I.J. Intelligent Systems and Applications, **2021, 5, 1-13**
Published Online October 2021 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijisa.2021.05.01

Comparing Performance of Supervised Learning Classifiers by Tuning the Hyperparameter on Face Recognition

M. Ilham Rizqyawan, Ulfah Nadiya
Technical Implementation Unit for Instrumentation Development, Indonesian Institute of Sciences, Bandung, 40135, Indonesia
E-mail: muha176@lipi.go.id, ulfa001@lipi.go.id

Aris Munandar, Jony Winaryo Wibowo, Oka Mahendra, Irfan Asfy Fakhry Anto, Rian Putra Pratama, Muhammad Arifin, Hanif Fakhrurroja
Technical Implementation Unit for Instrumentation Development, Indonesian Institute of Sciences, Bandung, 40135, Indonesia
E-mail: aris001@lipi.go.id, jony001@lipi.go.id, okam001@lipi.go.id, irfa009@lipi.go.id, rian018@lipi.go.id, muha2018@lipi.go.id, hani002@lipi.go.id

Received: 12 July 2021; Accepted: 29 August 2021; Published: 08 October 2021

Abstract: In this era, face recognition technology is an important component that is widely used in various aspects of life, mostly for biometrics issues for personal identification. There are three main steps of a face recognition system: face detection, face embedding, and classification. Classification plays a vital role in making the system recognizes a face accurately. With the growing need for face recognition applications, the need for machine learning methods are required for accurate image classification is also increasing. One thing that can be done to increase the performance of the classifier is by tuning the hyperparameter. For this study, the evaluation performance of classification is conducted to obtain the best classifier among four different classifier algorithms (decision tree, SVM, random forest, and AdaBoost) for a specific dataset by tuning the hyperparameter. The best classifier is obtained by evaluating the performance of each classifier in terms of training time, accuracy, precision, recall, and F1-score. This study was using a dataset of 2267 facial data (128D vector space) derived from the face embedding process. The result showed that SVM is the best classifier with a training time of 0.5 s and the score for accuracy, precision, recall, and F1-score are about 98%.

Index Terms: Classifier, Hyperparameter, Face Recognition.

1. Introduction

A face is a key to the human perception system [1]. Different facial cues can identify different people. The most important facial cues include nose, eyes, jaw, hair, and mouth [2]. The advances in technology, the development of existing hardware and software, allows faces to be processed into objects or often known as face recognition. Face recognition has been used for many applications, which are for security, image database investigation, face identification, access control, video surveillance, generally identify verification, smart card application, the criminal justice system, shopping sense like in Amazon Go, etc [3,4,5,6,7].

The general process of face recognition consists of face detection, face embedding, and classification. The first process of face recognition is started by detecting whether there is a face or not in a picture/frame. This is done by identifying the sizes of a known number of faces and their locations [8]. If the face has been found in a frame, then the face embedding will be performed. Face embedding acts as a feature extractor from the detected faces. The goals of this process are to reduce the machine training time and the space complexity [9]. Feature extractor transforms an input into a set of features and selects a feature containing the most relevant information [10]. The selected features are then classified by the classifier. Generally, this process consists of a machine learning algorithm that has been trained with a number of certain datasets so it can obtain the desired criteria.

The very important part of the face recognition process is classification because it is a learning process that allows the system to differentiate the classes by reaching the minimum possible error based on the training dataset to obtain accurate facial recognition. Face classifications are frequently performed using a supervised learning algorithm due to

high-dimensional data. Image classification can be done by using some kind of supervised learning algorithms such as support vector machine (SVM), random forest, AdaBoost, decision tree, etc. Each algorithm is modeled in such a way so it has certain different hyperparameters. This hyperparameter affects the performance of the classifier in building a robust and accurate model [11,12], hence, the optimal hyperparameters are needed to obtain maximum performance.

Previous studies have been comparing several classifiers on face recognition to find the best classifier. Emir et al., Abdel et al., and Huda et al. [13,14,15] have been comparing SVM classifier and random forest classifier on face recognition and using Histogram of Oriented Gradients (HOG) as the feature extractor. They got a result that random forest reached higher accuracy than SVM with the accuracy value is 97.2% [13], 95.1% [14], and 95.9% [15]. Nurbaity et al. [16] are also comparing naïve Bayes, SVM, and multi-linear perceptron (MLP) classification. The result showed that naïve Bayes has higher accuracy than the others with a value of 93.2%. Wang et al. [17] compared Adaboost, SVM, linear logic classification, nearest neighbor algorithm (KNN), and linear discrimination (LDA) and the result is linear logic got the highest accuracy among the others with the value is 99%. Last, Nabatchian et al. [18] tested the performance of KNN, SVM, and hidden Markov model (HMM) classifiers, and the best accuracy was reached by HMM with the value of 91%.

This study also evaluates several different classifier algorithms for a specific dataset by evaluating the performance of each classifier to obtain the best classifier by tuning its hyperparameters which most studies have not done before because tuning the hyperparameter would be a tedious and continuous task to randomly trying the hyperparameter values, whereas hyperparameter can improve the performance of the classifier. Hyperparameter tuning is also tricky in the sense that there is no direct way to calculate how a change in the hyperparameter value will reduce the loss of the model, so tuning a model is trial-and-error based engineering.

The used dataset contains 2267 facial data (128D vector space) derived from the face embedding process. In this paper, we conducted data pre-processing, model validation, hyperparameter tuning, and model evaluation on four classifiers: decision tree, SVM, random forest, and AdaBoost. This study also enriches a reference to the comparison of the classifier algorithm that developers can use in choosing the best classifier algorithm according to their needs to obtain an optimum face recognition system. Besides the accuracy level, this research also evaluates the performances in terms of training time, precision, recall, and F1-score. And the result showed that the best classifier is SVM with a training time is 0.5 s and the score for accuracy, precision, recall, and F1-score are about 98%.

This paper consists of five sections. The first is an introduction that discusses an overview of the face recognition process, particularly for classification parts, and discusses previous studies and the background of this research. The second section is a literature review which tells us about literature related to this research. The third discusses the methodology which consists of a dataset and experimental methods. The fourth discusses the experimental result of the proposed methods. The last part is the conclusion of this research and its future work.

