a time. Figure 3 provides a visualization of the GPU model created. We can see that because of parallelization, all nodes in the same layer could be calculated simultaneously with several threads. If we consider a time in the node calculation process as  $t$ , then we only need  $2t$  to process two hidden layers in a deep learning network. Figures 3(a)-3(b) shown that each node on the same layer could be computed simultaneously by allocating different thread on it.

The visualization given in Figure 3 illustrates the processes that occur in the GPU model. The process that occurred at the input layer only spread the data of each feature to every node in the next layer. The difference between this model and the previous model was visible when entering the first hidden layer. In the first hidden layer, the data received by each node was processed according to the deep learning algorithm.

In the first hidden layer, each node in the hidden layer was processed by a different thread so that the computing process could run simultaneously. After the process was complete, the same concept of parallelization was applied to the next hidden layer and remained the same until it reached the output layer. This process can save time because the work in each layer is done in parallel. However, the downside of this process was that GPU threads have lower computational capabilities than CPU threads, so the parallelization scheme needs to be adequately planned to have optimal results.

Block is a group of threads in a GPU that can access the same memory cache. In this research, the block size used was 1,024. This means that in one block, there are 1,024 threads with identifiers 0 to 1,023. In this study, we used the default value of the block size to speed up the research process and focus on the data processing capabilities of each thread between CPU and GPU. If the desired number of threads exceeded the block size, another block with the appropriate number of threads was called to fulfil the need for the remaining threads.

The training process used a train-test split by doing class balancing first. Class balancing is a method to equalize the proportion of positive class (existing interaction) and negative class (no interaction) in data. This method is essential because bioinformatics data often contain negative classes far exceeding the number of positive classes. Train-test split is a data training method that separates the dataset into training data and test data according to the specified proportion. In this case, it was 75:25, i.e., 75% for training data and 25% for testing data.

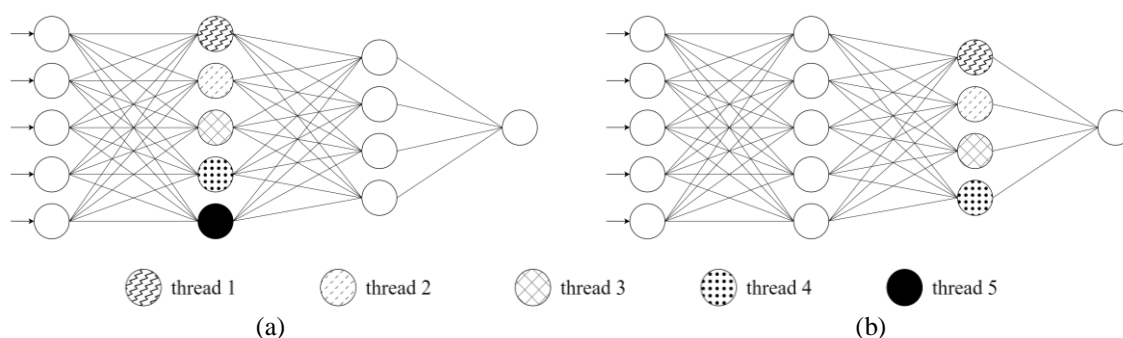


Figure 3. GPU model visualization with multiple thread parallel computation with (a) thread 1, 2, 3, 4 and 5 processing node 1, 2, 3, 4, 5, respectively in the first hidden layer and (b) thread 1, 2, 3, and 4 processing node 1, 2, 3, 4, respectively in the second hidden layer

## 2.4. Hyperparameter tuning

Hyperparameter tuning purpose is to select optimal hyperparameters for a machine learning model [35]. In this study, we used the grid search technique for hyperparameter tuning [36] because this technique had the same number of hyperparameter combinations to be carried out on CPU and GPU models. The grid search is a hyperparameter tuning approach that methodically builds and evaluates a model for each combination of hyperparameters written in a grid [37]. Although not the fastest algorithm, grid search was chosen because it clearly showed the comparison of CPU and GPU model performance in the case of hyperparameter tuning. The list of tested hyperparameters can be seen in Table 2. Each hyperparameter had two kinds of values which need to be selected.

Hyperparameter	List of values
Number of nodes in $HL_0$ ( $HN_0$ )	300, 500
Number of nodes in $HL_n$ ( $HN_n$ )	$\frac{1}{2} \times HN_{n-1}$ , $\frac{2}{3} \times HN_{n-1}$
Number of Hidden Layer (HL)	2, 3
Learning rate	0.01, 0.005
Activation function	ReLU, Sigmoid

This hyperparameter tuning process was carried out on two models, the CPU and GPU, to compare the performance achieved. The GPU parallelization process occurred in the learning stages, where the deep learning model leveraged hundreds of threads within the GPU to perform simple arithmetic operations such as addition, subtraction, multiplication and division. The process was adjusted to the stages in each node or neuron, such as the weight multiplication stage, the sum of all inputs, activation functions and backpropagation.

