Journal of
Information Systems Engineering
and Business Intelligence

Vol.7, No.1, April 2021
Available online at: http://e-journal.unair.ac.id/index.php/JISEBI

Comparative Analysis of Image Classification Algorithms for Face Mask Detection

Mohammad Farid Naufal[1)*], Selvia Ferdiana Kusuma[2)], Zefanya Ardya Prayuska[3)], Ang Alexander Yoshua[4)], Yohanes Albert Lauwoto[5)], Nicky Setyawan Dinata[6)], David Sugiarto[7)]

[1)3)4)5)6)7)]*Teknik Informatika, Universitas Surabaya, Indonesia*
Jl. Raya Kalirungkut, Surabaya
[1)] faridnaufal@staff.ubaya.ac.id, [3)]s160418044@student.ubaya.ac.id, [4)]s160418032@student.ubaya.ac.id, [5)]s160418047@student.ubaya.ac.id, [6)]s160418066@student.ubaya.ac.id, [7)]s160418025@student.ubaya.ac.id

[2)]*Manajemen Informatika, PSDKU Politeknik Negeri Malang, Indonesia*
Jl. Lingkar Maskumambang No.1, Kediri
[2)]selvia.ferdiana@polinema.ac.id

Abstract

Background: The COVID-19 pandemic remains a problem in 2021. Health protocols are needed to prevent the spread, including wearing a face mask. Enforcing people to wear face masks is tiring. AI can be used to classify images for face mask detection. There are a lot of image classification algorithm for face mask detection, but there are still no studies that compare their performance.
Objective: This study aims to compare the classification algorithms of classical machine learning. They are k-nearest neighbors (KNN), support vector machine (SVM), and a widely used deep learning algorithm for image classification which is convolutional neural network (CNN) for face masks detection.
Methods: This study uses 5 and 3 cross-validation for assessing the performance of KNN, SVM, and CNN in face mask detection.
Results: CNN has the best average performance with the accuracy of 0.9683 and average execution time of 2,507.802 seconds for classifying 3,725 faces with mask and 3,828 faces without mask images.
Conclusion: For a large amount of image data, KNN and SVM can be used as temporary algorithms in face mask detection due to their faster execution times. At the same time, CNN can be trained to form a classification model. In this case, it is advisable to use CNN for classification because it has better performance than KNN and SVM. In the future, the classification model can be implemented for automatic alert system to detect and warn people who are not wearing face masks.

Keywords: Face Masks, KNN, SVM, CNN, Classification, Deep Learning

Article history: Received 1 February 2021, first decision 1 March 2021, accepted 16 March 2021, available online 28 April 2021

I. INTRODUCTION

The COVID-19 pandemic became a major global problem in 2020. To date, it cannot be predicted when the COVID-19 pandemic will be fully under control. Vaccines are underway but the effectiveness cannot be fully assured. The implementation of health protocols is still the most effective way to prevent the transmission, such as by wearing face masks. People are obliged to wear a mask in public places. However, there are still people who do not adhere to health protocols. An autonomous system can be developed to detect unmasked people and alert them.

Classification methods include k-nearest neighbours (KNN) [1], support vector machine (SVM) [2], and deep learning (DL) [3]. Convolutional neural network (CNN) is a popular DL algorithm for image classification. CNN is like a traditional neural network because it consists of neurons that can learn. Each neuron will receive an input and perform operations; and these are connected in a network [4]. CNN has several stages of operation, namely convolutional layers, pooling layers, and fully connected layers. CNN is designed to handle input with a two-dimensional shape. Each layer in the network is composed of multi two-dimensional planes, and each plane consists of multi-independent neurons. Neurons that are in two adjacent layers will be connected to each other [5]. CNN is leading in the field of image recognition [6] .

* Corresponding author

ISSN 2443-2555 (online) 2598-6333 (print)
doi: http://dx.doi.org/10.20473/jisebi.7.1.56-66

There are many studies on face detection using KNN, SVM, and CNN. Reference [7] used KNN for face detection on low power processor and achieve 91.5% of accuracy. Reference [8] used SVM for face detection and achieve 90.82% of accuracy. Reference [9] used SVM for face detection in YALE and ORL database and achieve 96% and 96.5% of accuracy respectively. Reference [10] used CNN for face recognition with pose and illumination variation and achieve 99.5% of accuracy on AR database; and 85.13% of accuracy on FERET database. From those previous studies, SVM, KNN, and CNN are effective algorithms for classifying face images. Therefore, we consider comparing KNN, SVM, and CNN for face mask detection in the same dataset. We expect that this would result in comparable performance.

The purpose of this research is to compare and analyse the performance of classic machine learning algorithms, which are SVM and KNN, with DL algorithm, namely CNN. The feature used for classifying face masks is the pixel intensity. The performance being compared includes accuracy, precision, recall, F1 score, and execution time. The result is expected to be useful for researchers in choosing the best and most suitable algorithm for face masks detection.

The remaining of this paper consists of 4 parts. Chapter 1 explains the background to the research. Chapter 2 describes the research methodology. Chapter 3 explains the results and discussion. Chapter 4 concludes the paper.

II. Related Works

Jagadeeswari et al. [11] compared MobileNetV2, RESNET50, and VGG16; with Adam Optimizer, ADAGRAD, and SGD. Adam optimizer and MobileNetV2 resulted in the best accuracy. Han et al. [12] performed an object detection focusing on real time face masks detection in supermarket using single-shot detector (SSD). Vinitha et al. [13] used CNN with MobileNetV2 architecture, and library of OpenCV, tensorflow, keras, and Pytorch to detect whether people were wearing a face mask or not. Nagrath et al. [14] used SSDMNV2 (Single Shot Multibox Detector as a face detector and MobileNetV2) to perform real-time face masks detection. Ge et al. [15] propose LLE (Locally Linear Embedding) - CNNs for face masks detection. It combines two pre-trained CNNs for extracting facial regions before being described by LLE. Grassi et al. [16] used Discrete Cosine Transform (DCT) for feature extraction with Multilayer Perceptron (MLP) and RBF Neural Network as classifier.