2. Literature Review

A supervised learning classifier is being applied for many domains, one of them is face recognition. This algorithm predicts the values of a class from labeled data. This algorithm has some hyperparameters that can be tuned manually by configuring them to obtain optimal predictive performance. To be able to adjust the hyperparameters, we need to understand the means of the hyperparameter and the effect on the model, hence, this section discusses a brief explanation about the classifier model that we used for this experiment and its hyperparameter and the effect on the classifier model.

2.1. Support Vector Machine (SVM)

Support Vector Machine (SVM) is one of the supervised algorithms that is commonly used for image classification due to SVM performs better than another classifier in terms of less structured data [19]. SVM represents a geometrical model that finds the best margin that differentiates data points of different categories using this function (1) [20].

$$min_{w,w_0}\|w\|_2^2 \tag{1}$$

subject to this constraint in (2):

$$y^{(n)}(w^T x^n + w_0) \geq 1 \quad \forall_n \tag{2}$$

where $y^{(n)}$ being the label for data-point n, x being the matrix, w being the weight vector, and w_0 being the bias. The decision boundary of SVM is given in (3).

$$max_{w,w_0} M, \tag{3}$$

where M is the margin in the decision boundary as in (4).

$$M \leq \frac{1}{\|w\|_2} y^{(n)}\left(w^T x^{(n)} + w_0\right) \quad \forall_n \tag{4}$$

In this experiment, we optimized the hyperparameter of SVM before we trained the models. In SVM, hyperparameter helps to find the balance between the bias (w_0) and variance (w) that will affect the decision boundary ($max_{w,w_0} M$) and thus prevent the model from underfitting or overfitting. SVM has several parameters [21]. In this experiment, we use four different parameters. The parameters are in Table 1.

Table 1. The SVM Parameters

Parameters	Data Type	Default Value	Description
C	Float	1.0	The value must be strictly positive
Kernel	'sigmoid', 'precomputed', callable, 'linear', 'rbf', or 'poly'	'rbf'	• If none, uses 'rbf' • If 'callable', then the kernel matrix from data matrices will be precomputed; the matrix is an array of shape
Gamma	'scale', 'auto', or float	'scale'	Kernel coef. for 'sigmoid', 'rbf' and 'poly'. • If gamma = 'scale', then gamma = 1/(n_features* X.var()) • If gamma = 'auto', then gamma = 1/n_features
Polynomial Degree	Integer	3	Degree for kernel value 'poly'. Ignored by other kernels value

2.2. Decision Tree

The decision tree model has two components, which are nodes and branches, and has three important steps, splitting, pruning, and tree decision. [22]. A decision tree has an issue of robust to irregulates input data and it is insensitive to outliers, it is easily overfitting and unstable [23]. This overfitting can be counteracted by adjusting the hyperparameters. The hyperparameters will stop the recursive splitting process (splitting is a process of partitioning a node into two or more sub-nodes), so it is setting constraints on tree size. There are several parameters for the decision tree algorithm [24,25,26]. In this paper, the experiment uses four different parameters, which are as in Table 2.

Table 2. The decision tree parameters

Parameters	Data Type	Default Value	Description
Criterion	"gini", "entropy"	"gini"	It measures the split quality. The value is "gini" for the Gini impurity and "entropy" for the information gain
Max Depth	Integer	None	The tree maximum depth. If the value is the default, the node is expanded until all leaves are less than min_samples_split or until it is pure
Min Samples Split	Integer or float	2	It is a sample minimum number to split the internal node: • If an integer, min_samples_split is the min. number • If float, min_samples_split is a fraction, and ceil(min_samples_split * n_samples) is the samples min. number of each split
Min Samples Leaf	Integer or float	1	It is a sample minimum number at the leaf node. A split point can be allowed if it leaves at least min_samples_leaf training samples in the right and left branches. It may effects the smoothing of the model, particularly for regression. • If integer, min_samples_leaf is the min. number. • If float, min_samples_leaf is a fraction, and ceil(min_samples_leaf * n_samples) is the min. number of samples of each node

2.3. Random Forest

Random forest combines decision trees, feature sub-sampling, and bootstrap aggregation to reduce the trade-off in the decision tree by decreases the model variance, so it can increase the accuracy [27]. Random forest collects M randomized trees as denoted in (5) [28].

$$m_n\left(x, \ominus_j, \mathcal{D}_n\right) \tag{5}$$

the j-th tree gives the predicted value at point x, $\ominus_1 \ldots \ominus_M$ being independent random variables, and $\mathcal{D}_n$ being independent of sample. Random forest prediction is given by the average of the prediction of M randomized trees and it is denoted in (6).

$$m_{M,n}\,(x,\,\ominus_1 \ldots \ominus_M,\,\mathcal{D}_n) = \frac{1}{M}\sum_{m=1}^{M} m_n(x,\,\ominus_m,\,\mathcal{D}_n) \tag{6}$$

In the random forest, several hyperparameters that can be tuned during the training process. The hyperparameters include the number of the randomized tree (M) and the number of independent samples ($\mathcal{D}_n$). Several parameters that used in this paper are in Table 3 [29].

Table 3. The random forest parameters

Parameters	Data Type	Default Value	Description
n Estimators	Integer	100	The trees number in the forest
Max Depth	Integer	None	The tree maximum depth. If the value is the default, the node is extended until all of the leaves are less than min_samples_split or until it is pure
Min Samples Split	Integer or float	2	It is a sample minimum number to split the internal node: • If an integer, min_samples_split is the min. number • If float, min_samples_split is a fraction and ceil(min_samples_split * n_samples) is the samples min. number of each split
Min Samples Leaf	Integer or float	1	It is a sample minimum number at the leaf node. A split point can be allowed if it leaves at least min_samples_leaf training samples in the right and left branches. It may effects the smoothing of the model, particularly for regression. • If integer, min_samples_leaf is the min. number • If float, min_samples_leaf is a fraction and ceil(min_samples_leaf * n_samples) is the min. number of samples of each node

2.4. Adaboost

Adaboost improves different classification performances by combining the weak classifiers to become a strong classifier with a scheme of weighted majority voting [30]. Adaboost is popular for face recognition applications, for example in the Viola–Jones framework [31]. It utilizes the exponential loss function $L\left(y, f(x)\right) = e^{-yf(x)}$, where $f(x)$ being the prediction and y being the actual label. The equation for optimizing the cost function is denoted in (7) [30].

$$J(\{w^{\{t\}}, v^{\{t\}}\}t) = \sum_{n=1}^{N} L(y^{(n)}, f^{\{t-1\}}(x^{(n)}) + w^{\{t\}}\varphi(x^{(n)}, v^{\{t\}})) \tag{7}$$

where $y^{(n)}$ being the instance label $n, L(y^{(n)}, f^{\{t-1\}}(x^{(n)})$ is the loss function at the previous stage, $w^{\{t\}}$ being an updated weight depending on whether the data-points are classified correctly, and $\varphi(x^{(n)}, v^{\{t\}})$ being a weak learner which added into the model at stage t.