## 2.5. Performance evaluation

Performance was calculated through testing on test data for the CPU model and GPU model. We used accuracy (1), precision (3), recall (2), F-measure (4), receiver operating characteristic (ROC) curve and speed-up (5) as performance metrics. When viewed from the confusion matrix [38] in binary classification, there are four types of model prediction results when compared to the actual value, namely true positive (TP), true negative (TN), false positive (FP) and false negative (FN) [39]. Based on those four values, the accuracy value [40] can be defined as the number of correct predictions divided by the number of all data that went through the prediction process. The recall value is defined as the accuracy of the model in predicting the positive class by minimizing the positive data that is wrongly predicted [41]. In other words, recall is the number of correct positive predictions divided by the number of all data whose actual values are positive (2). The precision value is defined as the accuracy of the model in predicting the positive class by minimizing the negative data that is wrongly predicted. In other words, precision is the number of correct positive predictions divided by the number of all positive predicted data (3) [41]. The F-measure value is defined as a weighted comparison of the average precision and recall values [42]. F-measure pays attention to precision and recall values to measure the performance of the minority class as a whole (4) [10]. The speed-up value is a value that compares the processing time required between processes running with the CPU and processes running with the GPU (5) [43]. The process whose speed-up value is calculated is the training and testing process. In (7),  $t_{cpu}$  is the time needed to run the process with the CPU, while  $t_{gpu}$  is the time needed to run the process with the GPU. If the value is more than 1, then the processing time with the CPU is longer than the processing time with the GPU. Conversely, if the value is less than 1, then the processing time with the GPU is longer than the processing time with the CPU.

$$accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (1)$$

$$recall = \frac{TP}{TP+FN} \quad (2)$$

$$precision = \frac{TP}{TP+FP} \quad (3)$$

$$F - measure = \frac{2 \times precision \times recall}{precision + recall} \quad (4)$$

$$speed - up = \frac{t_{cpu}}{t_{gpu}} \quad (5)$$

## 2.6. Testing with COVID-19 dataset

Testing must be done to determine the model's performance with a more relevant dataset. In this study, the drug-protein interaction dataset from Sulistiawan *et al.* [25] was chosen as the test data because of the novelty of the data and the shape of the dataset which tends to be similar to the dataset of Yamanishi *et al.* [26]. Dataset of Sulistiawan *et al.* [25] is a collection of positive interaction data from drug compounds that can bind to proteins associated with COVID-19 disease in humans. This dataset took data from the research of Li and Clercq [44] and Wu *et al.* [45]. Li and Clercq [44] conducted a study on reusing existing or currently used antiviral agents in diseases similar to COVID-19. The antiviral agents referred to in this study are antiviral agents for human immunodeficiency virus (HIV), hepatitis C virus (HCV) and influenza. This research is based on the experience of drug discovery in other diseases caused by the corona virus, namely severe acute respiratory syndrome (SARS) and middle east respiratory syndrome (MERS). Wu *et al.* [45] analyzed the proteins in the SARS-CoV-2 gene and screened them with drug compound data in the ZINC database [46]. This is done to find compounds that can bind to these proteins so that they are helpful for inhibiting the metabolism and life cycle of the SARS-CoV-2 virus.

Dataset of Sulistiawan *et al.* [25] has 712 positive interactions between compounds and proteins. The number of unique compounds in this dataset is 123 compounds. The number of unique proteins present in this dataset is 325 proteins. Like the previous dataset, negative interactions were obtained from data on

proteins and compounds present in the dataset but did not have positive interactions. Seeing the number of positive interactions that are similar to the GPCR dataset, the results of the hyperparameter tuning used are the best hyperparameters in the GPCR dataset. Tests were conducted using the CPU and GPU to compare the performance of the two models. The best-performing model will be used to predict the possible binding of negative interactions in the COVID-19 dataset. The flow of the drug-target interaction prediction process can be seen in Figure 4, which briefly describes how to get the training data for the model and which data will be predicted.

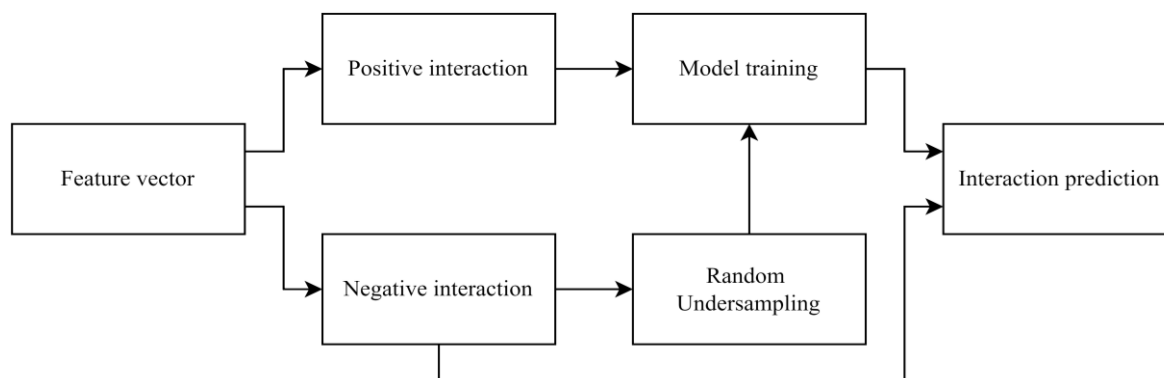


Figure 4. Process flow of drug-target interaction prediction in COVID-19 dataset

The feature vectors obtained from the data preprocessing results of the COVID-19 dataset were divided into two: data with positive interactions and data with negative interactions. Data with negative interactions needed to be randomized undersampling before being used in training of deep learning models so that the proportion of data with positive interactions and data with negative interactions was balanced. This balance of data aimed to improve the model's performance in the predictability of the data. After the model has been trained, all data with negative interactions would be predicted by the model.

