Int. J. Nonlinear Anal. Appl. 14 (2023) 3, 121–140
ISSN: 2008-6822 (electronic)
http://dx.doi.org/10.22075/ijnaa.2022.27970.3770

A proposed 3-stage CNN classification model based on augmentation and denoising

Mohanad Azeez Joodi*, Muna Hadi Saleh, Dheyaa Jasim Kadhim

Department of Electrical Engineering, College of Engineering, University of Baghdad, Baghdad, Iraq

(Communicated by Saman Babaie-Kafaki)

Abstract

This work proposed a CNN classification model that aims to classify the faces by three stages applied to a real data set. The first stage shows the effects of the augmentation technique on the real data set where these effects include online, offline, and without augmentation. At this stage, the proposed CNN model is a built-from-scratch that has low computational complexity, low layers and the smallest filter sizes. The second stage involved denoising the images in the real data set, where the images are preprocessed by applying the median, Gaussian, and mean filters to render the images more smooth and compare the effects of these filters based on the classification accuracy. The third stage involved a multi-class proposed model that contained 12 classes of images that were trained on the applied real data set, in addition to a benchmark set of images that was collected from the Internet. The findings reveal that the model accuracy reached 98.81% when the offline augmentation model or the median filter was applied to the real data set, while it reached 97.48% when the CNN multi-class proposed model was applied to identify the non-permission class. These processes were found to improve the performance parameters such as precision, recall, F1 score, and area under the curve (AUC). Finally, to enhance the test prediction accuracy and test time, pre-training and fine-tuning (transfer learning) are applied on the real data set so as test accuracy and test time of our proposed model are better as compared with other models reached 99.7% and 4 seconds respectively.

Keywords: Face recognition, Deep learning CNN model, online-offline augmentation, Median filter, Multi class face Identification, Fine tuning
2020 MSC: 68T07, 68T05

1 Introduction

Face recognition systems have become one of biometry's most fascinating study subjects. The direction and reflection of light, as well as emotional and physical changes in facial expression are the primary obstacles to face recognition. Training the system with readily accessible data in tiny data sets is a significant issue that has a detrimental impact on its performance. A convolutional neural network (CNN) is a deep learning architecture able to process vast volumes of training data. Various filters can be used to boost the quantity of images of employees in a small business, for instance [15, 10]. CNN requires a massive quantity of training data. Data augmentation (DA) is an effective method for overcoming this issue. In a prior study, the pupil centre point detection performance

*Corresponding author
Email addresses: m.mohammad1702d@coeng.uobaghdad.edu.iq (Mohanad Azeez Joodi), dr.muna.h@coeng.uobaghdad.edu.iq (Muna Hadi Saleh), dheyaa@coeng.uobaghdad.edu.iq (Dheyaa Jasim Kadhim)

Received: July 2022 *Accepted:* November 2022

of three DA approaches, namely the affine transformation-based approach, the synthetic image-based approach, and the generative adversarial network (GAN)-based approach, was investigated. Utilizing a commercially available CNN model, the detection accuracy was assessed. As seen by the results, applying affine alteration to an eye image improved its detection accuracy [12]. In another work, CNN models were tested under various noise levels. The results revealed that using data augmentation improves noise performance by more than 55 percent [17]. A Glow data augmentation technique was used to diversify a facial photo dataset in order to test its effects on a CNN-based face classification and identification system. In the initial phase, CNN was trained using the publicly accessible Labeled Faces in the Wild (LFW) database, and the suggested method's accuracy was tested at 92.2%. In the second step, the CNN network was trained after the LFW dataset was varied using the Glow method. Glow data augmentation increased the accuracy of the recommended network to 93.6% [18] [9]. In the context of limited training data, the influence of multiple data augmentation strategies on the automatic categorization of celiac disease using endoscopic imaging was studied. Cropping and flipping are common image improvement techniques that exploit the entire gamut of affine and even projective transformations. The scale changes caused by the augmentation are taken into account by a new way to change the lighting [27]. The experimental data computation set and network model are a generative adversarial network (GAN) with a few national elements. The obtained data is preprocessed and then normalized in order to create the basic sample data set. Secondly, the model obtains a feature vector by placing an image in the CNN layer. The trained CGAN model is then utilized to complete the data augmentation [25][8]. To improve the accuracy of face recognition systems, a new face recognition method based on improved convolutional neural networks was demonstrated. The improved algorithm worked when it was used on the data set [24][11]. Impulse noise is one of the most common types of noise that can mess up a mammogram. The duo-median filter should get rid of impulse noise with a high density. Using peak signal-to-noise ratio and structural similarities, the results of the proposed algorithm are compared to those of the median filter, the adaptive median filter, and the super mean filter for general pictures and mammograms. The proposed duo-median does a much better job of increasing noise percentages than AMF and SUMF [22][1]. Gaussian noise can be taken out of digital photos using the Discrete Wavelet Transform and the Thresholding methods. The goal of this technology is to make digital photos look better. When this method is used on noisy photos, the results are compared. There are comparisons of different filters [16]. The median filter, the Gaussian filter, the mean filter, and the Wiener filter are all used to filter a picture. To figure out which image filter works best, different noises, like Gaussian and salt and pepper, are added in different amounts to the image [4]. Innovative software architectures and hardware implementations are provided to enhance grayscale photographs that have been severely degraded by impulse noise. The main goal is to get rid of salt and pepper impulse noise using an enhanced median filter. This filter uses spatial information processing to figure out which pixels in a picture are affected by impulse noise and then replaces them with the median value of a 2-D area with a low variance value [2].

The work procedure of this research is initiated by creating real data sets from our cameras, which detect the face region by using the Haar cascade detector and input them to the proposed CNN classification model to train it and classify it into two classes: permission and non-permission. The first stage is to increase the validation accuracy and other performance parameters by applying the augmentation technique (online or offline) on the real data set to show the effect of it. The second stage involves applying the preprocessing (denoising) by choosing three types of filters and showing the effect of each one on the classification performance, such as validation accuracy and other performance. The third stage involved a multi-class proposed model that contained 12 classes of images that were trained on the applied real data set, in addition to a benchmark set of images that was collected from the Internet to identify the classes from the first stage.

The main contribution of this work is proposing an CNN model classification built from scratch low complexity , low layers , smallest filters sizes and low computational cost to train and test the real data set that are created from our cameras system and then applying augmentations and denoising algorithms to enhancement the classification accuracy and other metrics parameters. In addition to enhance the test prediction accuracy and time for the proposed model when compare it with other models as a fine tuning (transfer learning).

The organization of this research as follows: Section 1 describes the research background about face recognition systems and some literature survey about face recognition techniques that adopted upgraded convolutional neural networks. Section 2 presents the proposed model architecture of the two classes: permission and non-permission classification. Section 3 demonstrates the experimental results and discussion consisting of seven subsections: training the real data set; classification performance; test predictions system as a whole; enhanced images by applying denoising (preprocessing); fine-tuning and pre-training other models by applying them to the real data set; multi-class identification. Section 4 discusses the main contributions and conclusions that come with this work.

2 Proposed Classification Model

The proposed classification model consists of three stages applied to real data set collected from fixed cameras. The first stage involves building the proposed low-complexity CNN model of face recognition and classification from scratch; it consists of two classes: ”permitted person” and ”non-permitted person.” This stage aims to distinguish which person of those attempting to enter some high-security buildings, such as airports or tourist destinations, has permission to enter. In this stage, the proposed algorithm can be divided into three models: the first one is to create the real data set without augmentation effect, the second is with on-line augmentation, and the third is with off-line augmentation. The second stage involves testing the effects of denoising filters (preprocessing), such as median, Gaussian and mean filters, applied to the images before their input to the CNN proposed model. The third stage involves a multi-class proposed model that contains 12 classes which is trained on the applied real data set plus a benchmark data set taken from the internet. At the classification model first stage, the proposed model was applied to a real data set without data augmentation, which involved the creation of real data sets collected from cameras. As a test; take captures images from distributed fixed cameras located in the important regions of the high-security buildings to capture images of people entering and then classifying them as ”permission” or ”non-permission” based on a built-from-scratch proposed CNN training model for face recognition. images had a window size of 140×140 pixels to recognize the regions of a person’s face. The proposed model and architecture, as shown in Tables 1 and 2, respectively, consist of the convolutional, batch normalization, activation function, max pooling, and fully connected layers.

