Journal of Engineering

journal homepage: www.joe.uobaghdad.edu.iq

Volume 29 Number 4 April 2023

# Proposed Face Detection Classification Model Based on Amazon Web Services Cloud (AWS)

**Mohanad Azeez Joodi ***
Lecturer
Dept. of Electrical Engr.,
College of Engr., Univ. of Baghdad,
Baghdad, Iraq
m.mohammad1702d@coeng.uobaghdad.edu.iq

**Muna Hadi Saleh**
Assist. Prof., Ph.D
Dept. of Electrical Engr.,
College of Engr., Univ. of Baghdad
Baghdad, Iraq
dr.muna.h@coeng.uobaghdad.edu.iq

**Dheyaa Jasim Kadhim**
Assist. Prof., Ph.D
Dept. of Electrical Engr.,
College of Engr., Univ. of Baghdad
Baghdad, Iraq
dheyaa@coeng.uobaghdad.edu.iq

## ABSTRACT

One of the most important features of the Amazon Web Services (AWS) cloud is that the program can be run and accessed from any location. You can access and monitor the result of the program from any location, saving many images and allowing for faster computation. This work proposes a face detection classification model based on AWS cloud aiming to classify the faces into two classes: a non-permission class, and a permission class, by training the real data set collected from our cameras. The proposed Convolutional Neural Network (CNN) cloud-based system was used to share computational resources for Artificial Neural Networks (ANN) to reduce redundant computation. The test system uses Internet of Things (IoT) services through our cameras system to capture the images and upload them to the Amazon Simple Storage Service (AWS S3) cloud. Then two detectors were running, Haar cascade and multitask cascaded convolutional neural networks (MTCNN), at the Amazon Elastic Compute (AWS EC2) cloud, after that the output results of these two detectors are compared using accuracy and execution time. Then the classified non-permission images are uploaded to the AWS S3 cloud. The validation accuracy of the offline augmentation face detection classification model reached 98.81%, and the loss and mean square error were decreased to 0.0176 and 0.0064, respectively. The execution time of all AWS cloud systems for one image when using Haar cascade and MTCNN detectors reached three and seven seconds, respectively.

**Keywords:** CNN, AWS EC2, MTCNN, validation accuracy, execution time.

*Corresponding author

Peer review under the responsibility of University of Baghdad.
https://doi.org/10.31026/j.eng.2023.04.12

Article received: 30/11/2022
Article accepted: 02/03/2023
Article published: 01/04/2023

# نموذج تصنيف اكتشاف الوجه المقترح استنادًا إلى سحابة خدمات موقع الامزون

مهند عزيز جودي*
مدرس
كلية الهندسة جامعة بغداد

منى هادي صالح
أستاذ مساعد , دكتوراه
كلية الهندسة جامعة بغداد

ضياء جاسم كاظم
أستاذ مساعد, دكتوراه
كلية الهندسة جامعة بغداد

**الخلاصة**

من أهم ميزات سحابة خدمات موقع الامازون أنه يمكن تشغيل البرنامج المبرمج والوصول إليه من أي مكان ، ويمكنك الوصول إلى نتيجة المبرمجة ومراقبتها من أي مكان ، وحفظ العديد من الصور والسماح بشكل أسرع للحسابات. يقترح هذا العمل نموذجًا لتصنيف اكتشاف الوجوه استنادًا إلى سحابة AWS بهدف تصنيف الوجوه إلى فئتين: الأولى فئة لا يسمح لها بدخول بناية او موقع مهم ، والثانية فئة يسمح لها بالدخول ، من خلال تدريب مجموعة البيانات الحقيقية التي تم جمعها من كاميراتنا. تم استخدام النظام القائم على السحابة المقترح للشبكة العصبية الملتفة (CNN) لمشاركة الموارد الحسابية للشبكة العصبية الاصطناعية (ANN) من أجل تقليل الحسابات الزائدة عن الحاجة. يتم تشغيل نظام الاختبار باستخدام خدمات إنترنت الأشياء (IOT)من خلال نظام الكاميرات لدينا لالتقاط الصور وتحميلها إلى سحابة التخزين البسيط لخدمات موقع الامازون AWS) S3)، ثم تشغيل كاشفين ، Haar cascade وشبكة عصبية ملتفة متعددة المهام (MTCNN) في خدمات موقع الامازون للحوسبة السحابية المرنة Amazon Elastic Compute (AWS EC2) ، بعد ذلك تتم مقارنة نتائج مخرجات هذين الكاشفين باستخدام الدقة ووقت التنفيذ. ثم يتم تحميل الصور المصنفة بدون إذن إلى سحابة AWS S3 .وصلت دقة التحقق من صحة نموذج تصنيف الكشف عن الوجه في وضع عدم الاتصال إلى 98.81٪ ، وانخفض الخسارة ومتوسط الخطأ التربيعي إلى 0.0176 و 0.0064 على التوالي. بلغ وقت تنفيذ جميع أنظمة سحابة AWS لصورة واحدة عند استخدام Haar cascade وكاشفات MTCNN ثلاث وسبع ثوانٍ على التوالي.

**الكلمات الرئيسية:** الشبكة العصبية الملتفة ، خدمات موقع الامازون للحوسبة السحابية المرنة ، شبكة عصبية ملتفة متعددة المهام, دقة التحقق, وقت التنفيذ.