### 3. RESULTS AND DISCUSSION

#### 3.1. Data preprocessing

Data from Yamanishi *et al.* [26] are pairs of compounds and target proteins that interact positively in four datasets: E, IC, GPCR and NR. Each of the four datasets has a unique list of compounds and proteins. Because the data in each dataset is a type of positive interaction between the drugs and the target protein, negative interactions were obtained from combining drugs and proteins that were not in the dataset. Because the number of positive interactions was much less than the number of negative interactions, stratified random sampling was used for each dataset so that the ratio between positive and negative interactions is 1:1, i.e., the number of positive interactions equals the negative ones. This was done so that the training process runs optimally.

The next step was to convert the compound data into 881 bits of PubChem fingerprint via the SMILES descriptor and change the protein data to 400 features of dipeptide composition or DC. After the fingerprint and DC data were collected, the selected drug-target data was converted into fingerprint-DC data totaling 1,281 features along with the type of interaction, whether positive (value of 1) or negative (value of 0). After the data was ready to be treated as input into the deep learning model, the data was further divided in the proportion of 75:25 as training data and test data. Table 3 provides comprehensive information about the data contained in each dataset. We can see that each dataset has different number of training and testing data. This difference was the key comparison to evaluate whether or not the deep learning model could be applied in various DTI data. The complete source code of this research is available at [47].

Table 3. The number of drugs, protein targets, training data, and test data in each dataset

Dataset	Total drug	Total target	Training data	Testing data
E	445	664	4,389	1,463
IC	210	204	2,214	738
GPCR	223	95	952	318
NR	54	26	135	45

### 3.2. Hyperparameter tuning

The hyperparameter tuning step was performed on two identical models: the CPU version with C++ and the GPU version with CUDA. This step was carried out using the grid search method to test all possible combinations on the model. Table 4 shows the optimal hyperparameter results based on the level of accuracy and recall at the hyperparameter tuning stage. The runtime column shows the most significant difference between the two models in the same dataset. This result indicates that the use of GPU can increase the processing time significantly.

Table 4. The number of drugs, protein targets, training data and test data in each dataset

Dataset	Model	HN <sub>0</sub>	HN <sub>n</sub>	HL	Learning rate	Activation function	Runtime (s)	Speed-up
E	CPU	300	$\frac{1}{2} \times \text{HN}_{n-1}$	2	0.005	Sigmoid	155,173.74	-
	GPU	300	$\frac{1}{2} \times \text{HN}_{n-1}$	2	0.010	Sigmoid	1,348.28	115.09
IC	CPU	300	$\frac{1}{2} \times \text{HN}_{n-1}$	3	0.010	ReLU	78,304.19	-
	GPU	300	$\frac{2}{3} \times \text{HN}_{n-1}$	2	0.010	ReLU	701.94	111.55
GPCR	CPU	500	$\frac{1}{2} \times \text{HN}_{n-1}$	2	0.010	Sigmoid	33,542.97	-
	GPU	300	$\frac{1}{2} \times \text{HN}_{n-1}$	2	0.005	Sigmoid	328.33	102.16
NR	CPU	500	$\frac{2}{3} \times \text{HN}_{n-1}$	2	0.010	Sigmoid	4,769.28	-
	GPU	300	$\frac{2}{3} \times \text{HN}_{n-1}$	2	0.005	Sigmoid	85.67	55.67

The results in Table 4 show that the selected hyperparameters for each dataset only had a slight difference between the CPU and the GPU model. A significant difference was found in the runtime between models as indicated by the speed-up value (comparison of CPU runtime with GPU runtime) which was quite large. The lowest speed-up value was the NR dataset with a value of 55.67, then above it was the GPCR dataset with a value of 102.16, followed by the IC dataset with a value of 111.55 and the largest was the speed-up from the E dataset with a value of 115.09. This speed-up value was positively correlated with the amount of training data in each dataset, where dataset E had the largest amount of training data, then below that there are IC, GPCR and NR datasets, respectively. This showed that the larger the dataset that needs to be trained, the better the GPU optimization capability in building deep learning models.

### 3.3. Performance evaluation

Models that already have optimal hyperparameters were tested with test data that has been prepared. Table 5 shows each dataset's accuracy, recall, precision, F-measure and speed-up values. The significant difference can be seen in the runtime column between two models of the same dataset. Again, the runtime and speed-up values indicate that GPU models showed a better performance than CPU models.

Table 5. Accuracy, recall, precision, F-measure and speed-up values in each dataset

Dataset	Model	Accuracy (%)	Recall (%)	Precision (%)	F-measure (%)	Runtime (s)	Speed-up
E	CPU	76.90	78.69	75.99	77.32	3,233.88	-
	GPU	77.38	77.46	77.35	77.41	30.64	105.54
IC	CPU	70.88	75.34	69.67	72.40	1,654.47	-
	GPU	71.68	72.90	71.16	72.02	16.56	99.91
GPCR	CPU	64.47	66.67	63.86	65.23	1,310.59	-
	GPU	64.78	62.26	65.56	63.87	7.58	172.90
NR	CPU	68.89	56.52	76.47	65.00	193.83	-
	GPU	60.00	56.52	61.90	59.09	2.11	91.86

The accuracy of each model ranges from 60%-80%. The accuracy value that appears could be said to be pretty good, considering that the model structure was quite simple and had not implemented the latest techniques from deep learning models such as optimizer and dropout rate. The lowest accuracy value in the model was the NR dataset, namely 60.00% and 68.89%. Most likely this was due to the lack of available training data, so the model had not reached the optimal weight when the training process was completed. The highest accuracy value was in dataset E, with values of 76.90% and 77.38%. The accuracy value tends to increase as a dataset's training data increases.

