Improving stroke diagnosis accuracy using hyperparameter optimized deep learning

Tessy Badriyah [a,1,*], Dimas Bagus Santoso [a,2], Iwan Syarif [a,3], Daisy Rahmania Syarif [b,4]

[a] Politeknik Elektronika Negeri Surabaya (PENS), Indonesia
[b] University of Cologne, Germany
[1] tessy@pens.ac.id; [2] santoso.db@gmail.com; [3] iwanarif@pens.ac.id; [4] dsyarif@smail.uni-koeln.de
* corresponding author

ARTICLE INFO

Article history
Received July 21, 2019
Revised August 28, 2019
Accepted November 16, 2019
Available online November 17, 2019

Keywords
Feature selection
Deep learning
Hyperparameter optimization
Random search
Bayesian optimization

ABSTRACT

Stroke may cause death for anyone, including youngsters. One of the early stroke detection techniques is a Computerized Tomography (CT) scan. This research aimed to optimize hyperparameter in Deep Learning, Random Search and Bayesian Optimization for determining the right hyperparameter. The CT scan images were processed by scaling, grayscale, smoothing, thresholding, and morphological operation. Then, the images feature was extracted by the Gray Level Co-occurrence Matrix (GLCM). This research was performed a feature selection to select relevant features for reducing computing expenses, while deep learning based on hyperparameter setting was used to the data classification process. The experiment results showed that the Random Search had the best accuracy, while Bayesian Optimization excelled in optimization time.

1. Introduction

Cerebrovascular stroke or injury (CVA) is a loss of brain function caused by the sudden cessation of blood supply to parts of the brain. It is a condition that arises due to circulatory disorders in the brain, causes a person suffering from paralysis or death [1]. Stroke recognition is difficult because people do not regularly check up their brain and heart conditions [2]. The general diagnosis procedure uses Computed Tomography (CT) scans, Magnetic Resonance Imaging (MRI) and Electrocardiogram (EKG or ECG) [3].

Several studies in the health sector develop preprocessing on CT Scan image data to obtain better results. As was done by Singh and Gupta [4] to detect cases of classification of lung cancer, previously, the image was processed and produced texture features (GLCM) and statistical features. It used 7 (seven) different approaches, namely KNN, SVM, Decision Tree, Naïve Bayes, SGD, Random Forest and MLP (one type of Deep Learning architecture). From the results of the classification of medical image datasets containing 15,750 images, with the distribution of 6,910 are classed as benign and 8,840 for malignant classes. They obtain the highest accuracy of 88.55% using the MLP approach. Likewise, Marbun *et al.* [2] used CT brain scan images to detect strokes with preprocessing consisting of gray scaling, scaling, cloning, and then segmentation, namely binary image formation using thresholding. For classification, they used the Convolutional Neural Network method and obtained an accuracy value of 90%. Hence, the use of a blooming approach such as deep learning, could beneficial for stroke diagnosis using CT-scan.

Deep Learning is a variant of machine learning based on Neural Networks. It has many hidden layers that have the ability to learn data representations or features automatically. As other Neural Networks, in general, the Deep Learning architecture consists of visible and hidden layers where the weight of each perceptron unit is optimized using the backpropagation algorithm [5].

The use of Deep Learning in the health sector has gained more attention [6], Deep Learning produces impressive results in the areas of Speech Recognition [7], Computer Vision [8] and Natural Languages in recent times. However, Deep Learning has several drawbacks in terms of its framework. Some researchers use Deep Learning to solve their research problems and obtain satisfactory results. Hung *et al.* [3] used an Electronic medical claims (EMCs) database of 800,000 patients to compare DNN with three other approaches to predict stroke in five years. The results show that DNN and gradient boosting decision tree (GBDT) have the same accuracy as compared to Logistic Regression and Support Vector Machine. DNN can be optimized by using patient data that is less than GBDT. Using a more extended period of EMC data can help improve predictive quality. The use of the Deep Learning method in the health dataset was also carried out by Assodiky *et al.* [9], although the results of an Electrocardiogram (EKG) to detect Arrhythmia are sometimes difficult to observe and often cause diagnostic errors that may lead to death. However, a powerful Deep Learning approach can lead to improved diagnosis, with experimental results showing the best accuracy of 76.51%.

The effectiveness of Deep Learning ultimately relies on the implementation of hyperparameter. Determining the value of each hyperparameter requires one's value judgment. Hence a standardized approach is needed to set hyperparameter in Deep Learning. Some researchers use other approaches to optimize hyperparameters in Deep Learning. Research conducted by Qolomany *et al.* [4] shows the success of the application of Deep Learning by optimizing parameters using the Particle Swarm Optimization (PSO) method. PSO is very efficient in adjusting the number of optimal hidden layers and neurons. The results of the PSO experiment show that the search time for the right parameters can be 77% -85% faster than the search for the same parameters using manual search methods and Grid Search. Another way to optimize deep learning is also reported by Kingma and Ba [10] who used an efficient gradient descent algorithm named “Adam” and have been reported to have great results. The name Adam itself is implemented as the "Adam" optimizer in the Keras library in Python. On the other hand, Bergstra and Bengio [11] employed the Bayesian Optimization approach using the Gaussian Process for hyperparameter searches. The result shows that Bayesian Optimization is better at finding the right hyperparameters in the CIFAR-10 benchmarking dataset. The successful use of Bayesian optimization was also reported by Snoek *et al.* [12] who successfully used Bayesian Optimization in the Gaussian Process in its modeling. Other papers show the success of implementing hyperparameter in various fields, including what was done by Cui and Bai [13]; Di Francescomarino *et al.* [14]; Le *et al.* [15]; Martinez-De-Pison *et al.* [16]; Balaprakash *et al.* [17]; Neary [18]; Borgli *et al.* [19]; Talathi [20]; Vo *et al.* [21]; Dong *et al.* [22]; Yao *et al.* [23]; Yoo [24]; Candelieri *et al.* [25]; Laanaya *et al.* [26]; Laref *et al.* [27]; Strijov and Weber [28]; Tellez *et al.* [29]; Tsirikoglou *et al.* [30]; Mantovani *et al.* [31]; and Lee *et al.* [32].

Related works previously mentioned are in line with our previous research: using the Deep Learning method to diagnose Stroke by doing optimization at Hyperparameter. For the process of diagnosing stroke itself, in the hospital, medical staff must make a careful diagnosis because it can be classified into two types, namely ischemic stroke and hemorrhagic stroke, with both requiring different treatments. To speed up CT Scan image analysis, a computational system is needed to help speed up early detection of stroke. The main contribution of this research is the use of hyperparameter optimization in the Deep Learning method on CT scan data which could improve the effectiveness of the stroke detection results.

2. Method

The system design of the research can be seen in Fig. 1. the system has three main phases, namely, data preparation, data preprocessing, and hyperparameter optimization.