Table 1: The proposed CNN model.

Layer (Type)	**Output (shape)**	**Parameters**
conv2d (Conv2D)	(None, 138, 138, 16)	448
conv2d_1 (Conv2D)	(None, 134, 134, 32)	12832
conv2d_2 (Conv2D)	(None, 132, 132, 32)	9248
batch normalization	Batch No (None, 132, 132, 32)	128
activation (Activation)	(None, 132, 132, 32)	0
max_pooling2d (MaxPooling2D)	(None, 66, 66, 32)	0
conv2d_3 (Conv2D)	(None, 62, 62, 64)	51264
batch_normalization_1	(None, 62, 62, 64)	256
activation_1 (Activation)	(None, 62, 62, 64)	0
max_pooling2d_1 (MaxPooling2)	(None, 31, 31, 64)	0
conv2d_4 (Conv2D)	(None, 29, 29, 32)	18464
batch_normalization_2 (Batch)	(None, 29, 29, 32)	128
activation_2 (Activation)	(None, 29, 29, 32)	0
max_pooling2d_2 (MaxPooling2)	(None, 14, 14, 32)	0
flatten (Flatten)	(None, 6272)	0
dense (Dense)	(None, 300)	1881900
activation_3 (Activation)	(None, 300)	0
dropout (Dropout)	(None, 300)	0
dense_1 (Dense)	(None, 1)	301
activation_4 (Activation)	(None, 1)	0

The process of convolution is defined as the convolving of two primary functions (f) and (g), where the output is $(f * g)(n)$. The expression for a one-dimensional convolution is defined by [26]:

$$(f * g)(n) = \sum_m f(m)g(n-m). \tag{2.1}$$

In two-dimensional (2D) digital images, where (A) is a 2D image with $(i*j)$ dimensions, (K) is a filter with $(m*n)$ dimensions, and (F) is the feature map, convolution can be utilized. The output F is produced by convolving the picture A with the filter K. Because the operation is commutative in this circumstance, it is described by Equation 2.2, and the 2D equation can be expressed as Equation 2.3 below [26]:

$$F(i,j) = (A * K)(i,j) = \sum_m \sum_n A(m,n)K(i-m, j-n) \tag{2.2}$$

Table 2: Architecture of the layers of the proposed CNN model.

Layer name	**Kernels size, Stride**	**Output image shape**
Input image	-	(138,138,3)
Conv 1	16	(138,138)
Conv 2	32	(134,134)
Conv 3	32	(132,132)
Batch normalization	32	(132,132)
Relu	-	(132,132)
Max.pooling 1	2*2,2	(66,66)
Conv 4	64	(62,62)
Batch normalization 1	64	(62,62)
Relu	-	(62,62)
Max.pooling 2	2*2,2	(31,31)
Conv 5	32	(29,29)
Batch normalization 2	32	(29,29)
Relu	-	(29,29)
Max pooling 3	2*2,2	(14,14)
Flatten	-	6272 vectors
Dense	-	300 vectors
Relu	-	300 units
Dropout	0.4	300 units
Dens1	-	301 units
Sigmoid	-	1 units

$$F(i,j) = (A * K)(i,j) = \sum_m \sum_n A(i-m, j-n)K(m,n). \tag{2.3}$$

The RELU activation function is used to ignore negative values. Also, the adaptive moment estimation (Adam) is an optimization approach that can be used to update network parameters based on training data instead of the traditional stochastic gradient descent (SGD) procedure [13]. Adam has been empirically demonstrated to be faster than other algorithms to achieve convergence. Dropout is a popular generalization strategy. Neurons are released at random during each training phase. The large-scale network is used instead. Predictions are difficult to make during the testing procedure. The prediction process is set up at a first layer of 140×140×16 pixels, and the output is a 14×14×32-pixel feature map. Binary Cross-Entropy has been used as a loss function; it gives a measure of how far apart two values are in the range from [0] to [1]. The loss function can be used to define binary cross-entropy [7].

$$\text{loss}(\alpha, \acute{\alpha}) = -(a\log(\acute{\alpha}) + (1-a)\log(1-\acute{\alpha})). \tag{2.4}$$

where $\acute{\alpha}$ is the value returned by the model and α is the true label value. It is common to minimize $(\alpha, \acute{\alpha})$ for multiple images at the same time. In a feed-forward neural network, normalization is the process of altering data to have a mean of zero and a standard deviation of one. The mean of this hidden activation (μ) is determined as in Equation 2.5 in this phase, which has a batch input from layer h.

$$\mu = \frac{1}{m} = \sum h_i \tag{2.5}$$

where m represents the number of neurons in layer h. The standard deviation of the hidden activations (σ) is then computed using the following equation [6]:

$$\sigma = \left[\frac{1}{m}\sum (h_i - m)^2\right]^{\frac{1}{2}}. \tag{2.6}$$

The average and standard deviations have also been calculated. The concealed activations are normalized using these variables. The mean is subtracted from each input and the result is multiplied by the standard deviation plus the smoothing parameter (ε). The smoothing term maintains numerical stability inside the operation by finishing a division by a zero value, as:

$$h_{i(norm)} = \frac{h_i - \mu}{\sigma + \varepsilon}. \tag{2.7}$$

The final step involves rescaling and offsetting the data. Two components of the batch normalization (BN) procedure which come into play here are Y (gamma) and β (beta). In Equation 2.8 below, these parameters are used to re-scale (y) and shift the vector storing values from previous operations.

$$h_{i=Yh_{i(norm)+\beta}}. \tag{2.8}$$

These two parameters may be learned, and the neural network guarantees that the best Y and β values are picked during the training process. This will allow for the correct normalization of each batch. Batch Norm resembles convolutional neural networks in its operation. Even though it can be conducted in the same manner as before, the convolutional property has persisted. Convolutions share filters that run along the feature maps of the input. The feature map of a picture is typically the height and width. These filters are consistent across all feature maps. Similarly, the result can be normalized and distributed among the feature maps. In a typical batch norm, each trait would have its own mean and standard deviation. Table 3 shows the batch normalization settings chosen.

Table 3: Parameter chosen in the batch normalization layer

Axis	1
Momentum	0.99
Epsilon	0.001
Center	True
Scale	True
Beta initialization	Zeros
Gamma initialization	Ones
Moving mean initialization	Zeros
Moving variance initialization	Ones

The fully connected layer and the weights of the CNN proposed model are shown in Fig 1.

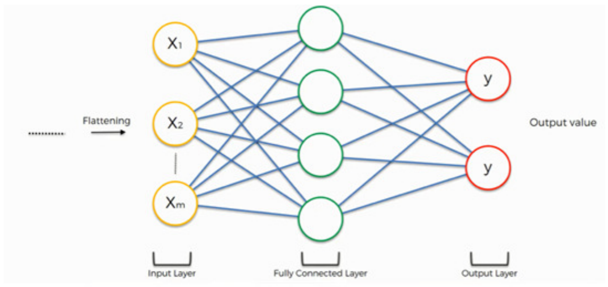


Figure 1: The fully connected layer on the CNN proposed model

In the second and third models of this first stage, two different types of data augmentation were applied. The second model involved online augmentation through the training process of the CNN model; its architecture is the same as that in Tables 2 and 3 shown earlier. In the third model, offline data augmentation was applied before the training and the result was used as input to the CNN model training to achieve the classification, as demonstrated in Fig 2.

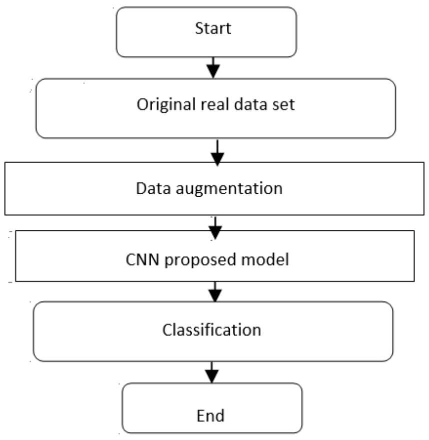


Figure 2: The first stage of the proposed model; the third model involving images with offline data augmentation.