Nevertheless, none of these studies has tried to compare classic machine learning algorithms, namely KNN and SVM, with deep learning, namely CNN, to compare the performance. The performance to be analyzed is accuracy, precision, recall, F1 score, and execution time. According to Shustanov et al. [17], CNN requires good hardware specifications that affect execution time. This study tries to consider this in relation to CNN performance, when compared to the KNN and SVM.

III. Methods

The research methodology consists of five stages, which are collecting datasets, forming classification models, training classification models, testing classification models, and calculating performance. Fig. 1 shows the flow of the research methodology.

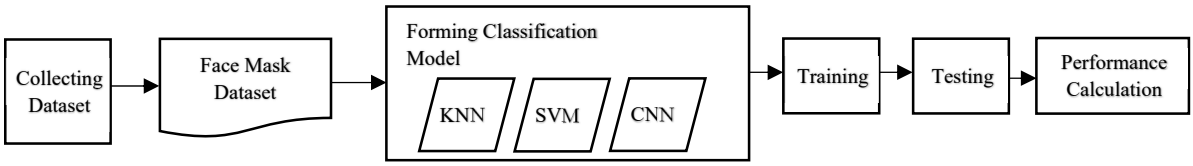

Fig 1. Research Method

A. Collecting Dataset

The dataset used in this study is a face mask images obtained from Kaggle [18]. It is public dataset which consists of faces with and without a mask. Fig. 2 shows examples of a face with and without a mask. The number of images of faces with a face mask is 3,725 and without a face mask is 3,828. Because the face mask image dataset has different image sizes, each image is resized to become 64x64 pixels.

(a) (b)

Fig. 2. Example of face without masks (a) and face with masks (b)

B. Forming Classification Model

At this stage, the various parameters from the KNN, SVM, and CNN algorithms are configured. The goal is to determine the effect of the parameters on the algorithm performance.

1) KNN

The parameter configurations used in the KNN algorithm are the number of neighbours and the type of distance. The number of neighbours used is 5, 7, and 9, while the types of distance used are Euclidean and Minkowski. Each distance is tested with the number of neighbours of 5, 7, and 9. The Euclidean distance formula can be seen in (1) where d is the Euclidean distance, n is the number of attributes, p_i is the attribute value of first dataset, q_i is the attribute value of second dataset. Minkowski distance formula can be seen in (2) where d is the Minkowski distance, n is the number of attributes, p_i is the attribute value of first dataset, q_i is the attribute value of second dataset.

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2} \quad (1)$$

$$d(p,q) = \sum_{i=1}^{n}|q_i - p_i| \quad (2)$$

2) SVM

The parameter configuration in SVM algorithm is the kernel types, which are Linear, Poly, and RBF. At the training stage, one of the best performing kernels will be selected. Linear kernel formula can be seen in (3) where x and x' are two samples. Poly kernel formula can be seen in (4) where x and x' are two samples, and d is the degree of the polynomial. RBF kernel formula can be seen in (5) where x and x' are two samples.

$$k(x,x') = x^T x' \quad (3)$$

$$K(x,x') = (x.x' + 1)^d \quad (4)$$

$$K(x,x') = exp\left(-\frac{||x-x'||^2}{2\sigma^2}\right) \quad (5)$$

3) CNN

The configuration parameters in the CNN algorithm are the number of epochs, the convolution model, the type of activation function, the number of dense layers, and the number of batch sizes. Table 1 shows the CNN parameters.

This study uses Keras library [19], which is an open source library for designing deep learning architecture. Keras can run on artificial intelligence frameworks such as TensorFlow. Currently, Keras is considered as one of the best machine learning libraries in Python. It also provides some of the best utilities when it comes to building models, processing datasets, visualising graphs, and other functions.

TABLE 1
CNN PARAMETERS

Parameters	Tensor Shape	Description
Convolution-1	(62, 62, 32)	Filter Size = 3x3. Activation = ReLu
MaxPooling-1	(31, 31, 32)	Pool Size = 2
Convolution-2	(29, 29, 32)	Filter Size = 3x3. Activation = ReLu
MaxPooling-2	(14, 14, 32)	Pool Size = 2
Flatten	(6272)	-
Dense Input	(128)	Activation = ReLu
Dense Output	(3)	Activation = Softmax
Optimizer	-	Adam
Number of epoch	-	50
Batch Size	-	8

The activation function at the two convolutions and dense stages in the first layer is the Rectified Linear Unit (ReLu). The ReLu activation function formula can be seen in (6). x is the value inputted into the activation function.

$$R(x) = \max(0, x) \tag{6}$$

The activation function on the second dense layer (output layer) is softmax. The softmax activation function formula can be seen in (7). x_i is the input value from the previous layer, n is the number of labels, and j is the order of the labels.

$$S(x_i) = \frac{e^{x_i}}{\sum_j^n e^{x_j}} \tag{7}$$

The epoch used during the training process is 50. The optimizer for updating the edge weight for each layer is Adam Optimizer. The batch size is 8.

4) Computer Specifications

All classification models that are formed, the training and testing process are executed on a computer with following specification Intel® Core™ i3-3220 CPU @ 3.30GHz (4 CPUs), 8192 MB of RAM, Windows 10 Education 64 bit, AMD Radeon HD 5500 Series, and 1 GB GPU Total Memory. This specification information is important because it affects the performance of the execution time.

C. Training

At the training and testing stages, the dataset is divided into two parts with the proportion of 80% of training and 20% of testing for the first experiment; and 66.66% of training and 33.33% of testing for the second experiment. So that the number of cross validations is 5 and 3. The distribution of training and testing data is generated randomly.