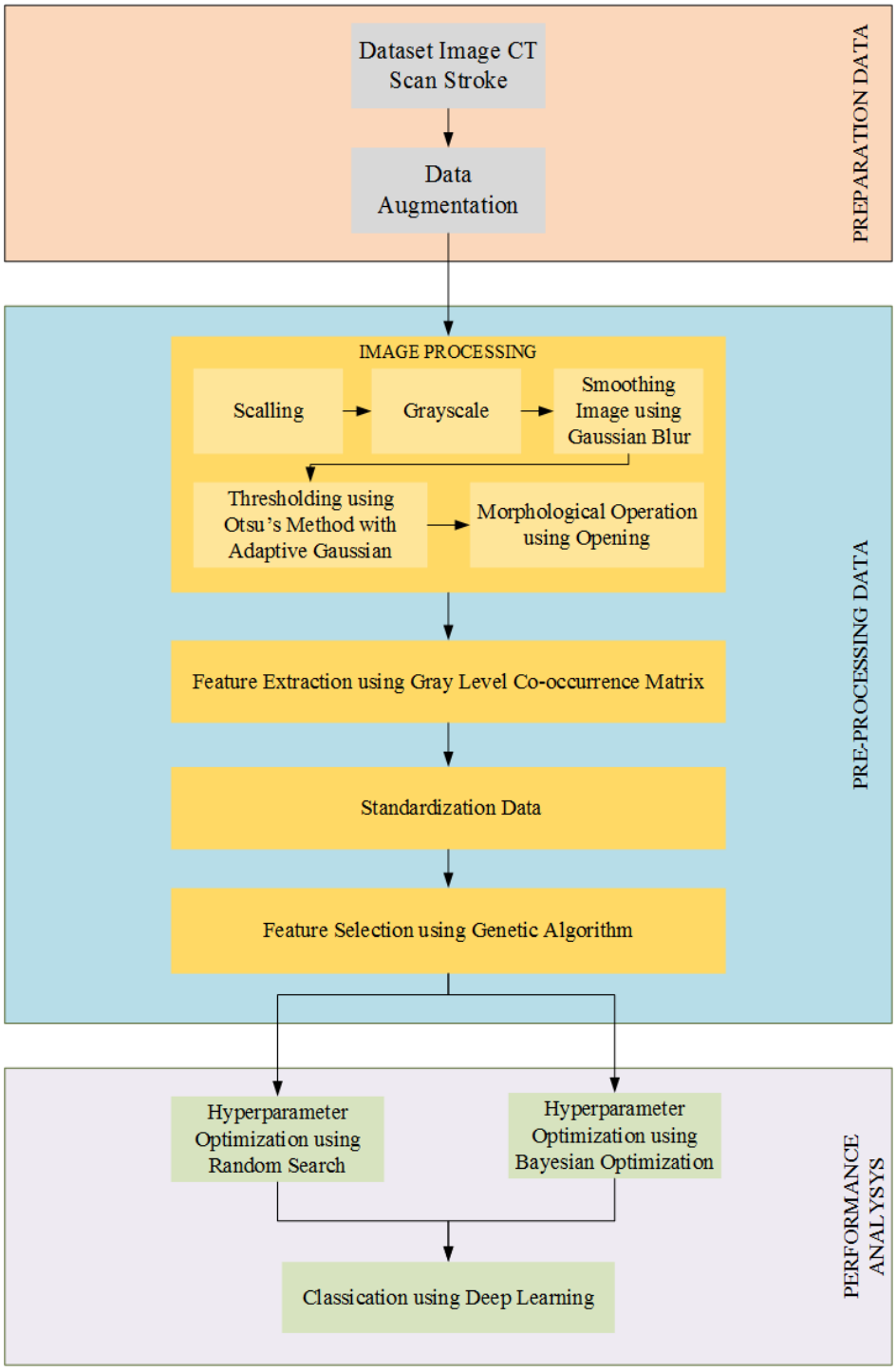

Fig. 1. System Design

At the phase of data preparation, the data obtained from the open dataset was downloaded from the link www.radiopaedia.org in the form of a patient's CT brain scan image. Since the number is not too much, datasets need to be processed in data augmentation to produce new data that comes from the previous data without eliminating the quality of the original image.

Then image processing is carried out with several steps, namely scaling, grayscale, smoothing, thresholding and morphological operation. After that, the texture feature was extracted using the Gray Level Co-occurrence Matrix (GLCM) which produced 6 features. Then the dataset is scaled back, and feature selection is performed to find relevant features to increase accuracy and shorten computing time.

The following will discuss each part of the element in the system from the proposed method to improve stroke diagnosis using Deep Learning with Hyperparameter Optimization.

2.1. Data Preparation

The data preparation process involves collecting data and data augmentation using the same data as [2]. The data has three classes: Normal, Hemorrhagic Stroke, and Ischemic Stroke; each class has 10 data (Fig. 2). The data augmentation process is intended as a strategy of the limited amount of data obtained. Data augmentation manipulates data without losing the data essence. The image can be rotated, flipped, and cropped.

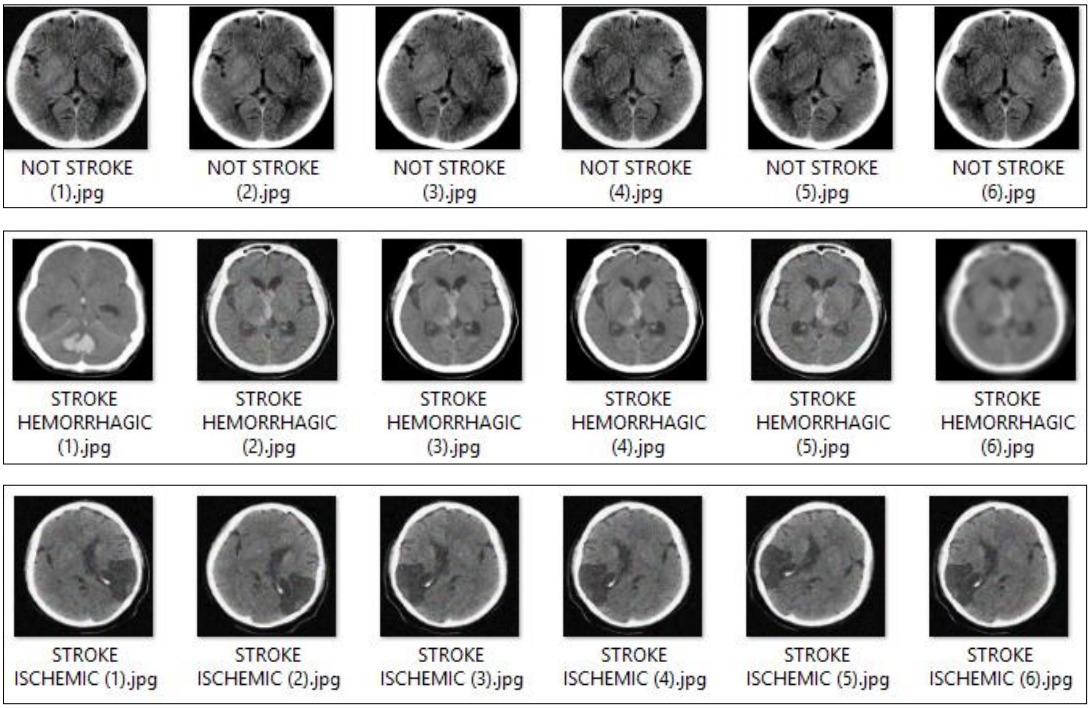

Fig. 2. Image from the data augmentation process. (a) not a stroke (b) hemorrhagic stroke (c) ischemic stroke

At this phase, three sample data are taken from each class (normal, ischemic stroke, hemorrhagic stroke), then the data augmentation output is collected into a new dataset. Each class contains ten data, meaning that there are a total of 30 patients used in the data. The data is then augmented, creating 1,000 rows of data in each class, meaning there are a total of 3,000 rows of data used in the dataset. This augmented data will be processed in the next phase.