Tuning the hyperparameter will affect the accuracy of the models. In the case of Adaboost, tuning the hyperparameters will change the number of weak learners ($\varphi(x^{(n)}, v^{\{t\}})$) and control the loss function ($L(y^{(n)}, f^{\{t-1\}}(x^{(n)})$) used for calculating the weight ($w^{\{t\}}$). This experiment uses three different parameters as in Table 4 [32][33].

Table 4. The Adaboost parameters

Parameters	Data Type	Default Value	Description
n Estimators	integer	50	The estimator maximum number
Learning rate	float	1	The weight is applied to each classifier at each boosting iteration. Each classifier contribution can increase by increasing the learning rate value
Algorithm	'SAMME', 'SAMME.R'	'SAMME.R'	• If 'SAMME.R', use SAMME.R as a real boosting algorithm. SAMME.R has faster performance than SAMME, it has a lower test error with fewer boosting iterations • If 'SAMME', uses SAMME as a discrete boosting algorithm

3. Methods

This research aims to find the best performance of a classifier. The general process of the face recognition task can be described by the following list: Face Detection, Face Embedding, and Classification. The experiment was conducted in a workstation with the specification as follows: CPU Intel Core i7-8700 3.2GHz 6 Core 12 Threads, RAM 32GB DDR4 PC-19200, GPU: GeForce RTX 2070.

The first process of face recognition is started by detecting whether there is a face or not in the picture/frame. The algorithm then returned the list of the face(s) and its location in the form of a bounding box(es). Histogram of Oriented Gradients (HOG) is used along with linear Support Vector Machine (SVM) to detect face and its location. HOG is used for this paper because it is a simple and fast algorithm and performs well in detecting face [34] by dividing the image into many cells where a histogram counts the occurrences of pixels' orientations given by their gradients [35]. It summarizes the distribution of measurements within the image regions [36]. In this experiment, we used the HOG library in python.

The next step is face embedding. Face embedding acts as a feature extractor from the detected face(s). Convolutional Neural Network (CNN) with metric loss is used to embed detected face(s) into a 128D vector space. We used CNN because it is a powerful feature extractor that can extract complex data layer by layer and finally form ideal features suitable for classification [37]. This paper also used a CNN library from python.

The vector space from the face embedding process is the final feature used in classification. The details of the classification process will be described in the next subsection. After the classification process, we will evaluate the performance of each classifier to obtain the best classifier. The overview of general methods is shown in Fig.1.

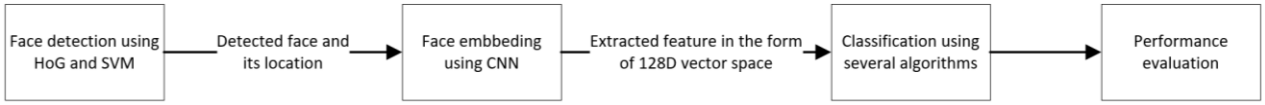

Fig.1. Overview of the General Methods

3.1. Dataset

The used dataset contains 2267 face data. The face data is a 128D vector derived from the face embedding process with the addition of the label column. There are nine classes of face, consisting of person 1 to 8 plus unknown class. The full dataset is then divided into two separate datasets for training and testing samples by random sampling. The training dataset contains 1813 data while the test dataset contains 454 data. The distribution class of the dataset showed in Fig.2. The training data becomes an input for hyperparameter optimization to obtain the best configuration of hyperparameter that will be used in the final model.

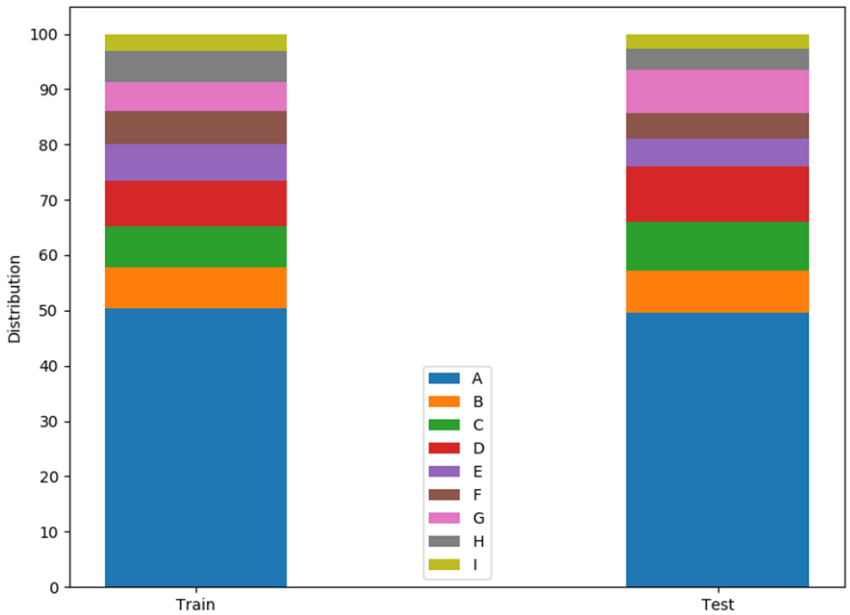

Fig.2. The distribution of dataset class

3.2. Experimental Methods: Classification Process

The experimental methods of the classification process can be seen in Fig.3. In this case, an experiment is conducted by measuring several metrics, mainly accuracy and computing performance to find the best classifier. There are four classification algorithms tested in this experiment: Decision Tree, SVM, Random Forest, and Adaboost. The best parameter is searched for each classifier with the process of hyper-parameter optimization. The parameters and their data type are shown in Table 5 as follows.