Training is carried out for the KNN, SVM, and CNN algorithms using the parameters described in the previous sub-chapter. In the CNN algorithm, there is data augmentation to enrich the variation of the dataset. The more varied the training dataset, the more useful it is to avoid overfitting. The types of data augmentation are horizontal flip, shear range, and zoom range. The data augmentation was performed using Keras library [19].

Horizontal flip is used to duplicate the training dataset by rotating the image by 90 degrees. Shear range [10] performs a shear transformation, which is used to rotate the image to a certain degree according to the parameter. Zoom range is used to enlarge an image into a certain size according to the parameters.

D. Testing

The testing stage is carried out to validate the model that has been formed at the training stage. Testing is done at each cross validation for the KNN, SVM, and CNN algorithms. Testing using cross validation aims to see whether the model built on the algorithm has a stable performance or not.

In the CNN algorithm, validation is carried out in each epoch using data testing. If there is a model in a particular epoch that has the best performance, that model is saved. The best model store uses checkpoints in Keras library.

E. Performance Calculation

At this stage, the process of calculating the performance of the testing phase in each algorithm is carried out. The performance metrics used are accuracy, precision, recall, F1 score, and the execution time of the training and testing process. Each algorithm with its respective parameters is calculated for its performance in each cross validation. The

parameter in an algorithm that has the best performance will be selected and then compared with another algorithm that has the best performance.

Equation (8) shows the calculation accuracy, which is used to calculate the total of True Positive (TP) and True Negative (TN) divided by the total of TP, TN, False Positive (FP), and False Negative (FN).

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

Equation (9) shows the calculation formula of Precision, which is calculated by dividing TP by the total of TP and FP.

$$Precision = \frac{TP}{TP+FP} \quad (9)$$

Equation (10) shows the calculation formula for recall, which is calculated by dividing TP by the total of TP and FN.

$$Recall = \frac{TP}{TP+FN} \quad (10)$$

Equation (11) shows the calculation formula of the F1 score, which is calculated by dividing the multiplication of Precision with Recall and the addition of Precision and Recall.

$$F1\ Score = \frac{Precision\ x\ Recall}{Precision+Recall} \quad (11)$$

IV. RESULTS

The results of face mask detection using the KNN, SVM, and CNN algorithms were obtained and compared in terms of performance and execution time.

A. Test Results of KNN Algorithm

Table 2 shows the test results of the KNN algorithm with the specified distance and neighbours' parameters. Dist is the type of distance, NN is the Number of neighbours, AVG is the average metric value, and AVG Perf is the sum of the average metric accuracy, precision, recall, and F1 score divided by 4. AVG Perf is used to see the overall average performance metric.

It can be seen that there is no significant difference in performance with the parameters used in the KNN. However, in this study 5 KNNs with the Euclidean distance type which had an AVG Perf of 0.8115 were chosen to be compared with the SVM and CNN algorithms because they had the best AVG Perf.

B. Test Results of the SVM Algorithm

Table 3 shows the test results on the SVM algorithm. There is a significant difference in performance between kernels. It can be seen that the SVM algorithm with the RBF kernel type has the best AVG Perf, which is 0.8721. Furthermore, the SVM performance with the Linear kernel will be compared with the KNN and CNN algorithms.

C. Test Results of CNN Algorithm

Table 4 shows the test results on the CNN algorithm. The performance produced by CNN is relatively good, which has AVG Perf 0.9683. From each cross-validation, the performance generated by CNN is also quite stable. All performance metrics yield values above 0.96.

D. Performance Comparison

Table 5 shows the comparison results of performance metrics from the KNN, SVM, and CNN algorithm. The KNN and SVM algorithm use the parameters with best performance as described in the previous section. It can be seen that CNN has the best performance compared to the KNN algorithm with the number of neighbours 5 and SVM with the RBF kernel. CNN has the best accuracy, precision, recall, and F1 score. Fig. 3 and Fig. 4 shows a graph of the performance comparison of the KNN, SVM, and CNN algorithms for 5 and 3 cross-validation respectively.

The gap between the KNN and SVM algorithm is around 0.062. Meanwhile, the gap between CNN and KNN and CNN with SVM is 0.1552 and 0.0968, respectively. The difference in performance between KNN and SVM is not far enough, but KNN and SVM show a significant gap compared to CNN.