There are 10 (ten) rows for each classification of patient data, which are NOT STROKE (normal class), ISCHEMIC (ischemic stroke) and HEMORRHAGIC (hemorrhagic stroke). So that in total there are 30 patient's data. With the augmentation process, these data become 1,000 rows distributed in three classes, so that there is a total of 3,000 rows used to classify whether the patient is normal or has an ischemic stroke or hemorrhagic stroke.

To evaluate performance algorithms, the values in the confusion matrix are needed: True Positive (TP), True Negative (TN), False Positive (FP), False Negative (FN). For example, in the NOT STROKE class:

1) True Positive (TP) is all NOT STROKE class data classified as NOT STROKE.
2) True Negative (TN) is all data other than the NOT STROKE class not classified as NOT STROKE.
3) False Positive (FP) is all data other than the NOT STROKE class that is classified as NOT STROKE.

4) False Negative (FN) is all NOT STROKE class data that is not classified as NOT STROKE

Similarly, the other two classes are HEMORRHAGIC STROKE and ISCHEMIC STROKE. The metrics score used to evaluate the algorithm are accuracy, precision, recall, and f1. In addition, there is a performance analysis using ROC Curve based on the results of plotting True Positive Rate (TPR) with False Positive Rate (FPR) in various threshold settings.

2.2. Data Preprocessing

After data preparation, in the next phase preprocessing data is carried out. It consists of image processing, feature extraction, standardization data and feature selection.

2.2.1. Image Processing

At this phase, the quality of image data is improved. This process consists of scaling (adjusting the size of pixels used), gray scaling (uniformity of the degree of gray image), smoothing image (eliminating noise and giving blur), thresholding (changing to binary images) and morphological operation (processing images based on shapes). In the scaling phase, images that have an original size of 300x300 are changed to 50x50. The grayscale phase is done to produce an uneven level of grayness in the image. The next stage is the smoothing image phase. This phase eliminates the image noise using Gaussian Blur, a very good for CT Scan images [17].

The next stage is thresholding. It is combining the Otsu thresholding function and adaptive thresholding to produce a better image. The last step is the morphological operation. There are two types of Morphological Operation, namely, erosion and dilation. Erosion is an operation to remove the boundaries of foreground objects. On the other hand, the dilation increases thesize of the original image. Here, a joint operation of erosion, followed by dilation, is performed.

2.2.2. Feature Extraction

At the feature extraction phase, the extracted feature is a texture feature with the Gray Level Co-occurrence Matrix (GLCM) method. This method is a compelling method in representing the characteristics of image texture.

GLCM produces six features: contrast, dissimilarity, homogeneity, correlation, angular second moment (ASM), and energy, which explained as follows.

1) Contrast is the result of calculations related to the amount of diversity in gray intensity in the image. Contrast amounts to 0 if the neighbor pixels have the same value. Contrast can be formulated as follows:

$$Contrast = \sum_{i,j=0}^{N-1} P_{i,j}(i-j)^2 \quad (1)$$

where P is matrix co-occurrence, i and j are an index on the matrix, and N for gray level co-occurrence matrix.

2) Dissimilarity is the result of measuring the difference in each pixel, dissimilarity will be high if the texture is random and will be low if the value is uniform. Dissimilarity can be formulated as follows:

$$Dissmilarity = \sum_{i,j=0}^{N-1} P_{i,j}|i-j| \quad (2)$$

where P is co-occurrence matrix, i and j are an index on the matrix, and N gray level co-occurrence matrix

3) Homogeneity is the result of homogeneity measurement. This value is very sensitive to values around the main diagonal. High value exists when all pixels have the same/uniform value. This feature is the opposite of contrast, which is great if it has the same pixel value when the energy is fixed.

$$Homogeneity = \sum_{i,j=0}^{N-1} \frac{P_{i,j}}{1+(i+j)^2} \quad (3)$$

where P is matrix co-occurrence, i and j are an index on the matrix, and N for gray level co-occurrence matrix.

4) Correlation is the result of measuring linearity (the joint probability) of a number of pixel pairs.

$$Correlation = \sum_{i,j=0}^{N-1} P_{i,j} \left[\frac{(i-\mu_i)(j-\mu_j)}{\sqrt{(\sigma_i^2)(\sigma_j^2)}} \right] \quad (4)$$

where P is matrix co-occurrence, i and j are an index on the matrix, N for gray level co-occurrence matrix, μ is the mean value of a pixel, and σ is the variance value of a pixel.

5) Angular Second Moment (ASM)is the result of the measurement of uniformity or often called the angular second moment. The energy has a high value when the pixel values are similar to each other; otherwise the value will be small, indicating the value of the normalized GLCM to be heterogeneous. The maximum value of energy is 1, which means the distribution of pixels is in a constant condition or in the periodic form (not random). ASM can be formulated as follows:

$$ASM = \sum_{i,j=0}^{N-1} P_{i,j}^2 \quad (5)$$

where P is matrix co-occurrence, i and j are an index on the matrix, and N for gray level co-occurrence matrix.

6) Energy is the result of measuring texture uniformity (repetition of pixel pairs). Energy has a value of one for images with constant gray values. Energy can be formulated as follows:

$$Energy = \sqrt{ASM} \quad (6)$$

2.2.3. Standardization data

In this study, we use a standardization process where the attributes of variables are converted into a standard Gaussian distribution with a mean of 0 and a standard deviation of 1.

$$new\ data = \frac{x-\mu}{\sigma} \quad (7)$$

Where μ is the mean value and σ is the standard deviation value of the feature column.

$$\mu = \frac{1}{N}\sum_{i=1}^{N}(x_i) \quad (8)$$

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i-\mu)^2} \quad (9)$$

2.2.4. Feature selection

Feature selection is the process of selecting a relevant and important subset of features that correlates with the class output so that the model produces better accuracy. In this study genetic algorithms based on the wrapper method are used; namely several combinations of feature subset are evaluated and compared with each other. Fig. 3 shows a general phase of genetic algorithms, starting from generating population, evaluating the fitness value of each individual, selecting individuals, and re-production (cross-over and mutation). This results in a new population are combined with the previous population and are chosen as the best one.