In the drug prediction step, the model was considered better if it has a high recall value to produce better drug candidates. The recall value that appeared was included in the reasonably good category, which is all in the range of 50%-80%. This recall value indicated how likely the model was to produce a false negative prediction (FN). The lowest recall value was found in the NR dataset, which was 56.52%. This was most likely due to the small training data, so the model did not yet have optimal weight. The model owned the



highest recall value was dataset E, namely 77.46% and 78.69%. This happened because the training data was quite large, so the model could better identify drug data with a potentially positive value (capable of binding), so the resulting FN value was low.

In order to avoid the emergence of drug candidates that cannot interact with proteins, we measured the model's precision. The overall precision value was quite good, which was in the range of 60%-80%. A high precision value indicates the model's low false positive (FP) prediction. The smallest precision value was in the NR dataset, with a value of 61.90% for the GPU model. This showed that the model with the NR dataset had a higher probability of producing FP predictions than other models. The highest precision values were in the model with dataset E, namely 75.99% and 77.35%. This shows that the model in dataset E had the possibility of producing fewer FP values than the model in other datasets.

Furthermore, the overall F-measure value was in the reasonably good category, ranging from 59-80%. The F-measure value shows how likely the model is to produce FN and FP values. In drug-target interaction prediction, the higher the F-measure value, the better the model. The lowest F-measure value was found in the model with the NR dataset, 59.09% and 65.00%. This value indicates that the model's performance was quite good, although there was the possibility of producing wrong predictions. The highest F-measure value was found in the model with dataset E, which is 77.32% and 77.41%. This showed that this model was the best among other models and the probability of making wrong predictions was much smaller.

Regarding the computational performance, we evaluated the runtime and speed-up. The overall runtime values showed significantly different values across the four datasets between the CPU and GPU models. In dataset E, which has 4,389 training data, it takes 3,233.88 seconds to train the model, while the GPU model with the same dataset takes 30.64 seconds to train the model. When converted to minutes, the CPU model in dataset E takes 53.9 minutes. The IC, GPCR and NR datasets also showed significant differences in the runtime of the CPU model and GPU model, although the time is relatively faster than the E dataset. This was due to the different amounts of training data in the four datasets. The less training data available, the faster the training process occurred.

Moreover, the speed-up value showed a relatively high number in the four datasets. The speed-up value for dataset E showed the number 105.54. This indicated that using the GPU model, the training time was 105.54 times faster than the CPU model. The speed-up value for the IC dataset showed the number 99.91. This indicated that the GPU model on the IC dataset requires 99.91 times faster training time than the CPU model. The speed-up value in the GPCR dataset showed the number 172.90. This indicated that the GPU model in the GPCR dataset requires 172.90 times faster training than the CPU model. The speed-up value in the NR dataset showed the number 91.86. This indicated that the GPU model on the NR dataset requires 91.86 times faster training time than the CPU model. Overall, using the GPU model could speed up the model training process by up to 90 times.

### 3.4. Testing with COVID-19 dataset

The first test is to look at the model's performance against the COVID-19 dataset as a whole. The results of performance testing can be seen in Table 6. A significant difference could be seen in the runtime column, which indicate that GPU model could do DTI prediction much faster than CPU model. Table 6 shows that the performance of the CPU and GPU models was quite similar, with the GPU model slightly better in terms of accuracy, recall and F-measure. A quite big difference was seen at model's runtime. The CPU model takes 179 times longer than the GPU model. This showed that using GPU in deep learning models could shorten training times by up to 179 times, with model performance likely to match the performance of the CPU model. The test was continued by predicting drug-target interactions on the COVID-19 dataset. The seven most considerable possible interaction outcomes are shown in Table 7. The prediction score value showed that these drug-target interactions were predicted to be highly possible by the deep learning model.

Table 6. Accuracy, recall, precision, F-measure and speed-up values in COVID-19 dataset

Dataset	Model	Accuracy (%)	Recall (%)	Precision (%)	F-measure (%)	Runtime (s)	Speed-up
COVID-19	CPU	76.12	67.42	81.63	73.85	2,952.82	-
	GPU	76.40	69.10	80.92	74.55	16.41	179.99

Arsentrioxide, also known as arsenic trioxide, currently has been widely used in India as health supplement, immune-booster and COVID-19 preventive drug in the form of homeopathic remedy [48]. Arsentrioxide in this case was processed and consumed in the form of Arsenic Album 30C. Further study needs to be done in order to prescribe the exact amount of Arsenic Album 30C to patients in order to prevent

acute severe liver injury as stated in [48]. Although Theruvath *et al.* [48] mentioned some harms of toxicity in a therapy of COVID-19 patients with arsenitrioxide, they concluded that the sources of toxicity are the poor manufacturing practices, use of concentrated tincture formulations, and adulteration and contamination of homeopathic remedies, and not the compound itself that act as a harmful medication. Arthrocin, also known as sulindac, was recently suggested as potential drug repurposing in COVID-19 therapy in [49] by using molecular dynamic simulation and docking analysis of NF- $\kappa$ B protein binding.