Table 5. The parameter and its type

#	Classifier	Parameter 1	Parameter 2	Parameter 3	Parameter 4
1	Decision Tree	Criterion	Max Depth	Min Samples Split	Min Samples Leaf
		Categorical	*Integer*	*Integer*	*Integer*
2	SVM	C	Kernel	Polynomial Degree	Gamma
		Real	*Categorical*	*Integer*	*Real*
3	Random Forest	n Estimators	Max Depth	Min Samples Split	Min Samples Leaf
		Integer	*Integer*	*Integer*	*Integer*
4	Adaboost	n Estimators	Learning Rate	Algorithm	-
		Integer	*Real*	*Categorical*	-

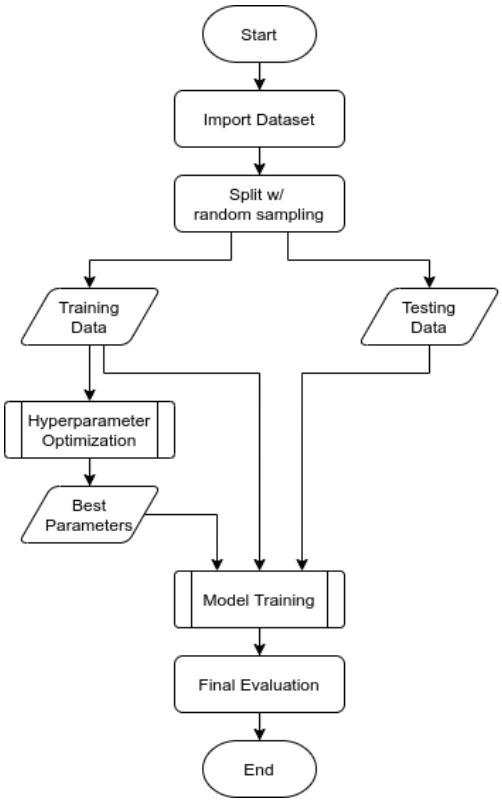

Fig.3. Classification Process

The process of hyperparameter optimization is shown in Fig.4. Each classifier is optimized in 1000 epochs using three optimizer algorithms: Forest, GBRT, and Gaussian Process. Accuracy in each epoch is measured using Cross-Validation with 3 folds, which then averaged to be used as the metric of the optimization. The average accuracy is multiplied with -1 as the optimizer set as a minimizer of the value function. Parameters of the best result in each classifier are then used as the parameters to train the final classifier.

The final training process of the classifier can be seen in Fig.5. Each classifier is trained for the final model test using the best configuration of the parameter. After training, the model is tested using a test dataset and then we calculated the performance of each classifier using several metrics measurements to obtain the best performance of the classifiers.

There are several metrics measured in this final model test. The first one is accuracy. Accuracy simply measures the percentage of correctly identified conditions. Accuracy is denoted as in (8).

$$Accuracy = \frac{(TP+TN)}{(TP+TN+FP+FN)} \quad (8)$$

where TP is true positive, TN being true negative, FP is false positive, and FN being false negative.

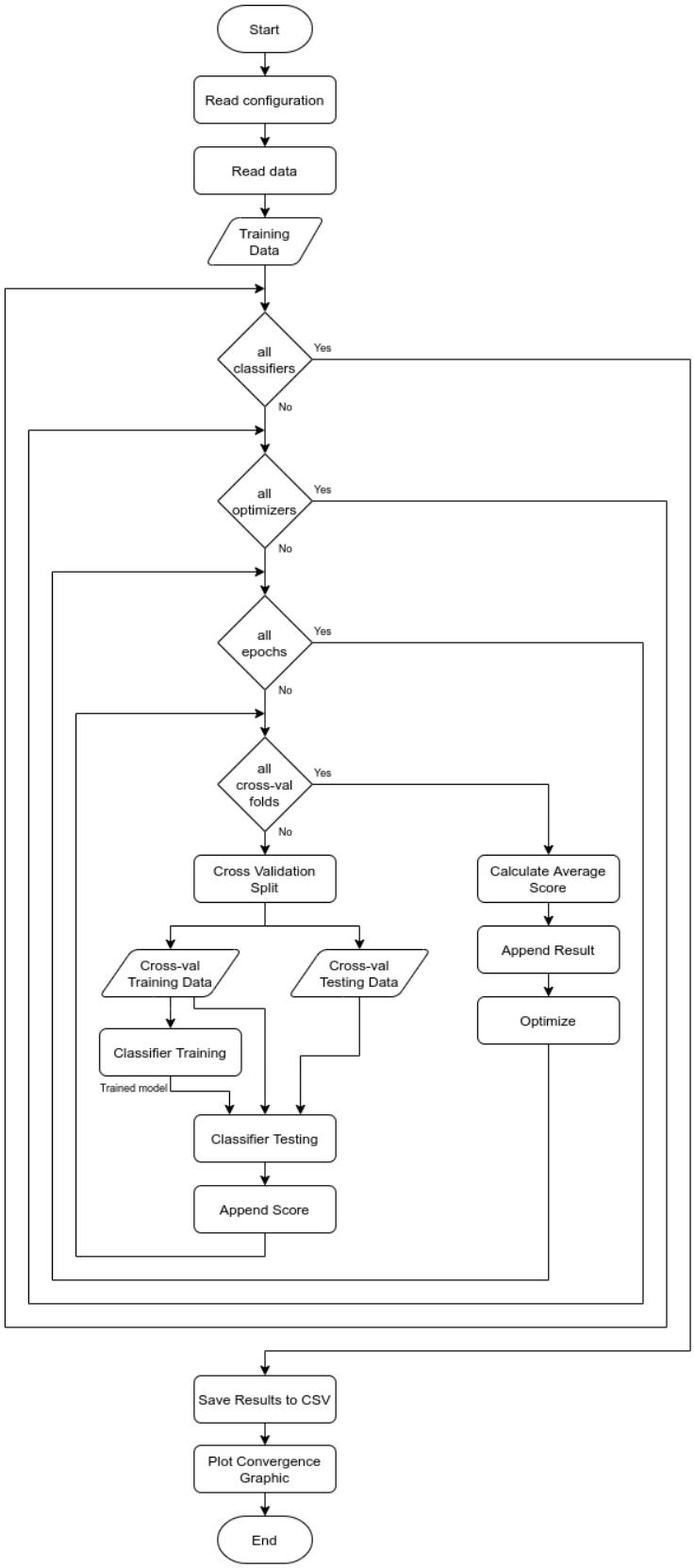

Fig.4. Hyperparameter Optimization Process

The next metrics are Precision and Recall. Precision is a fraction of true predicted instances over all of the predicted instances. While Recall is a fraction of true predicted instances over all of the true instances. Precision and Recall are calculated as in (9) and (10).