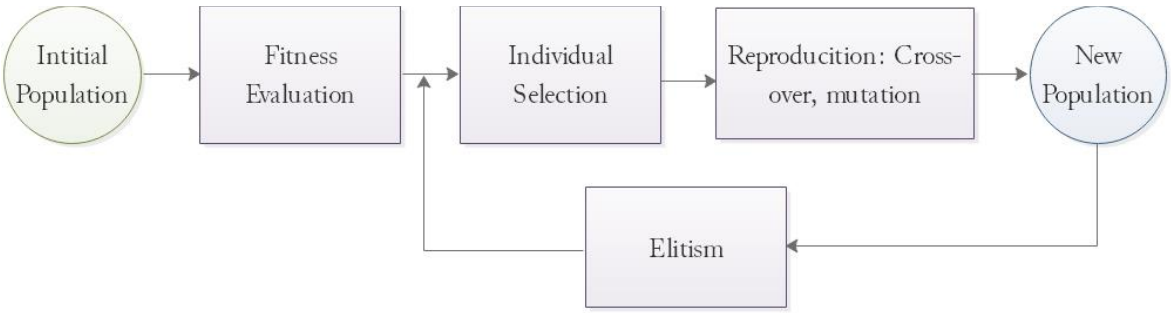

Fig. 3. Genetic Algorithms process flow

Each step of the genetic algorithm (Fig. 3) can be explained as follows:

1) Initial Population. In the initial population, a number of individuals are raised randomly. The chromosome representation of each individual is a representation of a feature, where a value of 1 means that the feature is used, while 0 features are not used or deleted.
2) Fitness Evaluation. After the population generation is evaluated, an Artificial Neural Network is used to calculate the error. Before calculating the fitness value, ranking is based on the calculation of the error multiplied by the constant (for example, constant = 1.5). Following is the formula for calculating fitness values.

$$\emptyset(i) = k.R(i), i = 1,2,3..,N \quad (10)$$

where $\emptyset$ is a fitness value of an individual, k is specified constant, and R is a ranking.
The individual which has the lowest error and ranking has the highest fitness value, and the individual which has the highest error and ranking has the lowest fitness value (Table 1).

Table 1. The Example of Fitness Values Calculation

Individual	Error rate	Ranking	Fitness Value
individual 1	0.5	3	4.5
Individual 2	0.1	4	6
Individual 3	0.9	1	1.5
Individual 4	0.6	2	3

3) Individual Selection. In this phase, the selection of the best parent candidates is passed to the next generation. The higher the fitness value, the higher the probability that the individual is chosen. At this phase, a roulette wheel engine is used for the selection phase.
4) Crossover. The process is a process between individuals chosen to produce new individuals. This process involves exchanging genes from two random individuals.
5) Mutation. This is the process of mutation of a gene or a gene exchanged with its opponent, for example, 0 to 1. The mutation process is carried out with the specified mutation rate.
6) Elitism. This process is a merger of parent and child then ranking based on the evaluation of fitness values and then looking for the best ranking.

2.3. Deep Learning Architecture

Deep Learning architecture used in this study is a fully connected multi-layer perceptron. The following is the process of Deep Learning with the Multilayer Perceptron (MLP) architecture:

1) Initialize the weight randomly, the topology and other hyperparameters.

2) Feedforward process which passes neurons in the hidden layer and calculating the output and target,
3) If both are not the same, then update the weighting process or the backpropagation process. This process is done until it finds a weight with an error approaching 0 (zero) or until the iteration is complete.

In training, there is a dropout process that is removed or ignores neurons or nodes in the hidden layer; this process can also prevent overfitting. The network uses the efficient ADAM gradient descent optimization algorithm.

2.4. Deep Learning using Hyperparameter Optimization

Before the hyperparameter optimization process is carried out, the hyperparameter must be defined by its values. There are many hyperparameters that are owned by Deep Learning. In this study, seven hyperparameters will be optimized, namely hidden layer, hidden node, epoch, learning rate, activation function, batch size and dropout rate. This study uses the Random Search method and Bayesian Optimization to optimize hyperparameter.

The following explains the classification performance used in this study by finding the value of performance measurement:

$$\text{Accuracy} = (\text{TP} + \text{TN})/(\text{TP} + \text{TN} + \text{FP} + \text{FN}) \quad (11)$$

$$Precision = TP/(TP + FP) \quad (12)$$

$$Recall = TP/(TP + FN) \quad (13)$$

$$F1 = 2 * ((Precision * Recall))/(Precision + Recall)) \quad (14)$$

$$TPR = FP/((FP + TN)) \quad (15)$$

$$FPR = TP/((TP + FN)) \quad (16)$$

3. Results and Discussion

Experiments to be discussed include: testing the use of Feature Selection, data classification without hyperparameter optimization and data classification using hyperparameter optimization. Afterward, a summary of the performance analysis between Random Search and Bayesian Optimization is discussed.

3.1. The use of Feature Selection

The result of the feature selection process is having a relevant subset of features. This process uses genetic algorithms. From the results of the feature selection process with 10 generations and 50 number of populations, the feature subset with the number of features 2 was chosen to be the best individual with an accuracy value of 0.99 (Table 2), but the conditions prior to feature selection which were as many as 6 features had the same accuracy of 0.99. What distinguishes the two is the computation time. Data conditions with 2 features have faster training time compared to 6 features. Besides increasing accuracy and finding relevant features, feature selection also reduces training time itself.

For computation time, the selection feature process takes a very long time. The number of generations greatly influences computation time. However, with large computing, good results are obtained and will reduce computation at the data classification phase.

Table 2. Benchmarking Classification With and Without Feature Selection

	Features	Number of features	Accuracy score	Training time (HH:MM:SS)
Without Feature Selection	Contrast, dissimilarity, asm, energy, correlation	6	99%	00:02:44.44
With Feature Selection	Dissimilarity, energy	2	99%	00:02:39.45

2.5. Data classification without hyperparameter optimization

At this phase, the experiment is carried out without optimizing the hyperparameter. For the training data validation and testing data models using the K-Fold or cross-validation model, the K-Fold will form as many as *k* parts initialized. In this experiment, k is initialized as 10. For Classification without Hyperparameter Optimization, we use the common parameters, as follows: ReLU activation function, 256 batch size, 0.6 dropout rate, 0.005 learning rate, 50 number of the epoch, 5 number of hidden layers, and 150 number of hidden nodes. The classification performance without hyperparameter optimization using 10-Cross Validation is shown in Table 3.

Table 3. Classification Performance Without Hyperparameter Optimization

Fold	Performance Measurement			
	Accuracy	Precision	Recall	F1
1	0.62	0.65	0.65	0.65
2	0.70	0.74	0.67	0.60
3	0.66	0.66	0.66	0.66
4	0.68	0.66	0.66	0.59
5	0.70	0.50	0.67	0.56
6	0.71	0.69	0.68	0.68
7	0.67	0.69	0.68	0.66
8	0.67	0.68	0.68	0.68
9	0.67	0.77	0.68	0.58
10	0.67	0.70	0.68	0.62

All training process in each iteration will calculate plot loss and accuracy of classification. The results of the training process will produce loss per epoch and accuracy epoch as shown in Fig 4. The loss values are still within 0.3 to 0.4 (Fig. 4(a) while the accuracy value is mostly at 0.6 to 0.7 (Fig. 4(b)). Hence, there is no indication of overfitting (a condition where the training accuracy is higher than validation). However, it can be expected that when testing the values of accuracy, precision, recall, and f1 it is not good. The scoring results show poor trends, due to the training loss value per epoch and accuracy per epoch being less than satisfactory because of the data is not good, or the settings for each hyperparameter are not appropriate.