NMDE4\_HUMAN is a component of glutamate receptor N-methyl-D-aspartate (NMDA) that can be used as an entry of SARS-CoV-2 into specific cell populations [50]. MRP2\_HUMAN is an ATP-dependent transporter of the ATP-binding cassette (ABC) family that binds and hydrolyzes ATP to enable active transport of various substrates including potential drugs for COVID-19 [51]. DPOL\_HHV11 is a protein that replicates viral genomic DNA of human herpesvirus. This protein is worth to be further studied because there is possible association between COVID-19 and herpes simplex virus [52]. Q13746\_HUMAN is a mRNA of acute lymphocytic leukemia patients that have high mortality rates when in contact with COVID-19 [53]. ERBB2\_HUMAN is a receptor tyrosine-protein kinase erB-2 which several studies have pointed out that a correlation between SARS-CoV-2 viruses and dysregulations of signaling pathways activated by tyrosine-protein kinase receptors can be established [54]. PYRD\_HUMAN is a protein that catalyze the conversion of dihydroorotate to orotate with quinone as electron acceptor in the inner layer of mitochondria, which becomes highly vulnerable in COVID-19 infected cells, and vulnerability increases with age [55]. 6VSB:A is a prefusion 2019-nCoV spike glycoprotein which has a role for virus attachment, fusion and entry into the host cell [56].

The prediction results in Table 7 indicate that Arsenitrioxide can bind to NMDE4\_HUMAN, DPOL\_HHV11, Q13746\_HUMAN, ERBB2\_HUMAN, PYRD\_HUMAN and 6VSB:A, while Arthrocin can bind to MRP2\_HUMAN. For target proteins that are in humans, further studies need to be carried out to know the impact of these interactions and what their effects are in the healing process of COVID-19. For target proteins other than those found in humans, namely DPOL\_HHV11 and 6VSB:A, it can be concluded that Arsenitrioxide has a high probability of interfering with the metabolism of these viruses, so the results of this study can be used as a reference in vitro and in vitro studies.

Table 7. Top seven results of drug-target interaction prediction in COVID-19 dataset

Drug	Protein target	Prediction score
Arsenitrioxide	NMDE4_HUMAN	0.928608
Arthrocin	MRP2_HUMAN	0.928425
Arsenitrioxide	DPOL_HHV11	0.928398
Arsenitrioxide	Q13746_HUMAN	0.928371
Arsenitrioxide	ERBB2_HUMAN	0.928365
Arsenitrioxide	PYRD_HUMAN	0.928348
Arsenitrioxide	6VSB:A	0.928274

#### 4. CONCLUSION

This study showed that the use of GPU in deep learning models for drug-target interaction cases gives good results. The results showed a significant increase when performing hyperparameter tuning and during model training. The values of accuracy, recall, precision and F-measure that do not differ much in the CPU and GPU models in the same dataset indicate that the difference is only due to the different weight initialization values and the performance of the CPU and GPU models is approximately the same.

The use of GPUs, proven to accelerate the training process of deep learning models, should be used as a basis for further research on the use of GPUs in bioinformatics. Because the use of GPU is not limited to personal computers (PC), bioinformatics research centers that want to process extensive data should use GPU either in PC, servers, or HPC so that the process of hyperparameter tuning and training on deep learning models can be done more quickly.

#### ACKNOWLEDGEMENTS

This research was supported by the Ministry of Education, Culture, Research and Technology under Competitive Research Grant from Directorate of Higher Education, Indonesia, 2022, contract no. 3643/IT3.L1/PT.01.03/P/B2022. We acknowledge support from Tropical Biopharmaca Research Center, IPB University on this research.