TABLE 2
TEST RESULT OF KNN

Dist	NN	Metric	5-Cross Validation (80% Train 20% Test)					3-Cross Validation (66.66% Train 33.33% Test)			AVG	AVG PERF
			1	2	3	4	5	1	2	3		
EUCLIDEAN	5	ACC	0.7988	0.8127	0.8140	0.8192	0.8126	0.8125	0.8022	0.7974	0.8087	0.8115
		PREC	0.8143	0.8238	0.8228	0.8321	0.8301	0.8296	0.8151	0.8102	0.8223	
		REC	0.7988	0.8127	0.8140	0.8192	0.8126	0.8125	0.8022	0.7974	0.8087	
		F1SCORE	0.7971	0.8100	0.8121	0.8174	0.8096	0.8095	0.8002	0.7946	0.8063	
	7	ACC	0.8001	0.8008	0.8140	0.8099	0.8066	0.8086	0.7994	0.7942	0.8042	0.8073
		PREC	0.8157	0.8130	0.8257	0.8235	0.8253	0.8266	0.8172	0.8079	0.8194	
		REC	0.8001	0.8008	0.8140	0.8099	0.8066	0.8086	0.7994	0.7942	0.8042	
		F1SCORE	0.7984	0.7976	0.8116	0.8079	0.8033	0.8053	0.7967	0.7912	0.8015	
	9	ACC	0.8001	0.8008	0.8140	0.8099	0.8066	0.8086	0.7994	0.7942	0.8042	0.8080
		PREC	0.8143	0.8238	0.8228	0.8321	0.8301	0.8296	0.8151	0.8102	0.8223	
		REC	0.8001	0.8008	0.8140	0.8099	0.8066	0.8086	0.7994	0.7942	0.8042	
		F1SCORE	0.7984	0.7976	0.8116	0.8079	0.8033	0.8053	0.7967	0.7912	0.8015	
MINKOWSKI	5	ACC	0.7988	0.8127	0.8140	0.8192	0.8126	0.8125	0.8022	0.7974	0.8087	0.8110
		PREC	0.8143	0.8238	0.8228	0.8321	0.8301	0.8296	0.8151	0.8102	0.8223	
		REC	0.7988	0.8127	0.8140	0.8127	0.8126	0.8125	0.8022	0.7974	0.8079	
		F1SCORE	0.7971	0.8100	0.8121	0.8100	0.8096	0.8095	0.8002	0.7946	0.8054	
	7	ACC	0.8001	0.8008	0.8140	0.8099	0.8066	0.8086	0.7994	0.7942	0.8042	0.8067
		PREC	0.8157	0.8130	0.8257	0.8235	0.8253	0.8266	0.8172	0.8079	0.8194	
		REC	0.8001	0.8008	0.8140	0.8008	0.8066	0.8086	0.7994	0.7942	0.8031	
		F1SCORE	0.7984	0.7976	0.8116	0.7976	0.8033	0.8053	0.7967	0.7912	0.8002	
	9	ACC	0.8001	0.8008	0.8140	0.8099	0.8066	0.8086	0.7994	0.7942	0.8042	0.8074
		PREC	0.8143	0.8238	0.8228	0.8321	0.8301	0.8296	0.8151	0.8102	0.8223	
		REC	0.8001	0.8008	0.8140	0.8008	0.8066	0.8086	0.7994	0.7942	0.8031	
		F1SCORE	0.7984	0.7976	0.8116	0.7976	0.8033	0.8053	0.7967	0.7912	0.8002	

TABLE 3
TEST RESULT OF SVM

Kernel	Metric	5-Cross Validation (80% Train 20% Test)					3-Cross Validation (66.66% Train 33.33% Test)			AVG	AVG PERF
		1	2	3	4	5	1	2	3		
LINEAR	ACC	0.8451	0.8418	0.8498	0.8583	0.8722	0.85068	0.84869	0.84029	0.8509	0.8505
	PREC	0.8467	0.8418	0.8499	0.8583	0.8583	0.85079	0.84907	0.84080	0.8494	
	REC	0.8451	0.8418	0.8498	0.8583	0.8722	0.85068	0.84869	0.84029	0.8509	
	F1SCORE	0.8451	0.8418	0.8498	0.8583	0.8721	0.85069	0.84866	0.84012	0.8508	
POLY	ACC	0.7684	0.7737	0.7829	0.7623	0.802	0.75576	0.75814	0.75765	0.7701	0.7719
	PREC	0.7949	0.781	0.7902	0.7792	0.7792	0.77061	0.78058	0.77460	0.7813	
	REC	0.7684	0.7737	0.7829	0.7623	0.802	0.75576	0.75814	0.75765	0.7701	
	F1SCORE	0.7644	0.7708	0.7808	0.7585	0.7996	0.75150	0.75332	0.75279	0.7665	
RBF	ACC	0.8756	0.8676	0.8584	0.8788	0.8808	0.86378	0.88125	0.86770	0.8717	0.8721
	PREC	0.8757	0.8701	0.8636	0.8801	0.8801	0.86655	0.88274	0.86884	0.8735	
	REC	0.8756	0.8676	0.8584	0.8788	0.8808	0.86378	0.88125	0.86770	0.8717	
	F1SCORE	0.8755	0.8677	0.8582	0.8787	0.8808	0.86366	0.88113	0.86769	0.8717	

TABLE 4
TEST RESULT OF CNN

Metric	Cross Validation (80% Train 20% Test)					Cross Validation (66.66% Train 33.33% Test)			AVG	AVG PERF
	1	2	3	4	5	1	2	3		
ACC	0.9636	0.9702	0.9682	0.9709	0.9742	0.9631	0.9670	0.9690	0.9683	0.9683
PREC	0.9636	0.9702	0.9683	0.9709	0.9742	0.9631	0.9670	0.9690	0.9683	
REC	0.9636	0.9702	0.9682	0.9709	0.9742	0.9631	0.9670	0.9690	0.9683	
F1SCORE	0.9636	0.9702	0.9682	0.9709	0.9742	0.9631	0.9670	0.9690	0.9683	

TABLE 5
PERFORMANCE COMPARISON

Algorithm	Metric	Cross Validation					Cross Validation			AVG	AVG PERF
		1	2	3	4	5	1	2	3		
KNN 5 EUCLIDEAN	ACC	0.7988	0.8127	0.8140	0.8192	0.8126	0.8125	0.8022	0.7974	0.8087	0.8115
	PREC	0.8143	0.8238	0.8228	0.8321	0.8301	0.8296	0.8151	0.8102	0.8223	
	REC	0.7988	0.8127	0.8140	0.8192	0.8126	0.8125	0.8022	0.7974	0.8087	
	F1SCORE	0.7971	0.8100	0.8121	0.8174	0.8096	0.8095	0.8002	0.7946	0.8063	
SVM RBF	ACC	0.8756	0.8676	0.8584	0.8788	0.8808	0.8638	0.8813	0.8677	0.8717	0.8721
	PREC	0.8757	0.8701	0.8636	0.8801	0.8801	0.8666	0.8827	0.8688	0.8735	
	REC	0.8756	0.8676	0.8584	0.8788	0.8808	0.8638	0.8813	0.8677	0.8717	
	F1SCORE	0.8755	0.8677	0.8582	0.8787	0.8808	0.8637	0.8811	0.8677	0.8717	
CNN	ACC	0.9636	0.9702	0.9682	0.9709	0.9742	0.9631	0.967	0.969	0.9683	0.9683
	PREC	0.9636	0.9702	0.9683	0.9709	0.9742	0.9631	0.967	0.969	0.9683	
	REC	0.9636	0.9702	0.9682	0.9709	0.9742	0.9631	0.967	0.969	0.9683	
	F1SCORE	0.9636	0.9702	0.9682	0.9709	0.9742	0.9631	0.967	0.969	0.9683	