The architecture of real data sets with the offline argumentation model is shown in Table 4. The size of image was resized to $100 * 100$ pixels before entering the model.

Table 4: The architecture of proposed CNN model with offline data augmentation.

Layer (Type)	**Output (shape)**	**Parameters**
conv2d (Conv2D)	(None, 98, 98, 16)	448
conv2d_1 (Conv2D)	(None, 94, 94, 32)	12832
conv2d_2 (Conv2D)	(None, 92, 92, 32)	9248
batch normalization	Batch No (None, 92, 92, 32)	128
activation (Activation)	(None, 92, 92, 32)	0
max_pooling2d (MaxPooling2D)	(None, 46, 46, 32)	0
conv2d_3 (Conv2D)	(None, 42, 42, 64)	51264
batch_normalization_1	(None, 42, 42, 64)	256
activation_1 (Activation)	(None, 42, 42, 64)	0
max_pooling2d_1 (MaxPooling2)	(None, 21, 21, 64)	0
conv2d_4 (Conv2D)	(None, 19, 19, 32)	18464
batch_normalization_2 (Batch)	(None, 19, 19, 32)	128
activation_2 (Activation)	(None, 19, 19, 32)	0
max_pooling2d_2 (MaxPooling2)	(None, 9, 9, 32)	0
flatten (Flatten)	(None, 2592)	0
dense (Dense)	(None, 300)	777900
activation_3 (Activation)	(None, 300)	0
dropout (Dropout)	(None, 300)	0
dense_1 (Dense)	(None, 1)	301
activation_4 (Activation)	(None, 1)	0

3 Experimental Results and Discussion

The third part consists of seventh sections related to conducting the proposed CNN model training; the first section is to prepare the real data set that contains images of people captured from cameras located in the buildings. This section therefore includes two classes; the permission people and the non-permission people. The second section involves the training of the real data set without augmentation, with online augmentation, and with offline augmentation applied on the real data set. The third section involves the use of some performance measures to determine the model's quality and accuracy. The fourth section involves performing a test prediction of some images to classify them into two classes; permission and non-permission people. The fifth section involves testing the effects of denoising (preprocessing). It includes applying the median filter, Gaussian filter, and the mean filter to the images on the real data set and preparing them to be used as a training input to the proposed CNN model. It also involves determining some performance measures of model's quality and accuracy. The sixth section uses different other models applied on the real data set as transfer learning and calculate performance parameters and also make test prediction to the all different models with proposed model. The seventh section uses a multi-class classification CNN model by preparing a new data set, consisting of the real data set plus the benchmark data set which is taken from the internet. In this section, the proposed model is trained by 12 classes. In addition, the non-permission class taken from the second section is tested after using it as an input to the multi-class model in order to achieve facial identification. Some performance measures were applied to determine the model's quality and the accuracy.

3.1 Dataset

For the experiment, a real data set was used, it consists of 347 frontal faces with a minimum size of 140×140 pixels.The dataset is divided into 2 parts for training and validation sets with 263,84 images, respectively ; it consists of two classes : permission and not permission classes. Figures 3 -4 show example images in the Real dataset

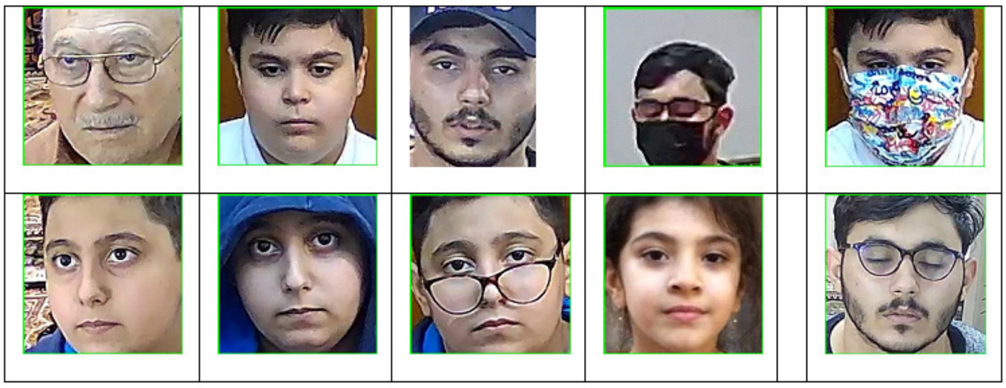


Figure 3: Real data set example images representing the permission class

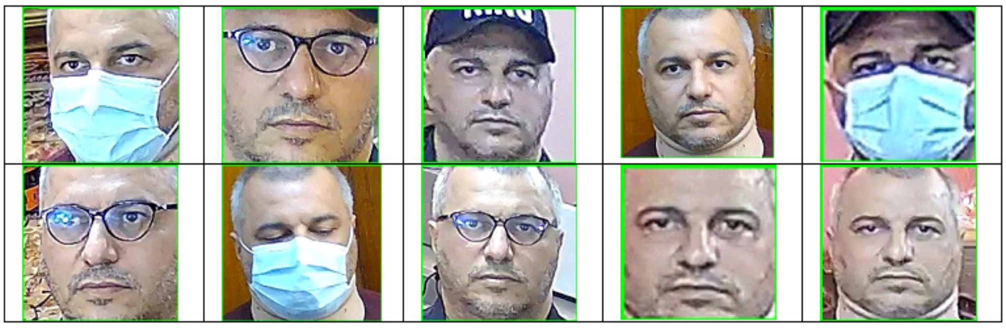
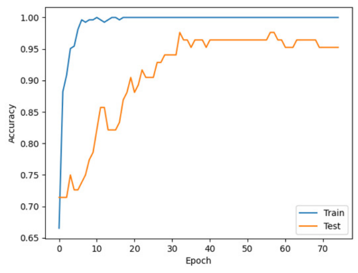


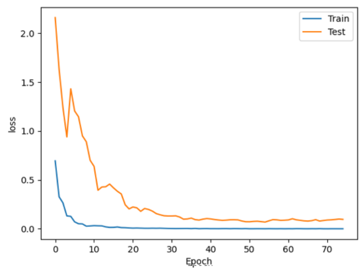
Figure 4: Real data set example images representing the non- permission class.

3.2 Training

The model included three stages of the training procedure. The first stage used 0.0001 learning rate applied to the real data set without augmentation. The entire performance parameters were calculated. The second stage is training on the real data set with online augmentation. The same method was used to test the training on the real data set. The third stage includes the real data set training with offline augmentation, which was also applied as a test on the real data set. The entire performance parameters were also calculated. The binary cross-entropy loss between the training and validation sets was measured, using a batch size of 32. The training process took place by using 75 Epoch, while a features vector location of 300 values was selected. Figure 5a illustrates a comparison between the accuracy results of the training and validation data without augmentation. Figure 5b shows a comparison between the binary cross-entropy loss values of the training and validation data without augmentation. Figures 6a and 6b illustrate the differences in the values of validation accuracy and loss between the training and validation data with online augmentation, whereas Figures7a and 7b illustrate the same differences with the offline augmentation.

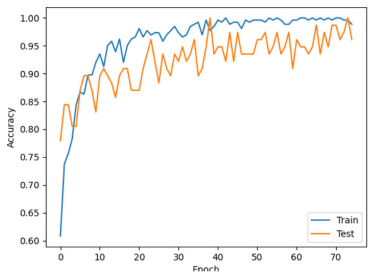


(a)

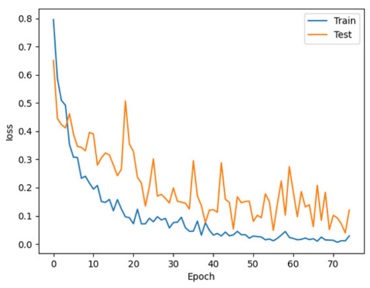


(b)

Figure 5: The accuracy (a) and the loss (b) values of real data set without augmentation.