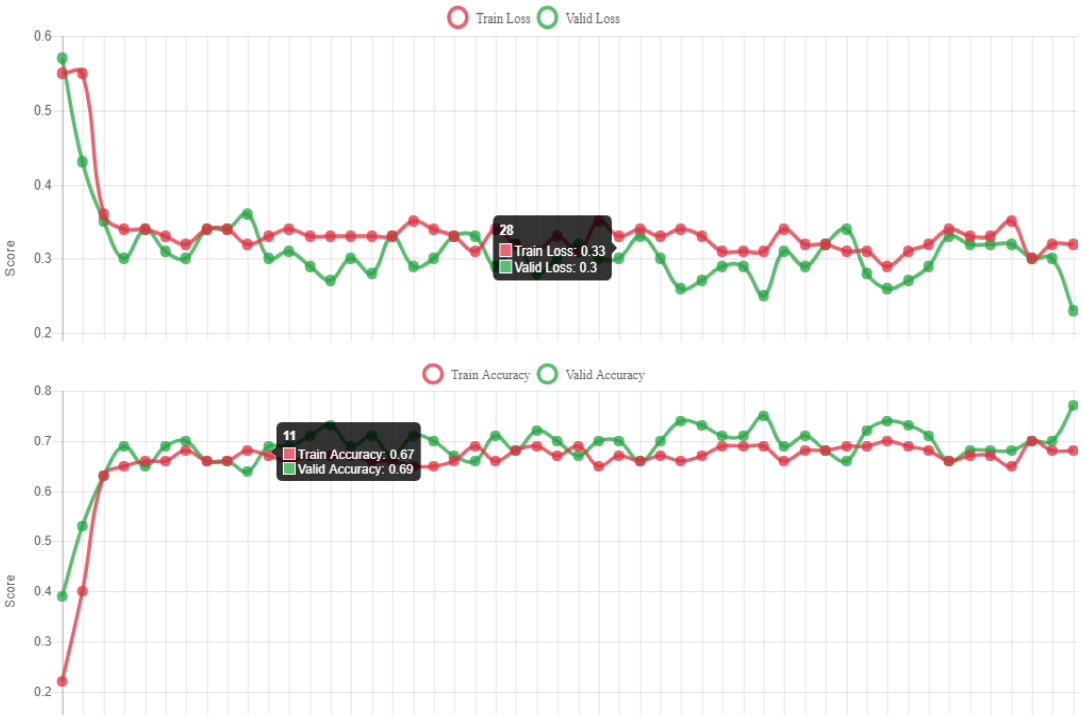

Fig. 4. Plot model loss (a) and accuracy (b) classification without hyperparameter optimization in fold-1;

Fig 5 shows the results of ROC Curve plots, based on the graph resulting in classification results in Table 3.

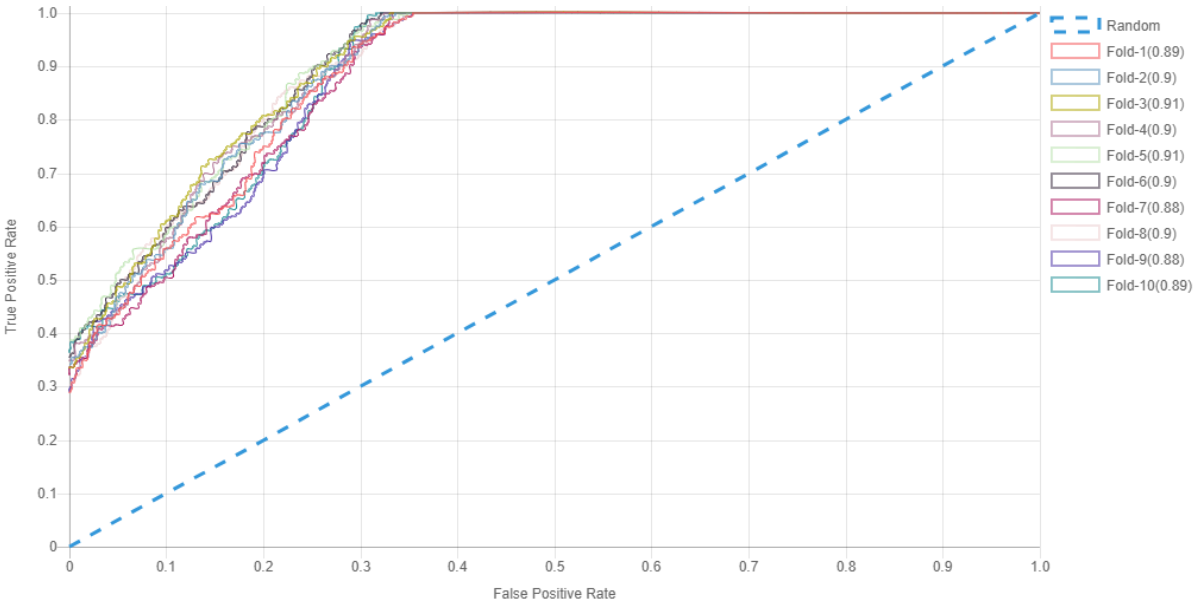

Fig. 5. ROC Curve from classification without hyperparameter optimization

In general, the two graphs above are used to evaluate classification results from datasets that are imbalance (the number of classes is more dominant than the other) but can be used to evaluate the results of a balanced dataset. Table 3 shows that the results of each fold are not good. Most precision values are very high though the recall value is very low, causing the results of the class to be re-called slightly to produce poor results. Whereas in Fig. 5, the value of the ROC Curve is not satisfy low, even though the graph area is still above the random line (>0.500)

2.6. Data classification with hyperparameter optimization

The previous experiment results using classification without hyperparameter optimization showed unsatisfactory results (Tabel 4). Hence improvements must be made, especially since results with high accuracy are necessary in health-related case studies. As explained earlier, Deep Learning can be powerful and produce good results when using the right hyperparameter. The following is the experiment of the hyperparameter optimization process. The first step is defining the range of each hyperparameter value.

Table 4. Hyperparameter Definition on Deep Learning

Hyperparameter	List output
Hidden Layer	[5,6,7,8,9]
Hidden Node	[100,200]
Number of Epoch	[50,60,70,90]
Activation Function	[ReLU, Tanh, Sigmoid, Linear, Softmax]
Learning Rate	[0.001, 0.002, 0.003, 0.004, 0.005, 0.006, 0.007, 0.008, 0.009]
Batch Size	[64, 128]
Dropout Rate	[0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9]

There are 8100 combinations if we look for all possible combinations from the list of parameters above. Therefore, in this study, the grid search optimization method was not used because it would evaluate 8,100 hyperparameter combinations. At this phase, 10 samples will be taken to be evaluated. The experiment was conducted using hyperparameter optimization with a random search and bayesian search.