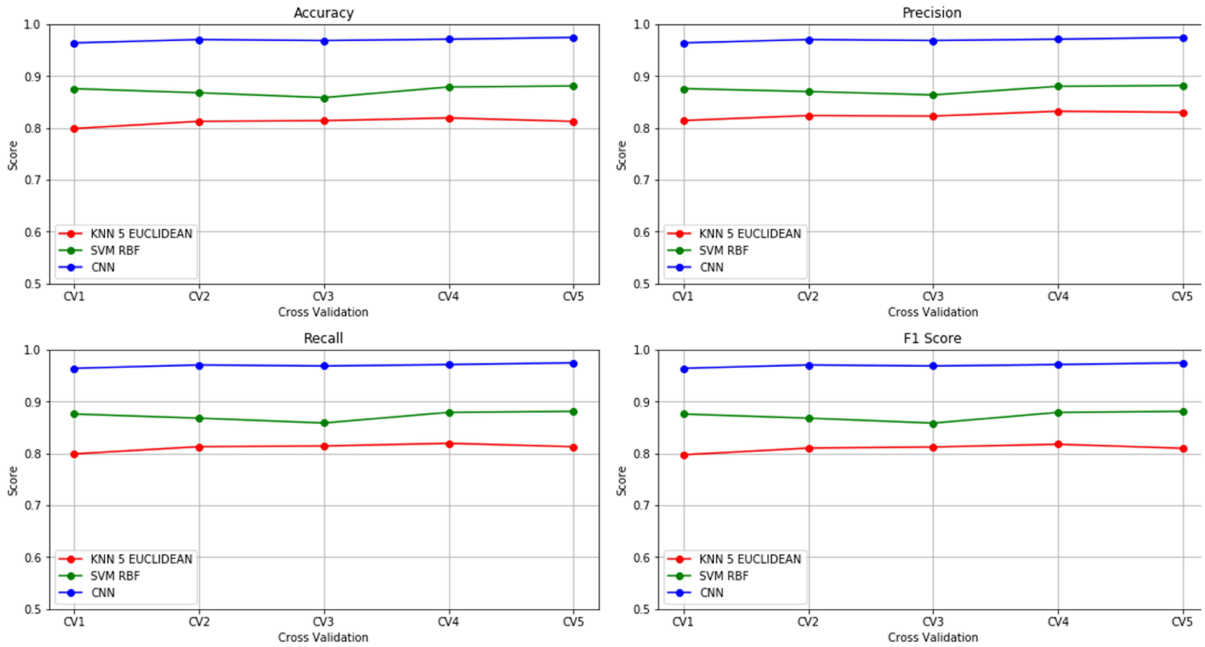

Fig 3. Performance Comparison Graph for 5-Cross Validation

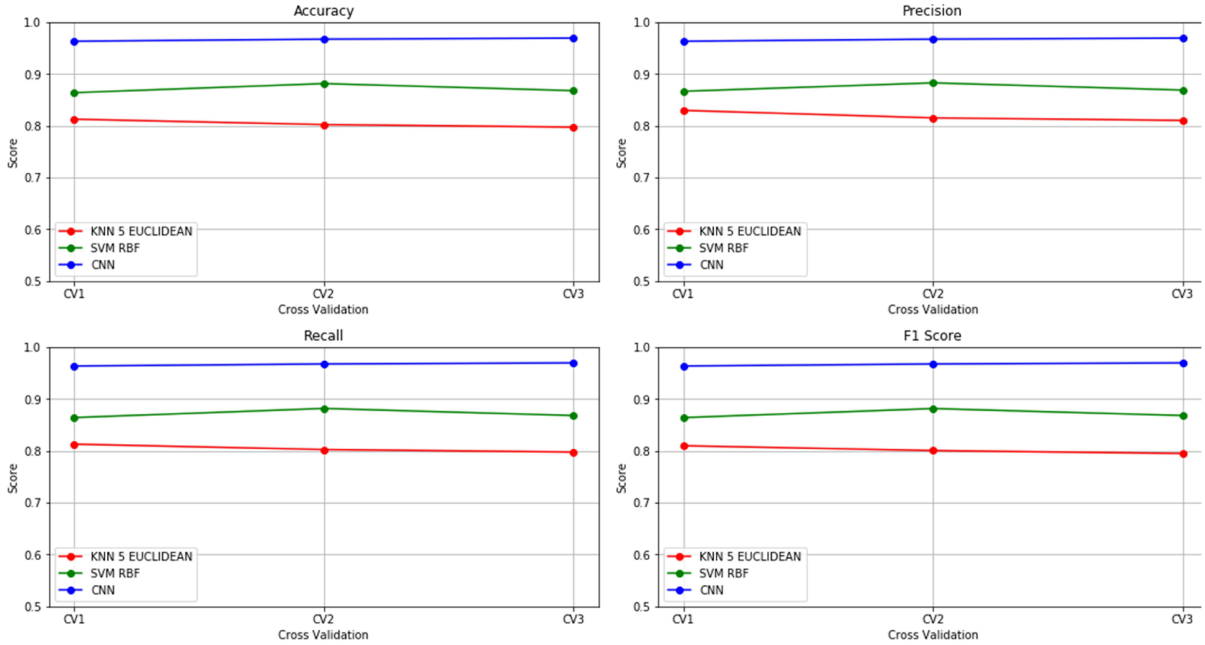

Fig 4. Performance Comparison Graph for 3 cross validation

Table 6 shows the comparison of the training and testing execution time. The calculated execution time is the duration of time the training process and the testing process for each cross-validation. It can be seen that CNN has the longest average execution time of 2,337.34 seconds despite having the best performance. Meanwhile, KNN with 5 neighbours shows the fastest average execution time of 203.62 seconds. SVM has an average execution time that is not far from the Linear KNN, which is around 309.23.