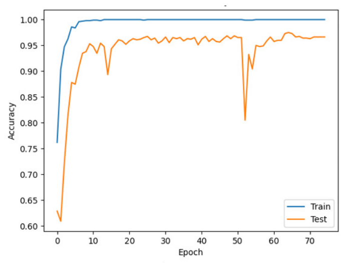


(a)

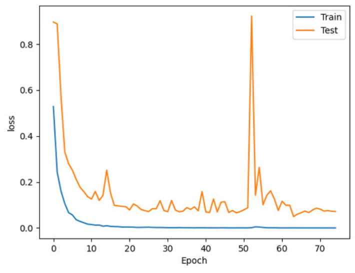


(b)

Figure 6: The accuracy (a) and loss (b) values of real data set with online augmentation.



(a)



(b)

Figure 7: The accuracy (a) and loss (b) values of real data set with offline augmentation.

3.3 Performance

To assess the performance of the proposed model, five metrics were used: precision or specificity (a measure of the fraction of negative class that is correctly classified), recall (a measure of the fraction of positive class that is correctly classified), validation accuracy (the number of correct predictions divided by the total number of instances evaluated), and training error (the number of incorrect predictions divided by the total number of instances evaluated), and FPR (false positive rate), which were calculated as expressed in Equations 3.1, 3.2, 3.3, 3.4, 3.5, and 3.6, respectively [3]. The precision and AUC (area under the curve) metrics indicate the model's accuracy and quality, as determined by the classification results. The retrieval metric, which presents the ability to find all of the relevant objects in the original real dataset, for data without augmentation, with online augmentation, and with offline augmentation is given in

Table 5. The applied feature vector selects 300 values.

$$\text{Precision} = \frac{TP}{TP+FP} \tag{3.1}$$

$$\text{TPR (True positive rate)=Recall} = \frac{TP}{TP+FN} \tag{3.2}$$

$$\text{Validation Acc} = \frac{TP+TN}{TP+FP+TN+FN} \tag{3.3}$$

$$\text{Training error} = \frac{FP+FN}{TP+FP+TN+FN} \tag{3.4}$$

$$\text{Specificity} = \frac{TN}{TN+FP} \tag{3.5}$$

$$\text{FPR (False positive rate)} = 1 - \text{Specificity} = \frac{FP}{TN+FP} \tag{3.6}$$

Where True Positive (TP): the class is positive and the model prediction also positive. True Negative (TN): the class is negative and the model prediction also negative. False Positive (FP): the model was misclassified in the negative class. False Negative (FN): the model was misclassified in the positive class. Where real positive class $(P) = TP+FN$, real negative class $(N) = FP + TN$

Table 5: Confusion matrix for the models without data augmentation model, with online augmentation, and with offline augmentation, all applied on the original real data set.

Models proposed applied on real data set	True positive	True negative	False positive	False negative	Precision	Recall (TPR)	FPR	Auc (area under the curve)	Validation accuracy	Loss	Mean square error
Without augmentation	21	59	1	3	0.9545	0.8750	0.0166	0.9958	95.24 %	0.0914	0.0338
With online augmentation	22	60	0	2	1	0.9167	0	0.9993	97.62 %	0.0433	0.0141
With offline augmentation	23	60	0	1	1	0.9583	0	1	98.81 %	0.0176	0.0064

Table 5 shows the validation accuracy value which was improved to 98.81% when using the offline data augmentation model. The validation loss was also improved to reach 0.0176, whereas the AUC value was equal to 1.00. Therefore, the offline data augmentation model was found to be better than the model with online augmentation and the model without augmentation when applied on real data set.

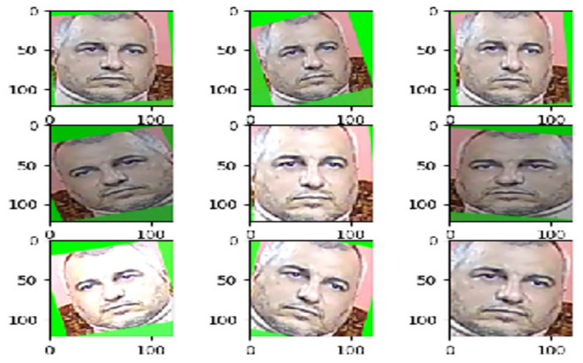


Figure 8: The offline data augmentation of one image

The geometric transformations of a single image, including rotation range, horizontal flipping, brightness range, and zooming, were selected to generate more data points. The offline augmentation parameters that were chosen are listed in Table 6 of the Keras Python Code for Deep Learning; they are the default parameters because the images generated by offline augmentation did not distort with these values, and it was obvious to use them in the CNN classification proposed model training.

Table 6: The geometric transformation parameters of a single image with offline augmentation

Function	**Parameter values**
Rotation range	20 degree (randomly rotate images 20-90 degree)
Horizontal flip	True
Brightness range	0.5-1.5 decrease/increase in range(0 dark-2 light)
Zoom range	(0-0.2) 20% zooming

The geometric transformations of a single image in the online augmentation are shown in Table 7. The same are shown in the offline augmentation except for the feature-wise centre and feature-wise standard normalization, which means the mean and standard deviation of the features. The feature normalization is used to normalize the features from (0 to 1) to make them more reliable for the computation process because the process deals with normalized parameters instead of pixels from (0 to 255).

Table 7: The geometric transformation parameters of a single image with online augmentation

Function	**Parameter values**
Feature wise center	True
Feature wise standard normalization	True
Rotation range	20 degree (randomly rotate images 20-90 degree)
Brightness range	0.5-1.5 decrease/increase in range(0 dark-2 light)
Zoom range	(0-0.2) 20% zooming

The confusion matrix in the offline augmentation applied on augmented real data set training and validation are 987,901 images respectively shown in the Table 8.
The validation accuracy is higher when it taken the augmentation model after training but applied on the real data set as a test instead of augmented data set, and all other metric parameters are high also.

Table 8: The confusion matrix of offline augmentation applied on augmented real data set.

Models proposed offline augmentation	True positive	True negative	False positive	False negative	Precision	Recall (TPR)	FPR	Auc (area under the curve)	Validation accuracy	Loss	Mean square error
on augmented real data set	377	510	0	14	1	0.9642	0.0166	0.9997	98.45 %	0.0338	0.0101

Figures 9 and 10 show charts representing the accuracy and loss valued when the three models were applied on the real data set; i.e. without augmentation, with online augmentation, and with offline augmentation. It was found that offline augmentation gives the best results, since it achieved accuracy of 98.81%, loss of 0.0176, and mean square error of 0.0064 in the epoch 65.

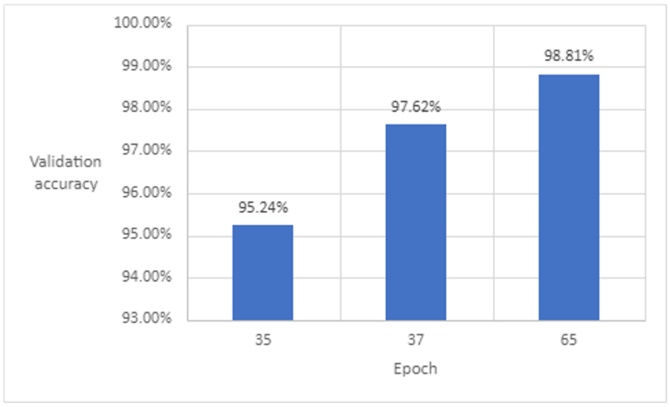


Figure 9: The validation accuracy values of the three models of without augmentation, with online, and with offline augmentation versus Epoch.