$$Precision = \frac{TP}{TP+FP} \tag{9}$$

$$Recall = \frac{TP}{TP+FN} \tag{10}$$

Precision and recall are often in the inverse relation, where trade-offs cannot be avoided to choose one over the other. In some cases, precision is better to measure than recall and vice versa. But in the more general case, a harmonic average of precision and recall can be utilized as a single metric to measure the model. This harmonic average of precision and recall is called F1-score or F-measure and can be calculated as in (11).

$$F1 - score = 2 \cdot \frac{Precision \cdot Recall}{Precision+Recall} \tag{11}$$

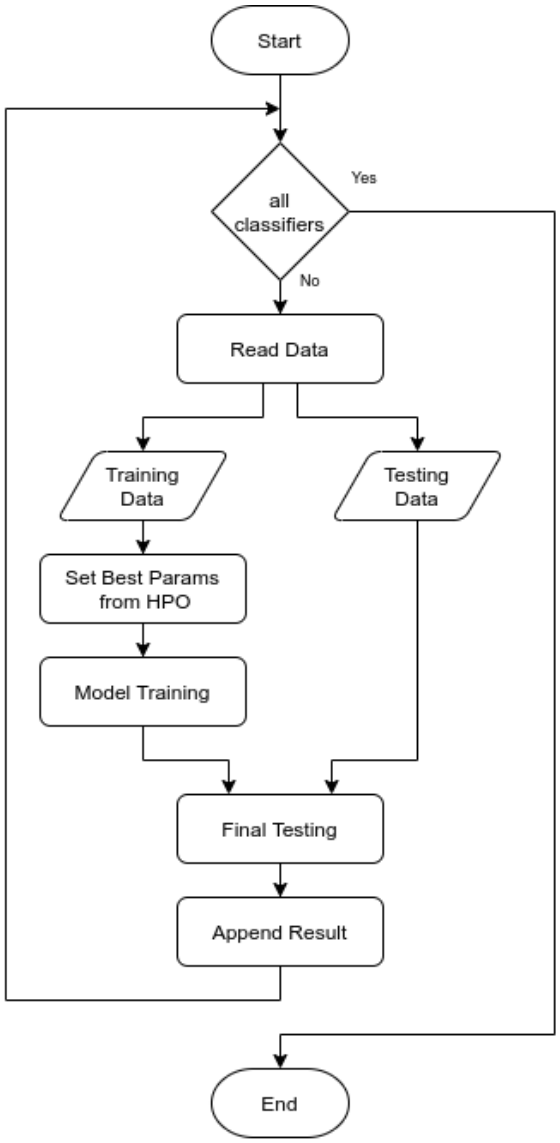

Fig.5. The Process of Final Model Test

4. Results and Discussion

The methods that have been proposed in the previous section are then implemented on the system. First, we optimized the hyperparameters of each classifier. The second one is we noted the configuration result of the hyperparameter from the training process and tested the hyperparameter configuration of each classifier to analyze the

performance. The results are discussed in the following sub-section.

4.1. Hyperparameter Optimization

Classifier algorithms have several parameters to set. The parameters need to be optimized for classifiers to reach the global minimum of error as close as possible. There are 12 configurations of optimization tested in this experiment. Each configuration is optimized in 1000 epoch. The convergence of the optimization showed in Fig.6.