TABLE 6
EXECUTION TIME OF KNN, SVM, AND CNN IN SECOND

Algorithm	Cross Validation					Cross Validation			AVG
	1	2	3	4	5	1	2	3	
KNN 5 EUCLIDEAN	189.15	182.98	188.53	181.73	183.96	270.62	266.87	165.16	203.62
SVM RBF	324.5	322.39	328.08	326.99	327.88	278.69	292.11	273.2	309.23
CNN	2464.88	2662.78	2477.37	2467.79	2466.19	2036.14	2045.66	2077.93	2337.3425

The long execution time for CNN is because there are a lot of epochs being used, which is 50. However, based on the historical data on the training and testing execution time of each epoch, CNN takes an average of 50.5 seconds and obtains an average accuracy of 0.9. This is consistent with the nature of the neural network, which requires more time but has better performance metrics.

From the training and testing CNN model, it can be concluded that the model used is not overfitting, because the accuracy generated in the training data validation is not much different from the accuracy generated when validating the testing data. Fig. 5 shows the loss and accuracy graph of each epoch of CNN for 5-cross validation. Fig. 6 shows the loss and accuracy graph of each epoch of CNN for 3-cross validation.

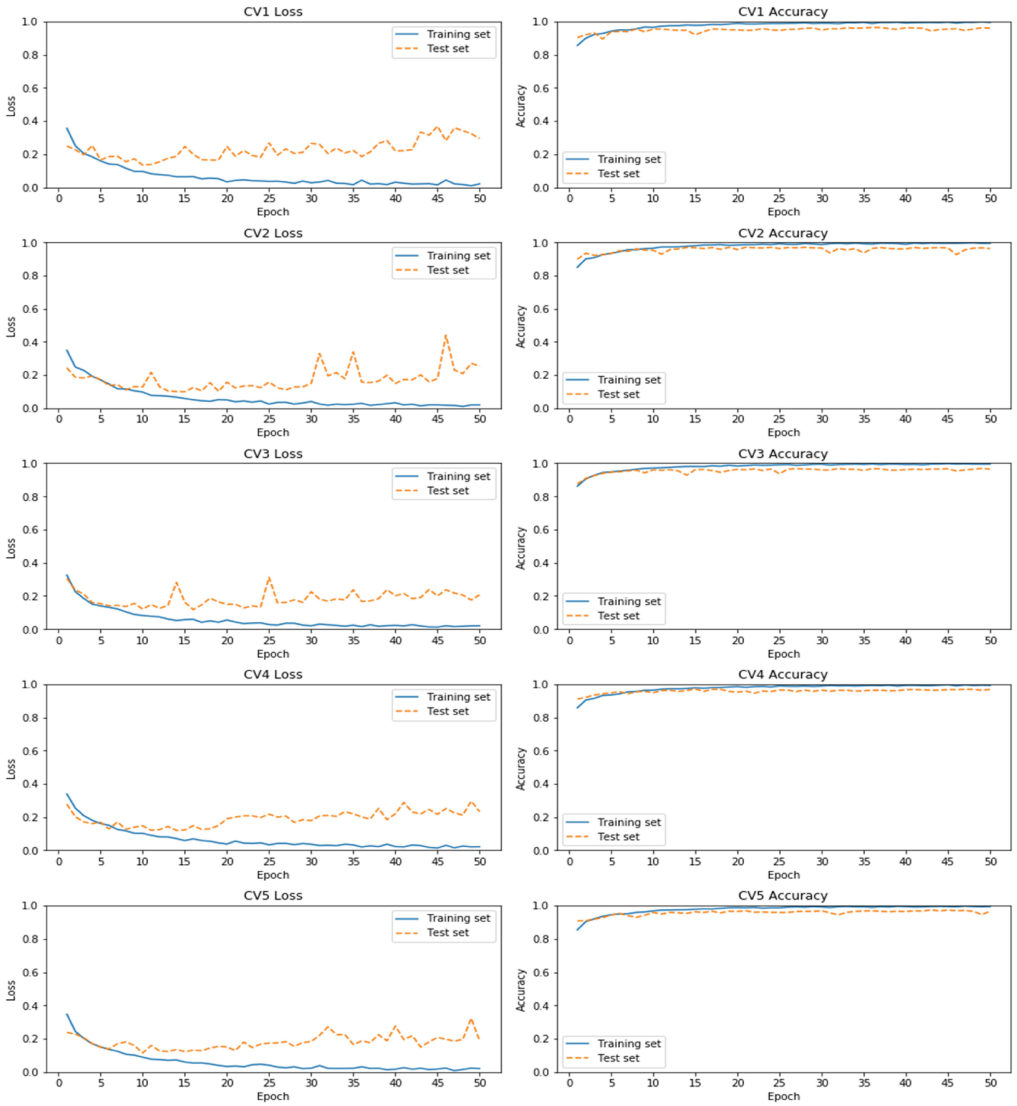

Fig. 5. CNN Loss and Accuracy Graphics for 5-Cross Validation

The KNN algorithm with a Euclidean distance and 5 neighbours, SVM with RBF Kernel, and CNN are classified as relatively stable in performance in each cross validation. KNN, SVM, and CNN algorithms are quite feasible to use in classifying face mask images using only pixel intensity features. However, the CNN algorithm requires a long execution time in the training process.