## REFERENCES

- [1] C. A. Lee, S. D. Gasster, A. Plaza, C.-I. Chang, and B. Huang, "Recent developments in high performance computing for remote sensing: a review," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 4, no. 3, pp. 508–527, Sep. 2011, doi: 10.1109/JSTARS.2011.2162643.
- [2] K. Heikamp and J. Bajorath, "The future of virtual compound screening," *Chemical Biology and Drug Design*, vol. 81, no. 1, pp. 33–40, 2013, doi: 10.1111/cbdd.12054.
- [3] Y.-C. Lo, S. E. Rensi, W. Tornig, and R. B. Altman, "Machine learning in chemoinformatics and drug discovery," *Drug Discovery Today*, vol. 23, no. 8, pp. 1538–1546, Aug. 2018, doi: 10.1016/j.drudis.2018.05.010.
- [4] C. Eurtivong and J. Reynisson, "The development of a weighted index to optimise compound libraries for high throughput screening," *Molecular Informatics*, vol. 38, no. 3, Mar. 2019, doi: 10.1002/minf.201800068.
- [5] Q. Liao, J. Wang, and I. A. Watson, "Accelerating two algorithms for large-scale compound selection on GPUs," *Journal of Chemical Information and Modeling*, vol. 51, no. 5, pp. 1017–1024, May 2011, doi: 10.1021/ci200061p.
- [6] P. Liu, D. K. Agrafiotis, D. N. Rassokhin, and E. Yang, "Accelerating chemical database searching using graphics processing units," *Journal of Chemical Information and Modeling*, vol. 51, no. 8, pp. 1807–1816, Aug. 2011, doi: 10.1021/ci200164g.
- [7] C. Ma, L. Wang, and X.-Q. Xie, "GPU accelerated chemical similarity calculation for compound library comparison," *Journal of Chemical Information and Modeling*, vol. 51, no. 7, pp. 1521–1527, Jul. 2011, doi: 10.1021/ci1004948.
- [8] N. Maleki, A. M. Rahmani, and M. Conti, "MapReduce: an infrastructure review and research insights," *The Journal of Supercomputing*, vol. 75, no. 10, pp. 6934–7002, Oct. 2019, doi: 10.1007/s11227-019-02907-5.
- [9] C. Wu, T. Geng, C. Yang, V. Sachdeva, W. Sherman, and M. Herbordt, "A communication-efficient multi-chip design for range-limited molecular dynamics," in *2020 IEEE High Performance Extreme Computing Conference (HPEC)*, Sep. 2020, pp. 1–8, doi: 10.1109/HPEC43674.2020.9286146.
- [10] L. Kai-Biao, W. Wei, R. K. Lai, and L. Ping, "Imbalance data classification algorithm based on SVM and clustering function," in *2014 9th International Conference on Computer Science and Education*, Aug. 2014, pp. 544–548, doi: 10.1109/ICCSE.2014.6926521.
- [11] G. Patnaik and K. Obenschain, "Using GPU on HPC applications to satisfy low-power computational requirement," in *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, Jan. 2010, pp. 1–7, doi: 10.2514/6.2010-524.
- [12] L. Shi, H. Chen, J. Sun, and K. Li, "vCUDA: GPU-accelerated high-performance computing in virtual machines," *IEEE Transactions on Computers*, vol. 61, no. 6, pp. 804–816, Jun. 2012, doi: 10.1109/TC.2011.112.
- [13] N. DeBardeleben *et al.*, "GPU behavior on a large HPC cluster," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8374 LNCS, 2014, pp. 680–689, doi: 10.1007/978-3-642-54420-0\_66.
- [14] D. Tiwari *et al.*, "Understanding GPU errors on large-scale HPC systems and the implications for system design and operation," in *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*, Feb. 2015, pp. 331–342, doi: 10.1109/HPCA.2015.7056044.
- [15] M. Bahi and M. Batouche, "Drug-target interaction prediction in drug repositioning based on deep semi-supervised learning," in *IFIP Advances in Information and Communication Technology*, vol. 522, Springer International Publishing, 2018, pp. 302–313, doi: 10.1007/978-3-319-89743-1\_27.
- [16] J. Cervantes, F. Garcia-Lamont, L. Rodríguez-Mazahua, and A. Lopez, "A comprehensive survey on support vector machine classification: Applications, challenges and trends," *Neurocomputing*, vol. 408, pp. 189–215, Sep. 2020, doi: 10.1016/j.neucom.2019.10.118.
- [17] J. Gou, H. Ma, W. Ou, S. Zeng, Y. Rao, and H. Yang, "A generalized mean distance-based k-nearest neighbor classifier," *Expert Systems with Applications*, vol. 115, pp. 356–372, Jan. 2019, doi: 10.1016/j.eswa.2018.08.021.
- [18] Y. Zhiwen, L. Le, L. Jiming, and H. Guoqiang, "Hybrid adaptive classifier ensemble," *IEEE Transactions on Cybernetics*, vol. 45, no. 2, pp. 177–190, Feb. 2015, doi: 10.1109/TCYB.2014.2322195.
- [19] M. Belgiu and L. Drăguț, "Random forest in remote sensing: A review of applications and future directions," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 114, pp. 24–31, Apr. 2016, doi: 10.1016/j.isprsjprs.2016.01.011.
- [20] Y. Y. Song and Y. Lu, "Decision tree methods: applications for classification and prediction," *Shanghai Archives of Psychiatry*, vol. 27, no. 2, pp. 130–135, 2015.
- [21] A. Ezzat, M. Wu, X.-L. Li, and C.-K. Kwok, "Drug-target interaction prediction using ensemble learning and dimensionality reduction," *Methods*, vol. 129, no. 2017, pp. 81–88, Oct. 2017, doi: 10.1016/j.ymeth.2017.05.016.
- [22] A. Buin, H. Y. Chiang, S. A. Gadsden, and F. A. Alderson, "Permutationally invariant deep learning approach to molecular fingerprinting with application to compound mixtures," *Journal of Chemical Information and Modeling*, vol. 61, no. 2, pp. 631–640, Feb. 2021, doi: 10.1021/acs.jcim.0c01097.
- [23] S. Redkar, S. Mondal, A. Joseph, and K. S. Hareesha, "A machine learning approach for drug-target interaction prediction using wrapper feature selection and class balancing," *Molecular Informatics*, vol. 39, no. 5, May 2020, doi: 10.1002/minf.201900062.
- [24] J. Ma, R. P. Sheridan, A. Liaw, G. E. Dahl, and V. Svetnik, "Deep neural nets as a method for quantitative structure-activity relationships," *Journal of Chemical Information and Modeling*, vol. 55, no. 2, pp. 263–274, Feb. 2015, doi: 10.1021/ci500747n.
- [25] F. Sulistiawan, W. A. Kusuma, N. S. Ramadhanti, and A. Tedjo, "Drug-target interaction prediction in coronavirus disease 2019 case using deep semi-supervised learning model," in *2020 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, Oct. 2020, pp. 83–88, doi: 10.1109/ICACSIS51025.2020.9263241.
- [26] Y. Yamanishi, M. Araki, A. Gutteridge, W. Honda, and M. Kanehisa, "Prediction of drug-target interaction networks from the integration of chemical and genomic spaces," *Bioinformatics*, vol. 24, no. 13, pp. 232–240, Jul. 2008, doi: 10.1093/bioinformatics/btn162.
- [27] M. Kanehisa, M. Furumichi, Y. Sato, M. Ishiguro-Watanabe, and M. Tanabe, "KEGG: integrating viruses and cellular organisms," *Nucleic Acids Research*, vol. 49, no. D1, pp. 545–551, Jan. 2021, doi: 10.1093/nar/gkaa970.
- [28] A. Chang *et al.*, "BRENDA, the ELIXIR core data resource in 2021: new developments and updates," *Nucleic Acids Research*, vol. 49, no. D1, pp. 498–508, Jan. 2021, doi: 10.1093/nar/gkaa1025.
- [29] N. Hecker *et al.*, "SuperTarget goes quantitative: update on drug-target interactions," *Nucleic Acids Research*, vol. 40, no. D1, pp. 1113–1117, Jan. 2012, doi: 10.1093/nar/gkr912.
- [30] D. S. Wishart *et al.*, "DrugBank 5.0: a major update to the DrugBank database for 2018," *Nucleic Acids Research*, vol. 46, no. D1, pp. 1074–1082, Jan. 2018, doi: 10.1093/nar/gkx1037.
- [31] M. Hirohara, Y. Saito, Y. Koda, K. Sato, and Y. Sakakibara, "Convolutional neural network based on SMILES representation of compounds for detecting chemical motif," *BMC Bioinformatics*, vol. 19, no. S19, Dec. 2018, doi: 10.1186/s12859-018-2523-5.