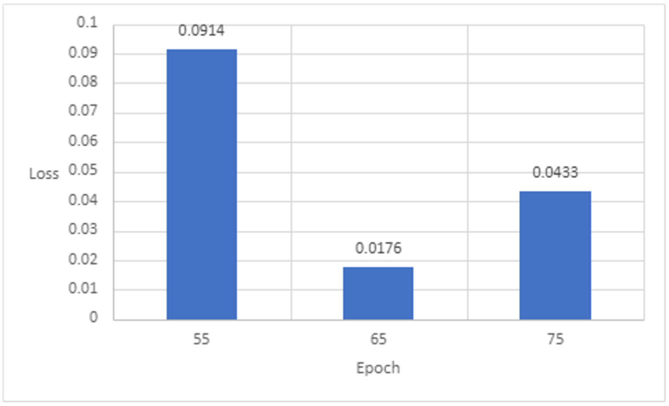


Figure 10: The values of loss of the three models of without, with online, and with offline augmentation with Epoch.

3.4 Test prediction system

The process of test prediction included several steps; the first step involved capturing the images from cameras and inputting them to the system; the second step involved the detection of the face image by the haar cascade detector and cutting the region of the face that has a size of 100x100 pixels; the third step involves inputting the resulting images to the proposed CNN model to classify them into the two classes of permission class and non-permission class. The running time per test was very short, including five seconds to predicate, four seconds to detect, and one second to classify all the eight images processed by using the proposed CNN model. Fig 11 shows a block diagram of the whole prediction system and Fig 12 shows the output test images resulting from the system after classification.

3.5 Enhanced images by applied denoising (preprocessing)

Real data set images were preprocessed with the median filter, the Guaisan filter, and the mean filter (3x3 matrices for each filter) to make them smoother before being fed into the proposed CNN training model. In terms of validation accuracy and loss values, the median filter demonstrates superior performance to the Guaisan and mean filters. Therefore, the classification model enhanced the quality and accuracy.

3.5.1 Median filter

Median filtering is a nonlinear technique for eliminating image noise. This filter is often used because it is effective at reducing noise while preserving edges. It is very effective at eliminating "salt and pepper" sounds. As the median filter traverses the image pixel by pixel, it replaces each pixel's value with the median value of neighboring pixels. The "window" is the pattern of neighbors that moves pixel by pixel across the entire image. The median is found

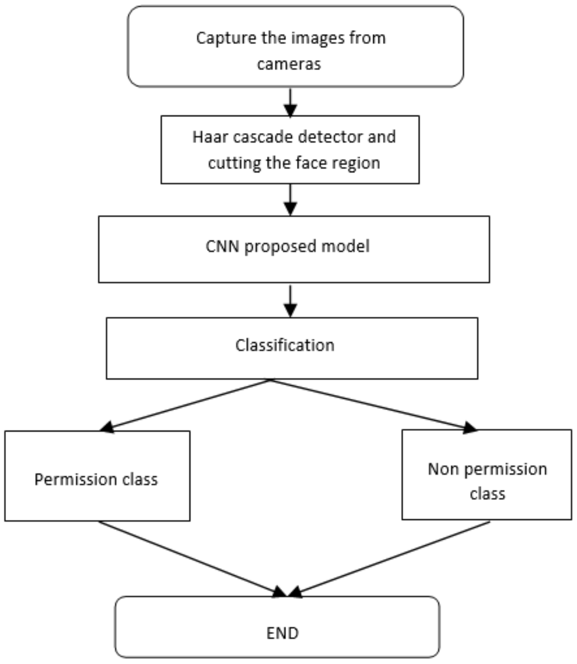

Capture the images from cameras
Haar cascade detector and cutting the face region
CNN proposed model
Classification
Permission class
Non permission class
END

Figure 11: A block diagram of the prediction system.

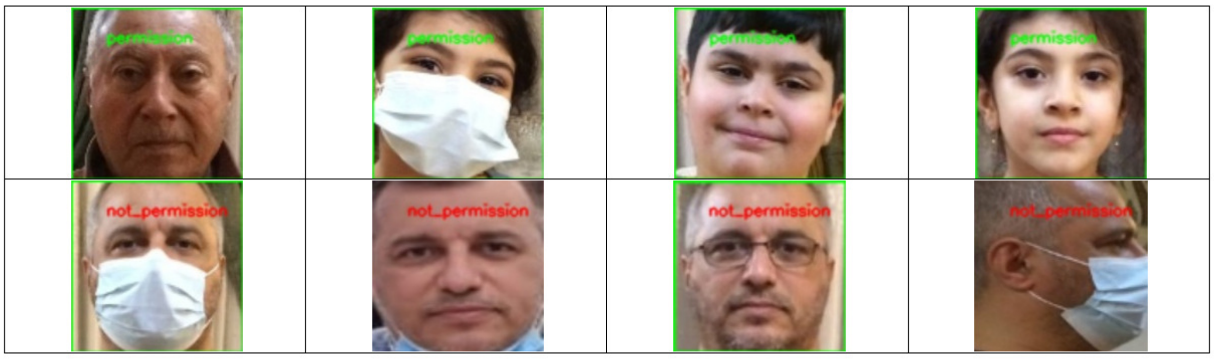


Figure 12: The results of the permission and non-permission classes test.

by putting the pixel values in the window in numerical order, then replacing the pixel in question with the middle (median) pixel value [23].

3.5.2 Gaussian filter

Gaussian filtering is utilized to blur images and eliminate noise and fine detail. The Gaussian function in one dimension is derived using Equation 3.7 [23].

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-x^2}{2\sigma^2}} \tag{3.7}$$

Where σ is the standard deviation of the distribution. The distribution is assumed to have a mean of 0.

3.5.3 Mean filter

Average (or mean) filtering is a method for ”smoothing” images by reducing the intensity variation between neighboring pixels. The average filter goes through the image pixel by pixel and replaces each value with the average value of the pixels around it, including its own.

Fig 13 depicts the effect of employing the median filter. After applying this filter, the real data set images became clearer because of their enhancement, while applying the Gaussian filter Fig 14 and mean filter Fig 15 caused no enhancement because these filters can have negative effects on the image edge. When the filter neighborhood straddles an edge, the filter will interpolate new values for pixels on the edge and thus will blur that edge. This may be a problem if sharp edges are required in the output. The median, mean, and Gaussian filters taken 3x3 matrix applied on the real data set images.

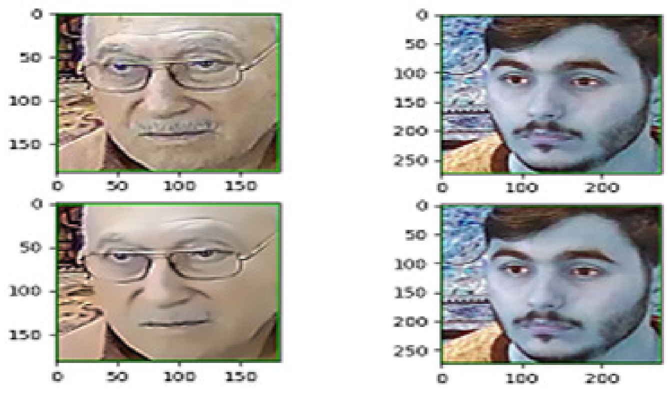


Figure 13: The effect of median filter on faces images of real data set.

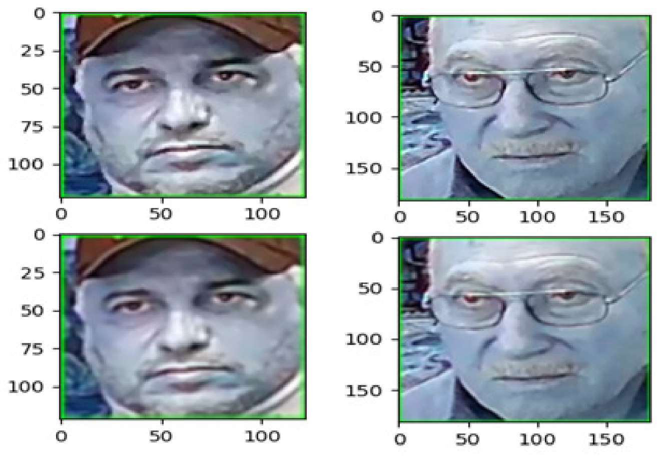


Figure 14: The effect of Gaussian filter on faces images of real data set.