V. Discussion

CNN needs long time in the training process, but results in good performances. Meanwhile, KNN and SVM is faster, but the resulting performance is not as good as CNN. The relation between execution time and performance can be seen in Table 5 and 6. If researchers have good computational machine and want a good performance model, then CNN execution time is not a problem. KNN and SVM can be chosen if the researchers want a classification model that requires a relatively faster training process.

CNN is a fairly popular algorithm for image classification, and it is proven to have the best performance compared to KNN and SVM for face mask detection. Long execution time is only needed at the beginning to form the CNN model and it is not a problem if CNN is chosen as the face mask classification algorithm. CNN can be used to form a classification model on face mask image dataset using a large number of training datasets. This makes the training

process to form a classification model long. While waiting for the CNN model forming process, researchers can use KNN or SVM to create a temporary classification model because the performance of the two algorithms is still good. After the CNN model is formed then the researchers can use it to perform face mask detection.

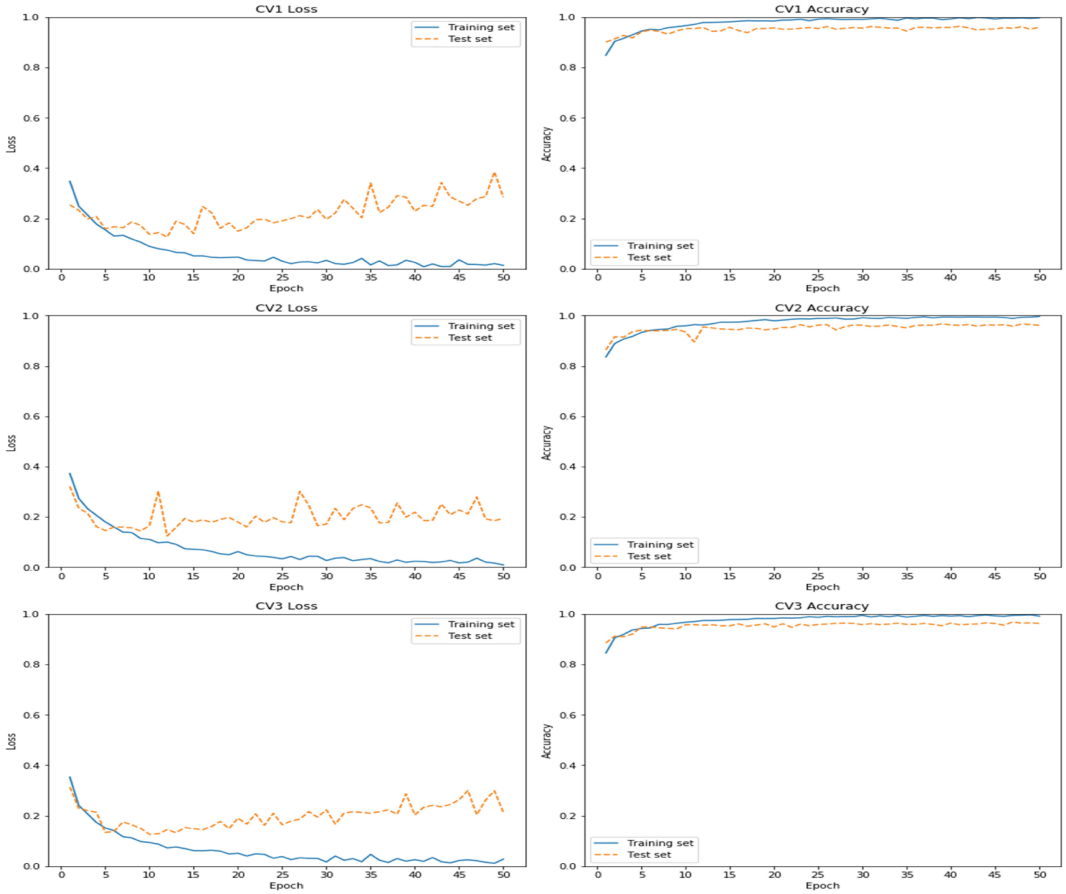

Fig. 6. CNN Loss and Accuracy Graphics for 3-Cross Validation

This study has been compared with previous comparative analysis of face mask detection. This study compared between KNN and SVN machine learning algorithms with CNN deep learning algorithm for face mask detection. Previous studies such as by Jagadeeswari et al. [11] and Nagrath et al. [14] compared only deep learning algorithm which are CNN with other architectures. This study also examines the execution time that can be useful for researchers as one of the consideration criteria to choose the algorithm. This was not included in the research by Jagadeeswari et al. [11] and Nagrath et al. [14].

The limitation of this study is that it uses only one CNN architecture. The recommendation for future research is to compare the performance of other CNN architectures such as Alex Net [20], GoogLeNet [21], VGGNet [22], ZFNet [23], and ResNet [24] with machine learning algorithm. In addition, the use of the image pre-processing stage can be useful to improve the performance of face mask detection.

VI. Conclusion

Based on the experiment result, it can be concluded that the KNN, SVM, and CNN are good algorithm for face mask detection using 5 cross-validations on 3,886 image data. The performance of KNN and SVM is good, each of which has an accuracy of 0.8115. and 0.8721. The execution time of the KNN and SVM training is also fast, which are 185.27 and 352.968 seconds, respectively. CNN has a better accuracy performance of 0.9683 but has a longer execution time of 2,507.802 seconds.

In the case of large amounts of image data, KNN and SVM can be used as temporary algorithms in image classification due to their faster execution times. At the same time, CNN can be trained to form a classification model.

When the CNN classification model has been formed, it is advisable to use CNN for the face mask detection because it has better performance than the KNN and SVM.