- [32] A. Bateman, "UniProt: a worldwide hub of protein knowledge," *Nucleic Acids Research*, vol. 47, no. D1, pp. 506–515, Jan. 2019, doi: 10.1093/nar/gky1049.
- [33] S. K. Burley *et al.*, "RCSB Protein Data Bank: powerful new tools for exploring 3D structures of biological macromolecules for basic and applied research and education in fundamental biology, biomedicine, biotechnology, bioengineering and energy sciences," *Nucleic Acids Research*, vol. 49, no. D1, pp. 437–451, Jan. 2021, doi: 10.1093/nar/gkaa1038.
- [34] N. S. Sattar and S. Anfuazzaman, "Data parallel large sparse deep neural network on GPU," in *2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, May 2020, vol. 423, pp. 1–9, doi: 10.1109/IPDPSW50202.2020.00170.
- [35] K. Shankar, Y. Zhang, Y. Liu, L. Wu, and C.-H. Chen, "Hyperparameter tuning deep learning for diabetic retinopathy fundus image classification," *IEEE Access*, vol. 8, pp. 118164–118173, 2020, doi: 10.1109/ACCESS.2020.3005152.
- [36] J. Wong, T. Manderson, M. Abrahamowicz, D. L. Buckeridge, and R. Tamblyn, "Can hyperparameter tuning improve the performance of a super learner?," *Epidemiology*, vol. 30, no. 4, pp. 521–531, Jul. 2019, doi: 10.1097/EDE.0000000000001027.
- [37] G. S. K. Ranjan, A. K. Verma, and S. Radhika, "K-nearest neighbors and grid search CV based real time fault monitoring system for industries," in *2019 IEEE 5th International Conference for Convergence in Technology (I2CT)*, Mar. 2019, pp. 1–5, doi: 10.1109/I2CT45611.2019.9033691.
- [38] M. Hasnain, M. F. Pasha, I. Ghani, M. Imran, M. Y. Alzahrani, and R. Budiarto, "Evaluating trust prediction and confusion matrix measures for web services ranking," *IEEE Access*, vol. 8, pp. 90847–90861, 2020, doi: 10.1109/ACCESS.2020.2994222.
- [39] N. J. Gogtay and U. M. Thatte, "Statistical evaluation of diagnostic tests (part 1): Sensitivity, specificity, positive and negative predictive values," *Journal of Association of Physicians of India*, vol. 65, pp. 80–84, 2017.
- [40] A. Luque, A. Carrasco, A. Martín, and J. R. Lama, "Exploring symmetry of binary classification performance metrics," *Symmetry*, vol. 11, no. 1, Jan. 2019, doi: 10.3390/sym11010047.
- [41] H. R. Sofaer, J. A. Hoeting, and C. S. Jarnevich, "The area under the precision-recall curve as a performance metric for rare binary events," *Methods in Ecology and Evolution*, vol. 10, no. 4, pp. 565–577, Apr. 2019, doi: 10.1111/2041-210X.13140.
- [42] D. Hand and P. Christen, "A note on using the F-measure for evaluating record linkage algorithms," *Statistics and Computing*, vol. 28, no. 3, pp. 539–547, May 2018, doi: 10.1007/s11222-017-9746-6.
- [43] P. Cunningham and S. J. Delany, "K-nearest neighbour classifiers-a tutorial," *ACM Computing Surveys*, vol. 54, no. 6, pp. 1–25, Jul. 2022, doi: 10.1145/3459665.
- [44] G. Li and E. De Clercq, "Therapeutic options for the 2019 novel coronavirus (2019-nCoV)," *Nature Reviews Drug Discovery*, vol. 19, no. 3, pp. 149–150, Mar. 2020, doi: 10.1038/d41573-020-00016-0.
- [45] C. Wu *et al.*, "Analysis of therapeutic targets for SARS-CoV-2 and discovery of potential drugs by computational methods," *Acta Pharmaceutica Sinica B*, vol. 10, no. 5, pp. 766–788, May 2020, doi: 10.1016/j.apsb.2020.02.008.
- [46] J. J. Irwin, T. Sterling, M. M. Mysinger, E. S. Bolstad, and R. G. Coleman, "ZINC: A free tool to discover chemistry for biology," *Journal of Chemical Information and Modeling*, vol. 52, no. 7, pp. 1757–1768, Jul. 2012, doi: 10.1021/ci3001277.
- [47] TropBRC-BioinfoLab, "Drug-target interaction prediction with CUDA," *GitHub*. <https://github.com/TropBRC-BioinfoLab/dti-gpu/> (accessed Dec. 23, 2022).
- [48] A. H. Theruvath, R. Raveendran, and C. A. Philips, "Dangerous placebo during the COVID-19 pandemic: a series of homoeopathic arsenicum album-induced liver injury," *Cureus*, vol. 14, no. 6, Jun. 2022, doi: 10.7759/cureus.26062.
- [49] S. Ahmad, "Molecular dynamics simulation and docking analysis of NF- $\kappa$ B protein binding with sulindac acid," *Bioinformation*, vol. 18, no. 3, pp. 170–179, Mar. 2022, doi: 10.6026/97320630018170.
- [50] R. F. Butterworth, "Potential for the repurposing of adamantane antivirals for COVID-19," *Drugs in R&D*, vol. 21, no. 3, pp. 267–272, Sep. 2021, doi: 10.1007/s40268-021-00351-6.
- [51] A. Telbisz *et al.*, "Interactions of potential anti-COVID-19 compounds with multispecific ABC and OATP drug transporters," *Pharmaceutics*, vol. 13, no. 1, Jan. 2021, doi: 10.3390/pharmaceutics13010081.
- [52] P. Bond, "Ethnicity and the relationship between covid-19 and the herpes simplex viruses," *Medical Hypotheses*, vol. 146, Jan. 2021, doi: 10.1016/j.mehy.2020.110447.
- [53] D. Buyuktas *et al.*, "COVID-19 infection in patients with acute leukemia; Istanbul experience," *American journal of blood research*, vol. 11, no. 4, pp. 427–437, 2021.
- [54] O.-S. Purcaru *et al.*, "The interference between SARS-CoV-2 and tyrosine kinase receptor signaling in cancer," *International Journal of Molecular Sciences*, vol. 22, no. 9, May 2021, doi: 10.3390/ijms22094830.
- [55] R. Ganji and P. H. Reddy, "Impact of COVID-19 on mitochondrial-based immunity in aging and age-related diseases," *Frontiers in Aging Neuroscience*, vol. 12, pp. 1–14, Jan. 2021, doi: 10.3389/fnagi.2020.614650.
- [56] L. Duan, Q. Zheng, H. Zhang, Y. Niu, Y. Lou, and H. Wang, "The SARS-CoV-2 spike glycoprotein biosynthesis, structure, function, and antigenicity: implications for the design of spike-based vaccine immunogens," *Frontiers in Immunology*, vol. 11, pp. 1–12, Oct. 2020, doi: 10.3389/fimmu.2020.576622.