The results of the performance parameters show that the model of preprocessing with median filter applied to a real data set before training improved validation accuracy to 98.81 percent, loss to 0.0668 percent, and mean square error to 0.0229 percent. In addition, all the other performance parameters were improved when compared with the results without denoising (i.e. without preprocessing), with mean filter, and with Gaussian filter, as shown in Table 9
.

Figures 16 and 17 show charts of validation accuracy and loss values with Epoch. The results following the use of the median filter were found to be better than those after the use of Gaussian filter, mean filter, and without filter when applied on real data set.

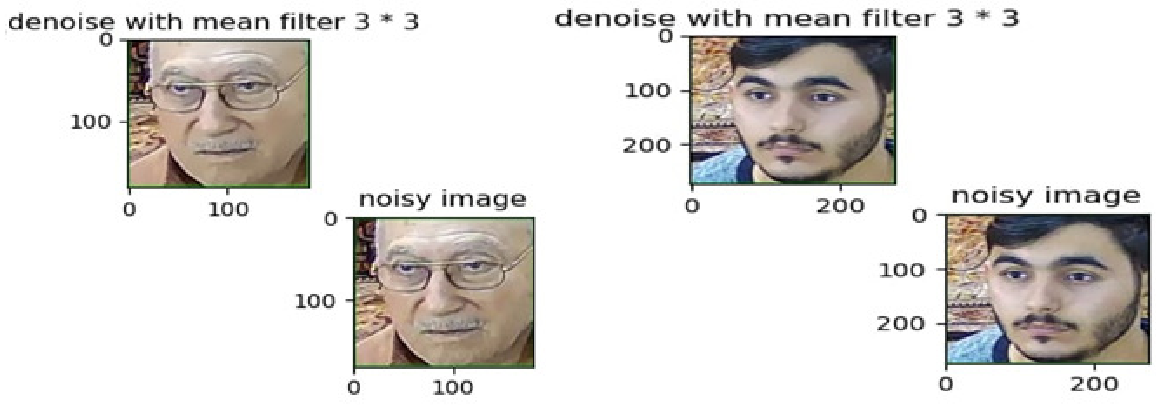

denoise with mean filter 3 * 3
noisy image
denoise with mean filter 3 * 3
noisy image

Figure 15: The effect of mean filter on faces images of real data set.

Table 9: A comparison of the performance between the models without noise, with Gaussian filter, with mean filter, and with median filter.

Models proposed of real data set	True positive	True negative	False positive	False negative	Precision	Recall (TPR)	FPR	Auc (area under the curve)	Validation accuracy	Loss	Mean square error
Without noise filter	21	59	1	3	0.9545	0.8750	0.0166	0.9958	95.24 %	0.0914	0.0338
With Gussian filter	21	59	1	3	0.9545	0.8750	0.0166	0.9965	95.24 %	0.0820	0.0265
With mean filter	21	59	1	3	0.9545	0.8750	0.0166	0.9938	95.24 %	0.0988	0.0340
With median filter	23	60	0	1	1	0.9583	0	0.9972	98.81 %	0.0668	0.0229

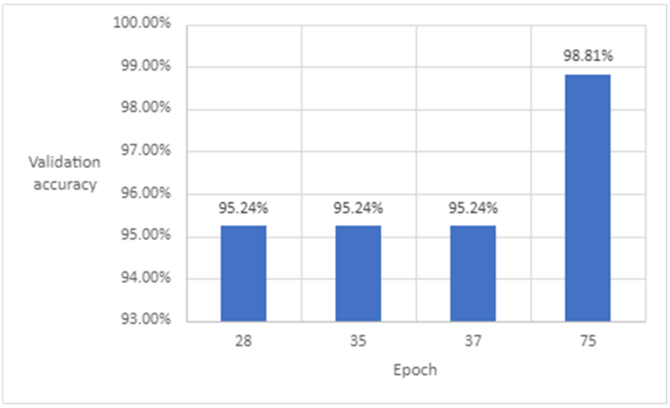


Figure 16: The validation accuracy values with median filter, with Gaussian filter, with mean filter and without filter with respect to Epoch.

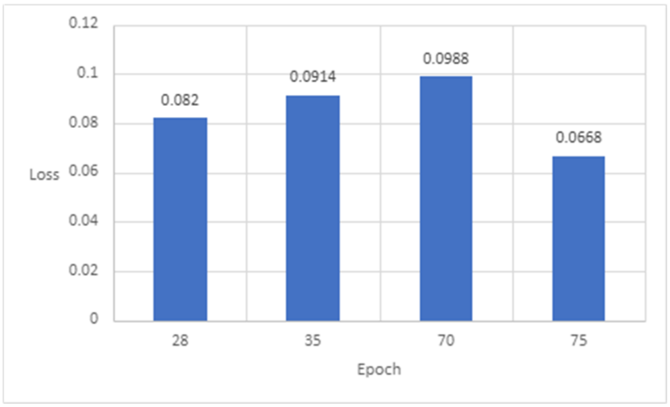
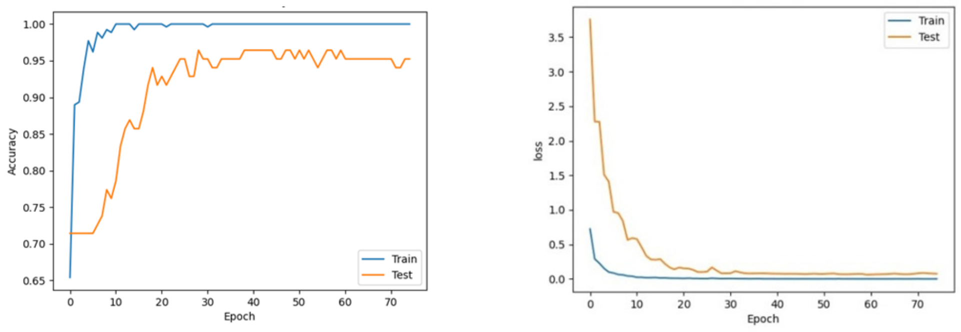


Figure 17: The values of loss with median filter, with Gaussian filter, with mean filter, and without filter with respect to Epoch.

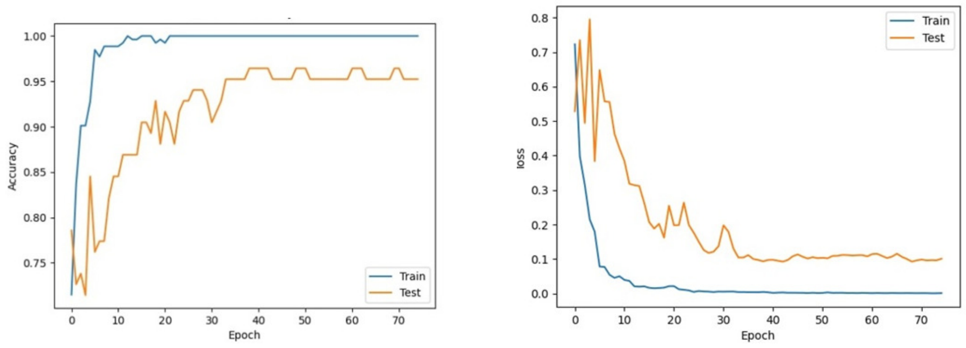
Figure 18 (a and b) shows the results of the values of validation accuracy and loss following the application of the Gaussian filter on real data set images before inputting them to the proposed model for the purpose of training the denoised data.



(a) (b)

Figure 18: The validation accuracy (a) and loss (b) after applying the Gaussian filter on real data set.

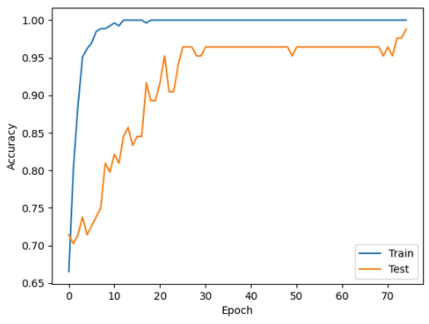
Figure 19 (a and b) shows the values of the validation accuracy and loss after applying the mean filter on real data set images before inputting them to the proposed model for the purpose of training the denoised data.