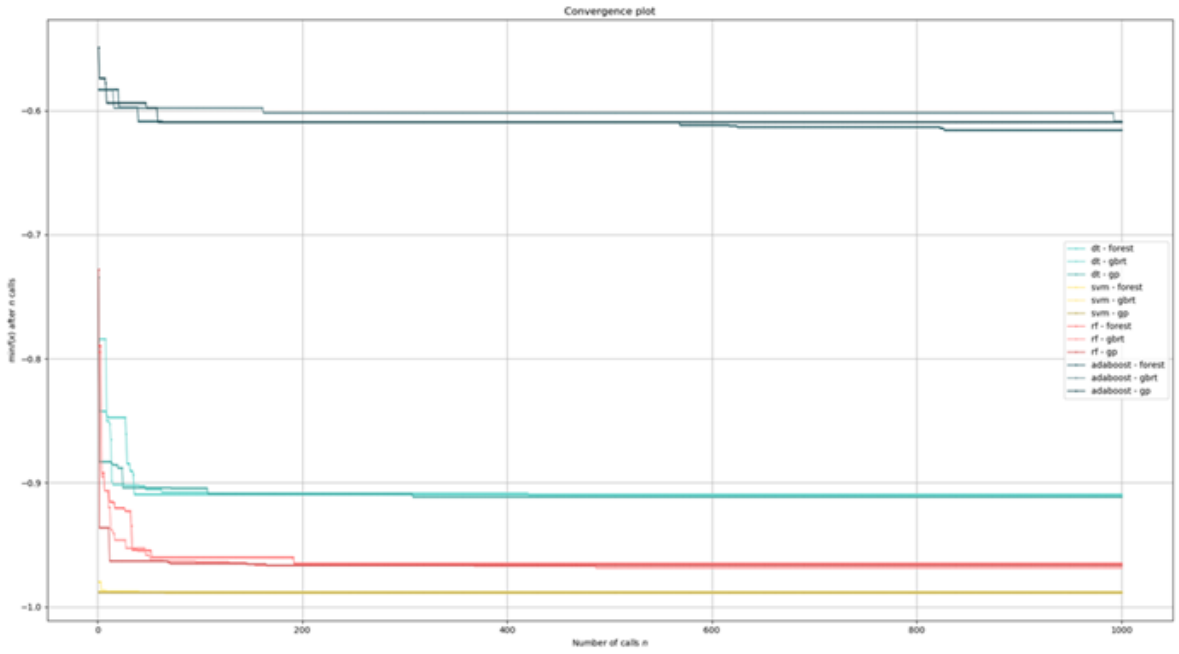

Fig.6. The convergence of the optimization

As can be seen, most classifiers need less than 200 epochs to converge, with only minor improvement seen after 200 epochs. Most classifiers also have relatively significant improvement within 100 epochs of optimization, with the exception of SVM. Adaboosts has around 5% improvement, Decision Trees has around 17% improvement, and Random Forest has around 24% improvement. Interestingly, while SVM has the lowest improvement, at 0 epoch SVM is still better than all of the other classifiers after convergence. The result of each configuration after 1000 epochs can be seen in Table 6.

Table 6. The result of each configuration

#	Classifier	Optimizer	Results	Parameter 1	Parameter 2	Parameter 3	Parameter 4
0	Decision Tree	forest	-0.90900	entropy	65	3	3
1	Decision Tree	gbrt	-0.90954	entropy	897	8	3
2	Decision Tree	Gp	-0.91119	entropy	381	2	1
3	SVM	forest	-0.98786	21.52478	Poly	4	1.043849
4	SVM	gbrt	-0.98896	36.77783	Rbf	8	1.757579
5	SVM	Gp	-0.98841	2.35168	Rbf	7	3.481410
6	Random Forest	forest	-0.96471	199	73	2	1
7	Random Forest	gbrt	-0.96857	62	364	2	1
8	Random Forest	Gp	-0.96636	84	864	2	1
9	Adaboost	forest	-0.61553	23	0.07733	SAMME.R	-
10	Adaboost	Gbrt	-0.60844	30	0.00492	SAMME.R	-
11	Adaboost	Gp	-0.60900	153	0.00095	SAMME.R	-

The best configuration of each classifier is noted, then the parameters of each configuration are applied to the final model training.

4.2. Final Model Test

Each classifier algorithm is then trained for the final model test, using the parameters of the best configuration of each classifier. After training, the model is tested using a test dataset with the result that can be seen in Table 7.

Table 7. The tested result

	Decision Tree	SVM	Random Forest	Adaboost
Training time (s)	1.1242	0.5311	1.7507	1.8799
Accuracy Score	0.9471	0.9846	0.9626	0.7753
Precision	0.9602	0.9896	0.9893	0.8266
Recall	0.9147	0.9830	0.9332	0.5714
Training time (s)	1.1242	0.5311	1.7507	1.8799

As shown in Table 3, SVM scores the best in all of the measurement metrics compared to all of the other classifiers. Random Forest closely follows SVM, especially in the Precision score with the difference of ±0.0003 points. The Adaboost classifier scores the worst with a relatively significant margin. While the other classifiers scores above 94% of accuracy, Adaboost falls behind with 77% accuracy. The confusion matrix of the SVM test can be seen in Fig.7.

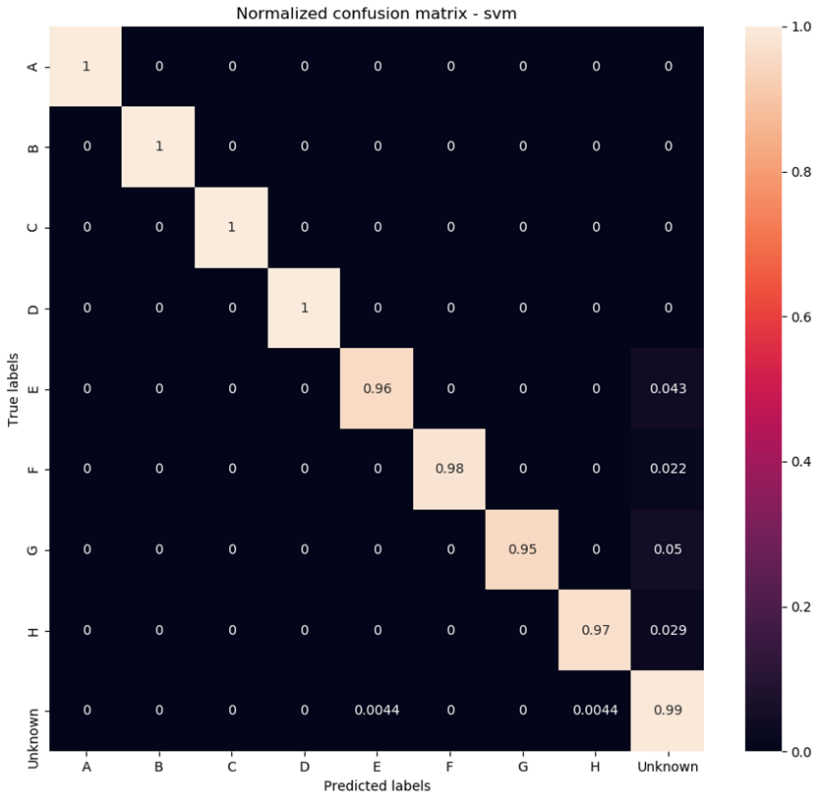

Fig.7. The confusion matrix of SVM

These results are also confirmed by research that has been done in the last five years that used SVM as a classifier for face recognition and has been proven in making a high level of accuracy. This SVM classifier algorithm is used for face recognition research with various purposes, such as P. C. Vasanth and K. R. Nataraj [38] used SVM for expression classification and produced good accuracy in displaying the expression and facial action units. Rustam and Ruvita [39] also used the SVM classifier for face recognition systems to perform gender classification by comparing the polynomial kernel ($d = 5$) and RBF kernel ($\sigma = 0.05$) the and the result is SVM with a polynomial kernel ($d = 5$) reached 100% of accuracy. Rayani and K. Rajakumar [40] obtained an accuracy of 85% while using SVM for face detection and face recognition. Another one is Sharma and Sachdeva [41] who obtained an accuracy value of 98.7% by utilizing SVM as a classifier for face recognition. Prakash and Singh [42] reviewed numerous papers and concluded that SVM has a more effective technique than the others and SVM can be trained successfully for face recognition and SVM is also a better learning algorithm compare to the nearest center approach in terms of face recognition, And the last, P. S. Hiremath et al. [43] and A. Tofighi et al. [44] have been comparing the SVM and K-Nearest Neighbor (KNN) for face recognition and got the result that SVM performs better than KNN with the scores are 99.2% on 3D face recognition [43] and 93.5% on recognizing the face of an image and a video [44].