Author Contributions: *Mohammad Farid Naufal*: Conceptualization, Methodology, Writing - Original Draft, Supervision. *Selvia Ferdiana Kusuma*: Methodology, Writing - Review & Editing. *Zefanya Ardya Prayuska, Ang Alexander Yoshua, Yohanes Albert Lauwoto, Nicky Setyawan Dinata, David Sugiarto*: Software, Investigation, Data Curation

Funding: This research received no specific grant from any funding agency.

Conflicts of Interest: All the authors declare that there was no conflict of interest.

REFERENCES

[1] S. Zhang, X. Li, M. Zong, X. Zhu, and D. Cheng, "Learning k for kNN Classification," *ACM Trans. Intell. Syst. Technol.*, vol. 8, no. 3, 2017, doi: 10.1145/2990508.
[2] X. Wu *et al.*, *Top 10 algorithms in data mining*, vol. 14, no. 1. 2008.
[3] N. O'Mahony *et al.*, "Deep Learning vs. Traditional Computer Vision," *Adv. Intell. Syst. Comput.*, vol. 943, no. Cv, pp. 128–144, 2020, doi: 10.1007/978-3-030-17795-9_10.
[4] K. O'Shea and R. Nash, "An Introduction to Convolutional Neural Networks," pp. 1–11, 2015, [Online]. Available: http://arxiv.org/abs/1511.08458.
[5] A. A. M. Al-Saffar, H. Tao, and M. A. Talab, "Review of deep convolution neural network in image classification," *Proceeding - 2017 Int. Conf. Radar, Antenna, Microwave, Electron. Telecommun. ICRAMET 2017*, vol. 2018-Janua, pp. 26–31, 2017, doi: 10.1109/ICRAMET.2017.8253139.
[6] M. Pak and S. Kim, "A review of deep learning in image recognition," *Proc. 2017 4th Int. Conf. Comput. Appl. Inf. Process. Technol. CAIPT 2017*, vol. 2018-Janua, pp. 1–3, 2018, doi: 10.1109/CAIPT.2017.8320684.
[7] E. Setiawan and A. Muttaqin, "Implementation of K-Nearest Neighbors face recognition on low-power processor," *Telkomnika (Telecommunication Comput. Electron. Control.*, vol. 13, no. 3, pp. 949–954, 2015, doi: 10.12928/telkomnika.v13i3.713.
[8] P. M. Shah, "Face detection from images using support vector machine," 2012.
[9] M. T. Ghazal and K. Abdullah, "Face recognition based on curvelets, invariant moments features and SVM," *Telkomnika (Telecommunication Comput. Electron. Control.*, vol. 18, no. 2, pp. 733–739, 2020, doi: 10.12928/TELKOMNIKA.v18i2.14106.
[10] A. R. Syafeeza, M. Khalil-Hani, S. S. Liew, and R. Bakhteri, "Convolutional neural network for face recognition with pose and illumination variation," *Int. J. Eng. Technol.*, vol. 6, no. 1, pp. 44–57, 2014.
[11] C. Jagadeeswari and M. U. Theja, "Performance Evaluation of Intelligent Face Mask Detection System with various Deep Learning Classifiers Keywords :," *Int. J. Adv. Sci. Technol.*, vol. 29, no. 11, pp. 3074–3082, 2020.
[12] W. Han, Z. Huang, A. Kuerban, M. Yan, and H. Fu, "A Mask Detection Method for Shoppers under the Threat of COVID-19 Coronavirus," in *Proceedings - 2020 International Conference on Computer Vision, Image and Deep Learning, CVIDL 2020*, Jul. 2020, pp. 442–447, doi: 10.1109/CVIDL51233.2020.00-54.
[13] V. Vinitha and V. Velantina, "Covid-19 Facemask Detection With Deep Learning and Computer Vision," *Int. Res. J. Eng. Technol.*, vol. 7, no. 8, pp. 3127–3132, 2020.
[14] P. Nagrath, R. Jain, A. Madan, R. Arora, P. Kataria, and J. Hemanth, "SSDMNV2: A real time DNN-based face mask detection system using single shot multibox detector and MobileNetV2," *Sustain. Cities Soc.*, vol. 66, no. December 2020, 2021, doi: 10.1016/j.scs.2020.102692.
[15] S. Ge, J. Li, Q. Ye, and Z. Luo, "Detecting masked faces in the wild with LLE-CNNs," *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 426–434, 2017, doi: 10.1109/CVPR.2017.53.
[16] M. Grassi and M. Faundez-Zanuy, "Face recognition with facial mask application and neural networks," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 4507 LNCS, pp. 709–716, 2007, doi: 10.1007/978-3-540-73007-1_85.
[17] A. Shustanov and P. Yakimov, "CNN Design for Real-Time Traffic Sign Recognition," *Procedia Eng.*, vol. 201, pp. 718–725, 2017, doi: 10.1016/j.proeng.2017.09.594.
[18] Larxel, "Face Mask Detection | Kaggle," 2020. https://www.kaggle.com/omkargurav/face-mask-dataset (accessed Feb. 01, 2021).
[19] F. Chollet and & O., "Keras: the Python deep learning API," *Keras: the Python deep learning API*, 2020. https://keras.io/ (accessed Dec. 18, 2020).
[20] T. F. Gonzalez, "Handbook of approximation algorithms and metaheuristics," *Handb. Approx. Algorithms Metaheuristics*, pp. 1–1432, 2007, doi: 10.1201/9781420010749.
[21] C. Szegedy *et al.*, "Going deeper with convolutions," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 07-12-June, pp. 1–9, 2015, doi: 10.1109/CVPR.2015.7298594.
[22] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, pp. 1–14, 2015.
[23] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8689 LNCS, no. PART 1, pp. 818–833, 2014, doi: 10.1007/978-3-319-10590-1_53.
[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 770–778, 2016, doi: 10.1109/CVPR.2016.90.