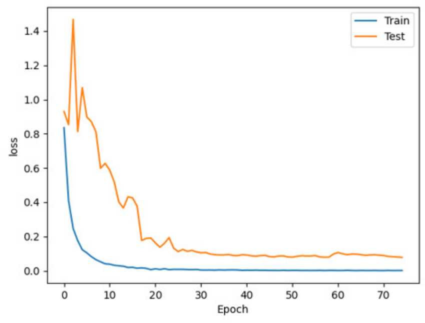
(a) (b)

Figure 19: The validation accuracy (a) and loss (b) values after the application of mean filter on real data set

Figure 20 (a and b) shows the validation accuracy and loss values after applying the median filter on real data set images before inputting them to the proposed model for the purpose of training the denoised data.



(a)



(b)

Figure 20: The validation accuracy (a) and loss (b) values after the application of median filter on real data set

3.6 Fine tuning and pre- training other models applied on the real data set

A real data set was used, it consists of 347 frontal faces with a minimum size of 140×140 pixels.The dataset is divided into 2 parts for training and validation sets with 263, 84 images, respectively; applied it to the different models pre-training and fine tuning as (transfer learning) such as Alex net[14], VGG16[20], VGG19[20], Resnet50 [5], and Google Net inspection V3 [21] .Table 10 shows the performance parameters of the all different models.

Table 10: Confusion matrix for the different models all applied on the original real data set.

Other Models	True positive	True negative	False positive	False negative	Precision	Recall (TPR)	Validation accuracy	Mean square error
Alex net	49	12	0	23	1	0.6805	72.619 %	0.2738
VGG16	49	31	0	4	1	0.9245	95.238 %	0.0476
VGG19	49	35	0	0	1	1	100 %	0
Resnet50	49	21	0	14	1	0.7777	83.33 %	0.1666
GoogleNet inspection V3	49	0	0	35	1	0.5833	58.33 %	0.4166

Table 11 show the test prediction of applied 344 images captured from camera and after detect the face region by Haar cascade detector, cutting the images and input them to the different models and proposed model show that the accuracy and loss are better than other models because in the Alex net model is not good result refer to the reason that image size is small and the layers unsuccessful to extract features. In the VGG16 and VGG19 models show that the result is good but still proposed model is high result .In the Resnet and GoogleNet inspection V3 show the results are not good because these models consist high layers are not suitable of applied on the real data set to extract the features.

Table 11: Confusion matrix for the different models and proposed model all applied on the original real data set.

Test prediction models	True positive	True negative	False positive	False negative	Precision	Recall (TPR)	Test prediction Accuracy	Mean square error
Alex net	228	33	0	83	1	0.7331	75.87 %	0.2412
VGG16	227	108	1	8	0.9956	0.9659	97.38%	0.0261
VGG19	226	113	0	5	1	0.9783	98.54 %	0.0145
Resnet50	228	64	0	52	1	0.8142	84.88 %	0.1511
GoogleNet inspection V3	228	0	0	116	1	0.66	66.27 %	0.3372
Proposed model	228	115	0	1	1	0.9956	99.70 %	0.0029

Table 12 shown the test prediction time applied on the 344 images input to the all different models with proposed model that the test time of the proposed model is better than the others models.

Table 12: Test time of the different models with proposed model.

Models	**Test Time (second)**
Alex net	6
VGG16	21
VGG19	22
Resnet50	15
GoogleNet inspection V3	35
Proposed model	4

3.7 Multi class identification

This section proposes a CNN model for the identification of images that belong to the non-permission class taken from the first stage after inputting them to this model as a test. The model was used to train 12 classes. As shown in Fig 21, the three kids are put in the same class by making a new data set that combines the real data set with benchmark data found on the internet.

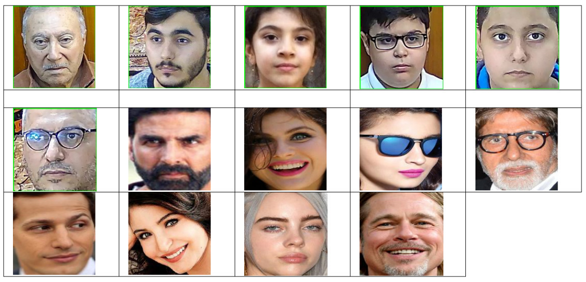


Figure 21: The real data set plus bench mark data set for 12 classes.

The proposed CNN model's architecture is shown in Table 13. The total images are using equal 595 of multi-class, 0.8 of training, and 0.2 of validation. After the training with the 12 classes, the validation accuracy was 97.48% and the loss was 0.0571.

The activation function used in the proposed multi-class model is the SoftMax function which returns the probability of each class, as expressed in equation 3.8 [19].

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}} \tag{3.8}$$

where z indicates the values of the output layer neurons, Exponentiation serves as the nonlinear function. These values are subsequently normalized and converted into probabilities by dividing them by the sum of exponential values. The specified features vector location was 400 values. The cost function is a function used to minimize a batch of images. Let $\acute{\alpha} i$ be the values returned by the model, and let αi be the true labels. S is the number of images in the batch. The mean of summation loss for a batch of images used as a Cost Function [7] is defined as in Eq.3.9.

$$\text{cost}(\alpha, \acute{\alpha}) = \frac{1}{s}\sum_{i=1}^{s} [\text{loss}\,(\alpha, \acute{\alpha})]\,. \tag{3.9}$$

The confusion matrix of the predicted and actual data of the multi-class model as test is shown in Table 14. Each column describes each class (image) prediction in the confusion matrix, where 119 images were used for testing 12 classes.

Table 13: The CNN proposed model multi class architecture.

Layer (Type)	**Output (shape)**	**Parameters**
conv2d (Conv2D)	(None, 138, 138, 16)	448
conv2d_1 (Conv2D)	(None, 134, 134, 32)	12832
conv2d_2 (Conv2D)	(None, 132, 132, 32)	9248
batch normalization	Batch No (None, 132, 132, 32)	128
activation (Activation)	(None, 132, 132, 32)	0
max_pooling2d (MaxPooling2D)	(None, 66, 66, 32)	0
conv2d_3 (Conv2D)	(None, 62, 62, 64)	51264
batch_normalization_1	(None, 62, 62, 64)	256
activation_1 (Activation)	(None, 62, 62, 64)	0
max_pooling2d_1 (MaxPooling2)	(None, 31, 31, 64)	0
conv2d_4 (Conv2D)	(None, 29, 29, 32)	18464
batch_normalization_2 (Batch)	(None, 29, 29, 32)	128
activation_2 (Activation)	(None, 29, 29, 32)	0
max_pooling2d_2 (MaxPooling2)	(None, 14, 14, 32)	0
flatten (Flatten)	(None, 6272)	0
dense (Dense)	(None, 400)	2509200
activation_3 (Activation)	(None, 400)	0
dropout (Dropout)	(None, 400)	0
dense_1 (Dense)	(None,12)	4812
activation_4 (Activation)	(None, 12)	0

Table 14: The confusion matrix between the predicted (p) and the actual multi class data.

Classes	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12
Class 1	10	0	0	0	0	0	0	0	0	0	0	0
Class 2	0	9	0	0	0	0	0	0	0	0	1	0
Class 3	0	0	10	0	0	0	0	0	0	0	0	0
Class 4	0	0	0	10	0	0	0	0	0	0	0	0
Class 5	0	0	0	0	9	0	0	0	0	0	0	0
Class 6	0	0	0	0	0	10	0	0	0	0	0	0
Class 7	0	0	0	0	0	0	10	0	0	0	0	0
Class 8	0	0	0	1	0	0	0	9	0	0	0	0
Class 9	0	0	0	0	0	0	0	0	10	0	0	0
Class 10	0	0	0	0	0	0	1	0	0	9	0	0
Class 11	0	0	0	0	0	0	0	0	0	0	10	0
Class 12	0	0	0	0	0	0	0	0	0	0	0	10

The F1 Score is called the F Score or the F Measure. The F1 score conveys the balance between the precision and the recall, as shown in eq. 3.10, and represents the quality of the classification.

$$\text{F1} = \frac{2(precision * recall)}{(precision + recall)} \tag{3.10}$$

The classification report for 119 images of all classes, with values of precision, recall, and F1 score, is shown in Table 15. F1 score average is equal 0.975 for all classes, indicating that the classifier is confers high quality and accuracy upon running the prediction process to identify the persons who belong to the non-permission class taken from the first stage.