5. Conclusions

An analysis has been made for the performances of four different classifiers which are SVM, decision tree, random forest, and Adaboost. This is done by calculating some metrics including recall, F1-score, accuracy, and precision. From the experiment, we obtain that SVM gives the best performance in all measurement matrices and reached an accuracy score of 98%. Hyperparameter tuning is proven to improve the performance of a classifier, especially in terms of accuracy, for example, SVM. In previous studies, it showed that SVM is not the best classifier compare to other classifiers, but when we optimize the hyperparameter like in this experiment, SVM has the highest accuracy than others. It proves that hyperparameters are critical in building a robust and accurate model of the classifier. This study also enriches a reference to the comparison of the classifier algorithm that developers/programmers can use in choosing the best classifier algorithm according to their needs. Further, this research is expected to continue by modeling the relationship between each hyperparameter, hence, it will be easier to tune and obtain the best classifier as needed. Another, we can also make another research to obtain a high accuracy value classifier that also has the best performance in a real-time embedded system with less time-consuming, efficient, and cost-effectively, due to face recognition creates a potential for many businesses aspect.

References

[1] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld, "Face recognition: A literature survey," *ACM Computing Surveys*, vol. 35, no. 4, pp. 399-458, 2003.

[2] M. Nusseck, D. W. Cunningham, C. Wallraven, and H. H. Bulthoff, "The contribution of different facial regions to the recognition of conversational expressions," *Journal of Vision*, vol. 8, no. 8, pp. 1-13, 2008.

[3] S. Umer, B. C. Dhara, and B. Chanda, "Face recognition using fusion of feature learning techniques," *Measurement*, vol. 146, pp. 43-54, 2019.

[4] L. Li, X. Mu, S. Li, and H. Peng, "A review of face recognition technology," *IEEE Access*, vol. 8, pp. 139110 – 139120, 2020.

[5] D. N. Parmar and B. B. Mehta, "Face recognition methods and applications," *Int. J. Comput. Tech. & Appl.*, vol. 4, no. 1, pp. 84-86, 2013.

[6] K. Wankhede, B. Wukkadada, and V. Nadar, "Just walk-out technology and its challenges: A case of Amazon Go," *IEEE 2018 Int. Conf. Inventive Research in Comput. Appl. (ICIRCA)*, 2018.

[7] Subrat Kumar Rath, Siddharth Swarup Rautaray,"A Survey on Face Detection and Recognition Techniques in Different Application Domain", IJMECS, vol.6, no.8, pp.34-44, 2014.DOI: 10.5815/ijmecs.2014.08.05

[8] A. Kumar, A. Kaur, and M. Kumar, "Face detection techniques: A review," *Artificial Intelligence Review*, vol. 52, pp. 927-948, 2018.

[9] R. Khokher, R. C. Singh, and R. Kumar, "Footprint recognition with principal component analysis and independent component analysis," *Macromolecular Symp.*, vol. 347, no. 1, pp. 16-26, 2015.

[10] Z. Sufyanu, F. S. Mohamad, A. A. Yusuf, and A. Nuhu, "Feature extraction methods for face recognition," *Int. J. Applied Eng. Research (IRAER)*, vol. 5, pp. 5658-5668, 2016.

[11] P. Probst, A. Boulesteix, and B. Bischl, "Tunability: Importance of hyperparameters of machine learning algorithms," *J. Machine Learning*, vol. 20, pp. 1-32, 2019

[12] P. Gaspar, J. Carbonell, and J. Oliveira, "On the parameter optimization of support vector machines for binary classification," *J. Integrative Bioinformatics*, vol. 9, no. 3, pp. 33-43, 2012.

[13] E. Kremic and A. Subasi, "Performance of random forest and SVM in face recognition," *The Int. Arab J. Inf. Tech.*, vol. 13, no. 2, pp. 287-293, 2016.

[14] A. I. Salhi, M. Kardouchi, and N. Belacel, "Fast and efficient face recognition system using random forest and histogram of oriented gradients," *IEEE 2012 The Int. Conf. The Biometrics Special Interest Group (BIOSIG)*, 2012.

[15] H. Mady, and S. M. S. Hilles, S.M.S. "Face recognition and detection using random forest and combination of LBP and HOG features," *IEEE Int. Conf. Smart Comput. and Electronic Enterprise (ICSCEE2018)*, 2018.

[16] N. Sabri, J. Henry, and Z. Ibrahim, "A comparison of face detection classifier using facial geometry distance measure," *2018 9th IEEE Control and Sys. Grad. Research Coll. (ICSGRC 2018)*, 2018.

[17] W. Changyuan, X. Pengxiang, L. Guang, and W. Qiyou, "A comparative study of face recognition classification algorithms," *Int. J. Adv. Network, Monitoring, and Controls*, vol. 5, no. 3, pp. 23-29, 2020.

[18] A. Nabatchian, I. Makaremi, E. Abdel-Raheem, and M. Ahmadi, "Pseudo-Zernike moment invariants for recognition of faces using different classifiers in FERET database," *IEEE Third Int. Conf. on Convergence and Hybrid Inf. Tech.*, 2008.

[19] D. A. Salazar, J. I. Velez, and J. C. Salazar, "Comparison between SVM and logistic regression: Which one is better to discriminate?," *Revista Colombiana de Estadística*, vol. 35, no. SPE2, pp. 223–237, 2012.

[20] S. Sperandei, "Understanding logistic regression analysis," *Biochemia Medica*, vol. 24, no. 1, pp. 12–18, 2014.