2.7. Hyperparameter Optimization using Random Search

The result of hyperparameter optimization using Random Search produces an accuracy value of 1.0 or 100%. From Table 5, the best results are obtained by using the Tanh activation function, 64 batch size, 0.5 dropout rate, 0.004 learning rate, 90 number of the epoch, 7 number of hidden layers, and 200 number of hidden nodes; which give perfect accuracy of 1.0 or 100%.

Table 5. Hyperparameter Optimization with Random Search

Fold	Performance Measurement	Hyperparameter						
	Accuracy rate	*Activation Function*	*Batch size*	*Dropout rate*	*Learning rate*	*Number of Epoch*	*Number of hidden layers*	*Number of hidden nodes*
1	1.00	Tanh	64	0.5	0.004	90	7	200
2	0.99	Tanh	64	0.2	0.002	50	6	100
3	0.99	Tanh	128	0.5	0.001	50	6	100
4	0.99	Linear	64	0.5	0.002	60	6	100
5	0.66	Tanh	64	0.7	0.009	50	7	200
6	0.62	Relu	128	0.9	0.004	50	6	200
7	0.52	Relu	64	0.3	0.008	70	9	200
8	0.37	Relu	64	0.7	0.005	80	5	100
9	0.12	Sigmoid	64	0.3	0.001	80	10	100
10	0.00	Sigmoid	128	0.1	0.006	60	8	200

The classification of data with the random search optimization process has the results of plot loss per epoch and the results of the accuracy plot per epoch as in Fig. 6. Fig. 6(a) shows a good trend where the loss value is still 0.1 and even 0, while the accuracy (Fig. 6(b)) is within 0.96 and 1.0 range, indicating that this process has a good scoring.

Fig. 6. Plot loss per epoch (a) and accuracy per epoch (b) using Random Search hyperparameter optimization in fold-1

In Table 6, the value in scoring is very high at 1.0 or 100%, as well as shown in the final results of optimization with Random Search.

Table 6. Scoring results from hyperparameter optimization with Random Search

Fold	Performance Measurement			
	Accuracy	*Precision*	*Recall*	*F1*
1	1.0	1.0	1.0	1.0
2	1.0	1.0	1.0	1.0
3	0.98	0.99	0.98	0.98
4	1.0	1.0	1.0	1.0
5	1.0	1.0	1.0	1.0
6	1.0	1.0	1.0	1.0
7	1.0	1.0	1.0	1.0
8	1.0	1.0	1.0	1.0
9	1.0	1.0	1.0	1.0
10	1.0	1.0	1.0	1.0

Fig. 7 shows the ROC Curve to evaluate the classification results. These results produce satisfied performance measurement and directly proportional to the results of plot loss per epoch in Fig. 6(a) and accuracy per epoch in Fig. 6(b).

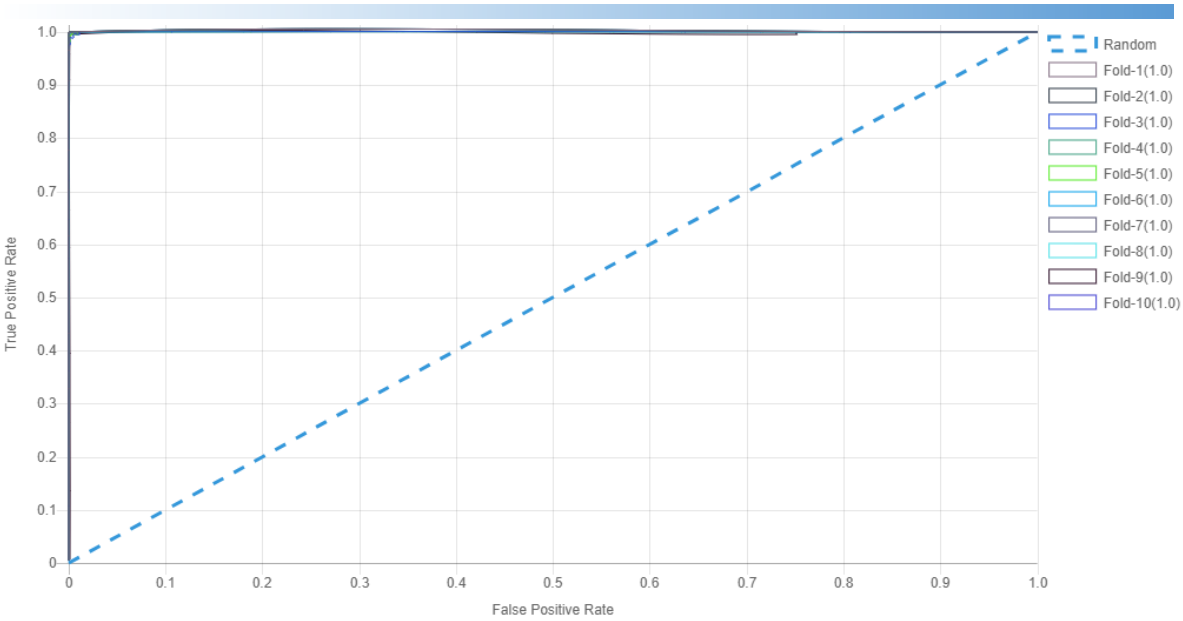

Fig. 7. ROC Curve from hyperparameter optimization with Random Search

2.8. Hyperparameter Optimization using Bayesian Search

Experiments with hyperparameter optimization using Bayesian Optimization produce an accuracy value of 0.99 or 99%. The best results are obtained by using Linear Activation function, 115 Batch size, 0.1 Dropout rate, 0.006 Learning rate, 90 Number of the epoch, 6 Number of hidden layers, and 149 Number of hidden nodes; which give accuracy value of 0.99 or 99% (Table 7).

Table 7. Hyperparameter Optimization with Bayesian

Fold	Performance Measurement	Hyperparameter						
	Accuracy rate	*Activation Function*	*Batch size*	*Dropout rate*	*Learning rate*	*Number of Epoch*	*Number of hidden layers*	*Number of hidden nodes*
1	1.00	Sigmoid	128	0.6	0.007	80	6	106
2	0.99	Tanh	118	0.5	0.003	90	9	173
3	0.99	Tanh	80	0.7	0.005	50	5	107
4	0.99	Linear	115	0.1	0.006	90	6	149
5	0.66	Tanh	105	0.5	0.001	60	6	259
6	0.62	Relu	72	0.4	0.006	70	8	171
7	0.52	Relu	108	0.9	0.004	80	8	123
8	0.37	Relu	121	0.5	0.003	90	8	145
9	0.12	Sigmoid	92	0.3	0.008	70	10	123
10	0.00	Softmax	126	0.3	0.008	50	10	191

The classification of data with the bayesian optimization process has the results of plot loss per epoch in fold-1 and accuracy plot per epoch in fold-1 as in Fig. 8. The results show a consistent trend, although in the middle of the iteration there is fluctuation at the loss value per epoch and accuracy per epoch, however, until the end, the results shown continue to be consistent. Scoring results can also be determined to be good and consistent with the results of the optimization with Bayesian Optimization.