Table 15: The classification report for each class of the predictions or testing.

Classes	precision	Recall	F1 score	Support sample (images in each class)
0	1	1	1	10
1	1	0.9	0.95	10
2	1	1	1	10
3	0.91	1	0.95	10
4	1	1	1	9
5	1	1	1	10
6	0.91	1	0.95	10
7	1	0.9	0.95	10
8	1	1	1	10
9	1	0.9	0.95	10
10	0.91	1	0.95	10
11	1	1	1	10

4 Conclusion

This paper presented face recognition or classification model with two classes, the first is the permission and the second is the non-permission class, applied on a real data set collected by fixed cameras. The main idea is to enable the system to distinguish if a person has the permission or not to enter the building. The proposed offline augmentation built-from-scratch CNN model showed low complexity, low layers, smallest filters sizes and more reliability when applied on RGB images with a size of 140×140 pixels. The proposed model demonstrated that the validation accuracy and classification parameters were better than the online augmentation, which required more time for image generation and training, because it fit inside the proposed CNN classification model, whereas the offline augmentation, which generated images from the original real data set, fit outside the proposed CNN classification model. This is due to the notion that data augmentation is a set of approaches applied for increasing the size and quality of training datasets so that stronger deep learning models can be built. This optimization strategy is used not only to increase data points but also to avoid over fitting, as training a model on numerous data points is the best way to avoid over fitting. The second stage used three types of filters as denoising tools on the real data set images before training. The median filter showed the highest smoothing effect and classification quality. These two stages improved the accuracy, precision, recall, and AUC of the system, whereas the values of loss validation and the mean square error were reduced. The third stage used the multi classification CNN light complexity model to identify the non-permission class from 12 classes. This identification model improved the performance metrics of the system and achieved a high F1 score value, indicating that the identification or classification model is of high quality and accuracy on testing mode.

References

[1] S.H. Abdulredah and D.J. Kadhim, *New approaches of cloud services access using Tonido cloud server for real-time applications*, J. Engin. **26** (2020), no. 8, 83–99.

[2] B. Beddad and K. Hachemi, *Efficient implementation of an improved median filter on TMS320C6416 digital signal processor*, Proc. IEEE Int. Conf. Electric. Sci. Technol. Maghreb, Algiers, Algeria, 2018.

[3] L. Cuadros-Rodrigues, E. Perez-Castano and C. Ruiz-Samblas, *Quality performance metrics in multivariate classification methods for qualitative analysis*, TrAC Trends Anal. Chem. **80** (2016), 612–624.

[4] M. Hafiz Ishak, N.Sofia, M. Marzuki, M. Abdullah, Z. Soh, I. Isa and S. Sulaiman, *Image quality assessment for image filtering algorithm: Qualitative and quantitative analyses*, Proc. IEEE 9th Int. Conf. Cont. Syst. Comput. Engin., Penang, Malaysia, 2019, pp. 162–167.

[5] K. He, X. Zhang, S. Ren and J. Sun, *Deep residual learning for image recognition*, Proc. IEEE Conf. Comput. Vision Pattern Recog., 2016, pp. 770–778.

[6] J. Huber, *Batch normalization in 3 levels of understanding*, Towards Data Science, 2020.

[7] L. Hughes, M. Schmitt, L. Mou, Y. Wang, X. Zuh and R. Letters, *Identifying corresponding patches in SAR and optical images with a pseudo-siamese CNN*, IEEE Geosci. Remote Sens. Lett. **15** (2018), 784–788.

[8] S.Q. Jabbar, D.J. Kadhim, *A proposed adaptive bitrate scheme based on bandwidth prediction algorithm for smoothly video streaming*, J. Engin. **27** (2021), no. 1, 112–129.

[9] S.Q. Jabbar, D.J. Kadhim and Y. Li1, *Developing a video buffer framework for video streaming in cellular networks*, Wireless Commun. Mobile Comput. **2018** (2018).

[10] M.A. Joodi, M.H. Saleh and D.J. Kadhim, *Increasing validation accuracy of a face mask detection by new deep learning model-based classification*, Indones. J. Electric. Engin. Comput. Sci. **29** (2023), 304–3014.

[11] D.J. Kadhim and O.A. Hamad, *Hamad Improving IoT applications using a proposed routing protocol*, J. Engin. **20** (2014), no. 11, 50–62.

[12] N. Kan, N. Kondo, W. Chinsatit and T. Saitoh, *Effectiveness of data augmentation for CNN-based pupil center point detection*, Proc. IEEE 57th Ann. Conf. Soc. Instrum. Cont. Engin. Japan, Nara, Japan, 2018, pp. 441-464.

[13] D. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 3rd Int. Conf. Learn. Represent., San Diego, 2015.

[14] A. Krizhevsky, I. Sutskever and G.Hinton, *Imagenet classification with deep convolutional neural networks*, Commun. ACM **60** (2017), no. 6, 84–90.

[15] M. Kutlugün, Y. Şirin and M. Karakaya, *The effects of augmented training dataset on performance of convolutional neural networks in face recognition system*, Proc. IEEE, Federated Conf. Comput. Sci. Inf. Syst., Leipzig, Germany, 2019, pp. 929—932.

[16] K. Lakhwani, H. Gianey and Sh. Gupta, *An enhanced approach to improve UIQI and PSNR of noised colored images using DWTT filter*, Proc. IEEE Int. Conf. Comput. Power Commun. Technol., Greater Noida, India, 2018, pp. 289–293.

[17] M. Ofori-Oduro and M. Amer, *Data augmentation using artificial immune systems for noise-robust CNN models*, Proc. IEEE Int. Conf. Image Process. (ICIP), Abu Dhabi, United Arab Emirates, 2020, pp. 833-837.

[18] J. Rasheed, E. Alimovski, A. Rasheed, Y. Sirin, A. Jamil and M. Yesiltepe, *Effects of glow data augmentation on face recognition system based on deep learning*, Proc. IEEE Int. Cong. Human-Comput. Interact. Optim. Robotic. Appl., Ankara, Turkey, 2020.

[19] S. Saxena, *Introduction to Softmax for Neural Network*, Analytic Vidhya, 2021.

[20] K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, Comput. Vision Pattern Recog. Vers. 4, 6, Conference ICLR , 2015.

[21] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna, *Rethinking the inception architecture for computer vision*, Proc. IEEE Conf. Comput. Vision Pattern Recog., 2016, pp. 2818–2826.

[22] A. Vikramathithan, S. Bhat and D. Shashikumar, *Denoising high density impulse noise using Duo-Median filter for mammogram images*, Proc. IEEE Int. Conf. Smart Technol. Comput. Electric. Electronic., Bengaluru, India, 2020, pp. 610–613.

[23] D. Villar, S. Torcida and G. Acosta1, *Median filtering: A new insight*, J. Meth. Imag. Vision **58** (2017), 130–146.

[24] D. Wang, D. Wang, Hongzhi Yu and G. Li, *Face recognition system based on CNN*, Proc. IEEE Int. Conf. Comput. Inf. Big Data Appl., Guiyang, China, 2020, pp. 470–473.

[25] Y. Weng and H. Zhou, *Data augmentation computing model based on generative adversarial network*, IEEE Access **7** (2019), 64223–64233.

[26] M. Wani, F. Bhat, S. Afzal and A. Khan, *Advances in deep learning*, Studies in Big Data, Springer, 2020.

[27] G. Wimmer, A. Uhl and A. Vecsei, *Evaluation of domain specific data augmentation techniques for the classification of celiac disease using endoscopic imagery*, Proc. IEEE 19th Int. Workshop on Multimedia Signal Proces., Luton, UK, 2017.