## BIOGRAPHIES OF AUTHORS






**Refianto Damai Darmawan**    received the B.Sc. degree in computer science from IPB University, Thailand, in 2021. Currently, he is a student at the Department of Computer Science, IPB University. His research interests include artificial intelligence, machine learning, deep learning, graphic processing unit optimization, drug-target interaction and network. He can be contacted at email: rdamaid@gmail.com.



**Wisnu Ananta Kusuma**    received his B.Sc. and M.Eng. from Bandung Institute of Technology, Indonesia. He received Ph.D. degree in Computer Science, especially in the field of Bioinformatics from Tokyo Institute of Technology, Japan. He is also received research grants; get involved in some competitive and collaborative research at IPB. Currently He is lecturer in Computer Science Department, Bogor Agriculture University, coordinator of Bioinformatics Working Group, Faculty of Mathematics and Natural Science, IPB, and Executive Secretary of Tropical Biopharmaca Research Center. His current research interest is machine learning, high performance computing and bioinformatics, especially in the field of next generation sequencing and analysis, SNP identification and analysis, metagenome analysis and network pharmacology. He can be contacted at email: [ananta@apps.ipb.ac.id](mailto:ananta@apps.ipb.ac.id).



**Hendra Rahmawan**    received his B.Sc. from IPB University, Indonesia, in 2004. He received his M.Eng. and Ph.D. degrees in Bandung Institute of Technology, Indonesia, in 2009 and 2018, respectively. Currently he is a lecturer in Computer Science Department, Faculty of Mathematics and Natural Science, Bogor Agricultural University. His current research interest is high performance computing, computer architecture, networking and parallel computation. He can be contacted at [hrahmawan@apps.ipb.ac.id](mailto:hrahmawan@apps.ipb.ac.id).