(a)

(b)

Fig. 8. Plot loss per epoch (a) and accuracy per epoch (b) using Bayesian hyperparameter optimization in fold-1

It can be seen that the result has a very high scoring value of 0.99 or 99% (Table 8). The final results of optimization with Bayesian Optimization differ slightly from those produced by Random Search. The results of plot loss per epoch (Fig. 8(a)), accuracy per epoch (Fig. 8(b)), and scoring produce very good things. It means that these results are directly proportional to the results of ROC Curve plots in Fig. 9, which shows very good classification results.

Table 8. Scoring results from hyperparameter optimization with Bayesian

Fold	Performance Measurement			
	Accuracy	*Precision*	*Recall*	*F1*
1	1.0	1.0	1.0	1.0
2	1.0	1.0	1.0	1.0
3	0.98	0.99	0.98	0.98
4	1.0	1.0	1.0	1.0
5	1.0	1.0	1.0	1.0
6	1.0	1.0	1.0	1.0
7	1.0	1.0	1.0	1.0
8	1.0	1.0	1.0	1.0
9	1.0	1.0	1.0	1.0
10	0.9	0.93	0.89	0.89

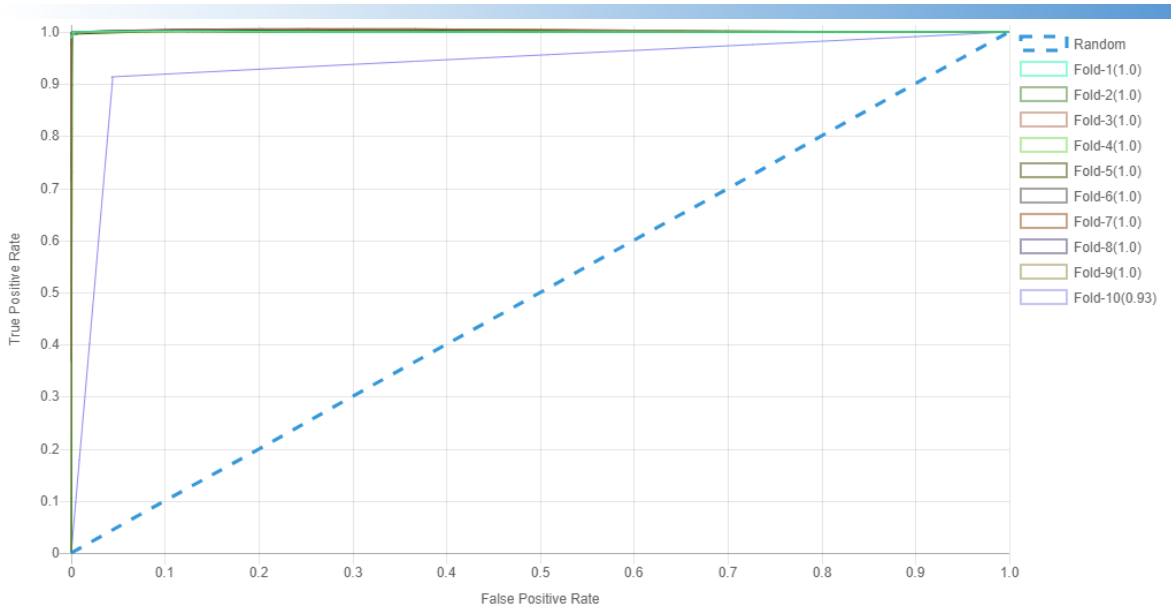

Fig. 9. ROC Curve from hyperparameter optimization with Bayesian in fold-1

A summary of performance analysis between Random Search and Bayesian Optimization is shown in Table 9.

Table 9. Hyperparameter Definition on Deep Learning

Metric Comparison	Random Search	Bayesian Optimization
Time optimization	04:15:11.72	03:52:37.32
Highest accuracy	1.0	0.99
Time of classification testing	01:06:41.80	00:23:15.02

In the Stroke dataset, Bayesian Optimization is more effective and efficient than Random Search. Although in terms of accuracy, Random Search produces an accuracy value of 1.0 or 100% but only a difference of 0.1 or 1% of that produced by Bayesian Optimization (0.99 or 99%). Even so, both of these methods are better than the common methods, Grid Search, which works to find a combination of all possible parameters (brute force), hence requiring a longer processing time.

4. Conclusion

This study seeks to improve the accuracy of a stroke diagnosis by using Hyperparameter Optimization. From the results of the experiment, it was concluded that Deep Learning requires the correct setting of hyperparameters so that Deep Learning produces good knowledge. The results of the hyperparameter optimization in the Stroke dataset in this study succeeded in increasing the value of accuracy, precision, recall and f1-score up to 100%. Random Search and Bayesian Optimization are better than Grid Search, a common method for hyperparameter optimization. In terms of classification results, Random Search produces higher accuracy than Bayesian Optimization. Yet, in terms of time optimization, Bayesian Optimization is better than Random Search. For the next research, we have to use big data platforms to reduce computational burdens and training times. The parallel system may run quickly without reducing its accuracy. We will consider the use of other Deep Learning architectures such as Recurrent Neural Network (RNN), Convolutional Neural Network (CNN), and Long Short-Term Memory (LSTM). We will consider the implementation of a boosting method to improve accuracy, may decrease due to the feature selection.

References

[1] L. S. Brunner, *Brunner & Suddarth's textbook of medical-surgical nursing*, vol. 1. Lippincott Williams & Wilkins, 2010., available at: Google Scholar.

[2] J. T. Marbun, Seniman, and U. Andayani, "Classification of stroke disease using convolutional neural network," *J. Phys. Conf. Ser.*, vol. 978, p. 012092, Mar. 2018, doi: 10.1088/1742-6596/978/1/012092.

[3] C.-Y. Hung, W.-C. Chen, P.-T. Lai, C.-H. Lin, and C.-C. Lee, "Comparing Deep Neural Network and Other Machine Learning Algorithms for Stroke Prediction in a Large-Scale Population-Based Electronic Medical Claims Database," in *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2017, pp. 3110–3113, doi: 10.1109/EMBC.2017.8037515.

[4] G. A. P. Singh and P. K. Gupta, "Performance analysis of various machine learning-based approaches for detection and classification of lung cancer in humans," *Neural Comput. Appl.*, vol. 31, no. 10, pp. 6863–6877, 2019, doi: 10.1007/s00521-018-3518-x.

[5] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015, doi: 10.1038/nature14539.

[6] A. Esteva *et al.*, "A guide to deep learning in healthcare," 2019, doi: 10.1038/s41591-018-0316-z.