[21] C. C. Chang and C. J. Lin, "LIBSVM: A library for support vector machine," 2021. [Online]. Available: https://www.csie.ntu.edu.tw/~cjlin/papers/libsvm.pdf. [Accessed: 25-June- 2021].

[22] 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.

[23] B. Kami´nski, M. Jakubczyk, and P. Szufel, "A framework for sensitivity analysis of decision trees," *Central Eur. J. Oper. Research*, vol. 26, pp. 135–159, 2017.

[24] T. Hastie, R. Tibshirani, and J. Friedman, The Elements of statistical learning: Data mining, inference, and prediction, 2nd ed., Springer, 2009.

[25] L. Breiman and A. Cutler, "Random Forests", 2021. [Online]. Available: https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm. [Accessed: 27-July-2021].
[26] L. Breiman, J. Friedman, R. Olshen, and C. Stone, Classification and Regression Trees, 1st ed., Wadsworth, Belmont, CA, 1984.
[27] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, pp. 832–844, 1998.
[28] E. Scornet, "Tuning parameters in random forests," *ESAIM: Proceedings and Surveys*, vol. 60, pp.144-162, 2018.
[29] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5-32, 2001.
[30] R. Wang, "AdaBoost for feature selection, classification and its relation with SVM, A Review," *Physics Procedia*, vol. 25, pp. 800–807, 2012.
[31] C. Papageorgiou, M. Oren, and T. Poggio, "A general framework for object detection," *IEEE The Sixth Int. Conf. on Comput. Vision*, 1998.
[32] J. Zhu, S. Rosset, H. Zou, and T. Hastie, "Multi-class Adaboost," 2006. [Online]. Available: https://web.stanford.edu/~hastie/Papers/samme.pdf. [Accessed: 25-June-2021].
[33] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. of Comput. and Sys. Scie.*, vol. 55, pp. 119-139, 1997.
[34] Shivanand S. Gornale, Pooja U. Patravali, Kiran S. Marathe, Prakash S. Hiremath," Determination of Osteoarthritis Using Histogram of Oriented Gradients and Multiclass SVM", International Journal of Image, Graphics and Signal Processing(IJIGSP), Vol.9, No.12, pp. 41-49, 2017.DOI: 10.5815/ijigsp.2017.12.05
[35] A. P. Singh and A. Kumar, "Robust face recognition system using HOG features and SVM classifier," *Int. J. of Information Scie. and Appl. (IJISA)*, vol. 11, no. 1 (special issue), pp. 105 -109, 2019.
[36] S. Chang, D. Xiaoqing, and F. Chi, "Histogram of the oriented gradient for face recognition," *Tsinghua Scie. and Tech.*, vol. 16, no. 2, pp. 216-224, 2011.
[37] Y. Li , Z. Wang, Y. Li, X. Zhao, and H. Huang, " Design of face recognition system based on CNN," *J. Phys.: Conf. Ser.*, vol. 16, no.1, 2020.
[38] V. P. C. and N. K. R., "Facial expression recognition using SVM classifier," *Indonesian J. of Elect. Eng. and Inform. (IJEEI)*, vol. 3, no. 1, pp. 16-20, 2015.
[39] Z. Rustam and A. A. Ruvita, "Application support vector machine on face recognition for gender classification," *J. of Phys.: Conf. Series*, vol. 1108, no. 012067, 2018.
[40] C. Rayani and R. K., "Face detection and recognition using support vector machine," *Int. J. of Eng. and Adv. Tech. (IJEAT)*, vol. 8, no. 4, pp. 382-384, 2019.
[41] S. Sharma and K. Sachdeva, "Face recognition using PCA and SVM with surf technique," *Int. J. of Comput. Appl.*, vol. 129, no. 4, pp. 41-46, 2015.
[42] N. Parakash and Y. Singh, "Support vector machine for face recognition," *Int. Research J. of Eng. and Tech. (IRJET)*, vol. 2, no. 8, pp. 1517-1529, 2015.
[43] P. S. Hiremath, Manjunatha Hiremath,"3D Face Recognition based on Radon Transform, PCA, LDA using KNN and SVM", IJIGSP, vol.6, no.7, pp.36-43, 2014.DOI: 10.5815/ijigsp.2014.07.05
[44] Alireza Tofighi, Nima Khairdoost, S. Amirhassan Monadjemi, Kamal Jamshidi,"A Robust Face Recognition System in Image and Video", IJIGSP, vol.6, no.8, pp.1-11, 2014.DOI: 10.5815/ijigsp.2014.08.01

Authors' Profiles

M. Ilham Rizqyawan was born in 1990. He received B.Eng. Degree in Computer Science/Informatics from Indonesian Computer University (UNIKOM) in 2013 and M.Eng. in electrical engineering from Institut Teknologi Bandung (ITB) in 2017. He currently works in Technical Implementation Unit for Instrumentation Development, Indonesian Institute of Sciences, Indonesia as a researcher. His research interests include applied artificial intelligence in healthcare, human-computer interaction, and data visualization.

Ulfah Nadiya was born in 1996. She received BSc degree in Electrical Engineering from Institut Teknologi Bandung, Indonesia in 2017, MSc in Electrical Engineering from Institut Teknologi Bandung, Indonesia in 2019. Her research interests include information security, internet of things (IoT), and healthcare. She is currently a researcher in Technical Implementation Unit for Instrumentation Development, Indonesian Institute of Sciences, Indonesia.

Aris Munandar, Jony Winaryo Wibowo, Oka Mahendra, Irfan Asfy Fakhry Anto, Rian Putra Pratama, Muhammad Arifin, and Hanif Fakhrurroja are currently a researcher in Technical Implementation Unit for Instrumentation Development, Indonesian Institute of Sciences, Indonesia.

How to cite this paper: M. Ilham Rizqyawan, Ulfah Nadiya, Aris Munandar, Jony Winaryo Wibowo, Oka Mahendra, Irfan Asfy Fakhry Anto, Rian Putra Pratama, Muhammad Arifin, Hanif Fakhrurroja, "Comparing Performance of Supervised Learning Classifiers by Tuning the Hyperparameter on Face Recognition", International Journal of Intelligent Systems and Applications(IJISA), Vol.13, No.5, pp.1-13, 2021. DOI: 10.5815/ijisa.2021.05.01