[7] L. Deng and J. C. Platt, "Ensemble deep learning for speech recognition," in *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, 2014, available at: Google Scholar.

[8] S. Gollapudi and S. Gollapudi, "Deep Learning for Computer Vision," 2019, doi: 10.1007/978-1-4842-4261-2_3.

[9] H. Assodiky, I. Syarif, and T. Badriyah, "Deep learning algorithm for arrhythmia detection," in *2017 International Electronics Symposium on Knowledge Creation and Intelligent Computing (IES-KCIC)*, 2017, pp. 26–32, doi: 10.1109/KCIC.2017.8228452.

[10] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *International Conference on Learning Representations (ICLR)*, 2015, vol. 5, available at: Google Scholar.

[11] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, no. Feb, pp. 281–305, 2012, available at: Google Scholar.

[12] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in neural information processing systems*, 2012, pp. 2951–2959, available at : Google Scholar.

[13] H. Cui and J. Bai, "A new hyperparameters optimization method for convolutional neural networks," *Pattern Recognit. Lett.*, 2019, doi: 10.1016/j.patrec.2019.02.009.

[14] C. Di Francescomarino *et al.*, "Genetic algorithms for hyperparameter optimization in predictive business process monitoring," *Inf. Syst.*, vol. 74, pp. 67–83, May 2018, doi: 10.1016/j.is.2018.01.003.

[15] N. Q. K. Le, T.-T. Huynh, E. K. Y. Yapp, and H.-Y. Yeh, "Identification of clathrin proteins by incorporating hyperparameter optimization in deep learning and PSSM profiles," *Comput. Methods Programs Biomed.*, vol. 177, pp. 81–88, Aug. 2019, doi: 10.1016/j.cmpb.2019.05.016.

[16] F. J. Martinez-de-Pison, R. Gonzalez-Sendino, A. Aldama, J. Ferreiro-Cabello, and E. Fraile-Garcia, "Hybrid methodology based on Bayesian optimization and GA-PARSIMONY to search for parsimony models by combining hyperparameter optimization and feature selection," *Neurocomputing*, vol. 354, pp. 20–26, Aug. 2019, doi: 10.1016/j.neucom.2018.05.136.

[17] P. Balaprakash, M. Salim, T. Uram, V. Vishwanath, and S. Wild, "DeepHyper: Asynchronous Hyperparameter Search for Deep Neural Networks," in *2018 IEEE 25th International Conference on High Performance Computing (HiPC)*, 2018, pp. 42–51, doi: 10.1109/HiPC.2018.00014.

[18] P. Neary, "Automatic Hyperparameter Tuning in Deep Convolutional Neural Networks Using Asynchronous Reinforcement Learning," in *2018 IEEE International Conference on Cognitive Computing (ICCC)*, 2018, pp. 73–77, doi: 10.1109/ICCC.2018.00017.

[19] R. J. Borgli, H. Kvale Stensland, M. A. Riegler, and P. Halvorsen, "Automatic Hyperparameter Optimization for Transfer Learning on Medical Image Datasets Using Bayesian Optimization," in *2019 13th International Symposium on Medical Information and Communication Technology (ISMICT)*, 2019, pp. 1–6, doi: 10.1109/ISMICT.2019.8743779.

[20] S. S. Talathi, "Hyper-parameter optimization of deep convolutional networks for object recognition," in *2015 IEEE International Conference on Image Processing (ICIP)*, 2015, pp. 3982–3986, doi: 10.1109/ICIP.2015.7351553.

[21] N. N. Y. Vo, X. He, S. Liu, and G. Xu, "Deep learning for decision making and the optimization of socially responsible investments and portfolio," *Decis. Support Syst.*, vol. 124, p. 113097, Sep. 2019, doi: 10.1016/j.dss.2019.113097.

[22] X. Dong, J. Shen, W. Wang, Y. Liu, L. Shao, and F. Porikli, "Hyperparameter Optimization for Tracking with Continuous Deep Q-Learning," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 518–527, doi: 10.1109/CVPR.2018.00061.

[23] C. Yao, D. Cai, J. Bu, and G. Chen, "Pre-training the deep generative models with adaptive hyperparameter optimization," *Neurocomputing*, vol. 247, pp. 144–155, Jul. 2017, doi: 10.1016/j.neucom.2017.03.058.

[24] Y. Yoo, "Hyperparameter optimization of deep neural network using univariate dynamic encoding algorithm for searches," *Knowledge-Based Syst.*, vol. 178, pp. 74–83, Aug. 2019, doi: 10.1016/j.knosys.2019.04.019.

[25] A. Candelieri *et al.*, "Tuning hyperparameters of a SVM-based water demand forecasting system through parallel global optimization," *Comput. Oper. Res.*, vol. 106, pp. 202–209, Jun. 2019, doi: 10.1016/j.cor.2018.01.013.

[26] H. Laanaya, F. Abdallah, H. Snoussi, and C. Richard, "Learning general Gaussian kernel hyperparameters of SVMs using optimization on symmetric positive-definite matrices manifold," *Pattern Recognit. Lett.*, vol. 32, no. 13, pp. 1511–1515, Oct. 2011, doi: 10.1016/j.patrec.2011.05.009.

[27] R. Laref, E. Losson, A. Sava, and M. Siadat, "On the optimization of the support vector machine regression hyperparameters setting for gas sensors array applications," *Chemom. Intell. Lab. Syst.*, vol. 184, pp. 22–27, Jan. 2019, doi: 10.1016/j.chemolab.2018.11.011.

[28] V. Strijov and G. W. Weber, "Nonlinear regression model generation using hyperparameter optimization," *Comput. Math. with Appl.*, vol. 60, no. 4, pp. 981–988, Aug. 2010, doi: 10.1016/j.camwa.2010.03.021.

[29] E. S. Tellez, D. Moctezuma, S. Miranda-Jiménez, and M. Graff, "An automated text categorization framework based on hyperparameter optimization," *Knowledge-Based Syst.*, vol. 149, pp. 110–123, Jun. 2018, doi: 10.1016/j.knosys.2018.03.003.

[30] P. Tsirikoglou, S. Abraham, F. Contino, C. Lacor, and G. Ghorbaniasl, "A hyperparameters selection technique for support vector regression models," *Appl. Soft Comput.*, vol. 61, pp. 139–148, Dec. 2017, doi: 10.1016/j.asoc.2017.07.017.

[31] R. G. Mantovani, A. L. D. Rossi, E. Alcobaça, J. Vanschoren, and A. C. P. L. F. de Carvalho, "A meta-learning recommender system for hyperparameter tuning: Predicting when tuning improves SVM classifiers," *Inf. Sci. (Ny).*, vol. 501, pp. 193–221, Oct. 2019, doi: 10.1016/j.ins.2019.06.005.

[32] W.-Y. Lee, S.-M. Park, and K.-B. Sim, "Optimal hyperparameter tuning of convolutional neural networks based on the parameter-setting-free harmony search algorithm," *Optik (Stuttg).*, vol. 172, pp. 359–367, Nov. 2018, doi: 10.1016/j.ijleo.2018.07.044.