## 1. INTRODUCTION

The massive adoption and expansion of IoT in these industries generate massive amounts of data. In hospital surveillance applications, for instance, IoT devices such as cameras generate tremendous images. Face recognition is crucial for safeguarding medical facilities; detecting patient fraud; analyzing hospital traffic patterns; and analyzing patients' emotions and sentiments. Automatic and intelligent face recognition systems have high accuracy in a controlled environment; in an uncontrolled one, they have low accuracy. The systems must also function in real-time for various applications, including smart healthcare. A deep tree-based method for cloud-based facial recognition software is studied. The suggested deep model uses less computing time without compromising accuracy. A volume of input is divided into several volumes in the model, and a tree is generated for each volume **(Masud et al., 2020; Joodi, 2023)**. Facial recognition and cloud-based mobile edge computing are suggested to provide an immersive online biometric authentication method for online

guiding. A combination of technologies was used to create an effective model. The suggested framework is verified against various state-of-the-art methods **(Saad, 2018; Su, 2021)**.
The cloud-based system was used to share computational resources for ANN to reduce redundant computation. A cloud-based intelligent monitoring system was presented to provide intelligent monitoring services. Built into the system are hybrid convolutional neural networks. This technology has been used for several intelligent monitor functions, such as recognizing strangers, recognizing facial expressions, and recognizing activities **(Yong et al., 2016)**. A cloud-based deep learning face video retrieval system is represented. A dataset is initially collected and preprocessed. To generate a viable dataset for CNN models, blurry photos are eliminated, and face alignment is performed on the remaining images. The resulting dataset is then used to pre-train the CNN models (VGG Facial, Arc Face, and Face Net) for face recognition **(Jabbar, 2018; Lin et al., 2020)**.
Using a cloud-computing infrastructure, this study aims to develop and implement automatic customer face recognition. This platform employs cloud-based features to overcome traditional systems' resource and scalability limitations, which rely on local computing capacity. When a consumer is detected, the front-end device takes his or her photograph. The image is saved in the cloud and analyzed through software as a Service (SaaS) **(Dersingh, 2016 )**. Using cloud Hopfield neural network (CHNN), a method is described for identifying low-resolution grayscale face pictures. This strategy comprises three steps: First, Otsu's approach is used to convert grayscale facial pictures into binary facial images, then the Hebb rule is used to store binary faces in the weight matrix of the network, and lastly, the CHNN retrieval algorithm is used to return the proper face from a deformed face.
In contrast to typical asynchronous retrieval, in which only a single neuron is updated at a time, CHNN consists of clouds containing several distinct neurons that are updated asynchronously **(Neha, 2016)**. Several emerging technologies are discussed, including the idea of a smart hotel, artificial intelligence, face recognition, key algorithms, cloud computing, big data, AI (artificial intelligence), the Internet of Things, and others. People's faces are studied from the perspective of the smart hotel so that facial recognition technology may be applied to them **(Chen, 2020)**. Utilizing a local server and the Amazon Web Service (AWS) cloud recognition Application Programming Interface, the real-time attendance tracking employs a remote-operable web application (API). The first method consists of five sections: face detection, preprocessing, training, and face recognition, through which attendance is recorded and sent to each teacher. The second method is based on the AWS Recognition API, which analyses cloud-based data **(Pattnaik, 2020)**.
In an anomaly-based technique for unauthorized entry detection and signature analysis, a face recognition algorithm running on the AWS cloud is used to tell an authorized person from an intruder, improving the accuracy of authorizing the legitimate person and granting access to the private/personal zone, and lowering the risk of sending false alerts or alarms **(Mahendra, 2020)**. AWS Deep Lens is an embedded device built for machine learning that compares the device's performance while using Amazon Web Services (AWS) cloud computing services for an AI-trained model against on-premises performance deep learning **(Gregorius Rafael, 2020; Jabbar, 2021)**. Under the thorough examination of cloud QoS and media application needs, a performance evaluation technique for the image and video services offered by various cloud platforms is developed. In the current state of technology find the most popular picture and video cloud services. These include facial

recognition, image analysis, optical character recognition, video on demand, live streaming, and transcending **(Xue et al., 2018)**.
Face recognition and Amazon Web Services, such as S3 and Quick Sight will be used to create an automatic employee attendance system. The opening-closing procedure of the doors is modified based on the face recognition findings **(Sharma et al., 2020a)**. Using a Raspberry Pi camera coupled with a Raspberry Pi, a visitor rating system is constructed. The system relies mainly on the cloud services provided by AWS for storing and evaluating the received ratings. The system uses a Facial Emotion Recognition model to get real ratings from real visitors **(Sharma et al., 2020b)**. This system is built on an OpenCV module for image recognition to create a face recognition application that may be used for check-in. OpenCV is the principal control recognition module that detects and tracks the target face using image processing.
In the meantime, Baidu Cloud holds a face database and offers face recognition and matching scores for face photos **(Feng et al., 2021)**. This study looks at real-time security and surveillance systems that use cloud-based facial recognition, focusing on cloud architecture **(Jha et al., 2022)**. Our faces change with age, yet the photographs in our database remain unchanged. We plan to investigate the precision of Residual Network (ResNet) for cross-age face recognition. Cross-age reference coding (CARC), Amazon Web Services (AWS) Recognition, and other approaches are compared to the cross-age celebrity dataset (CACD) and a verification subset (CACD-VS) performance **(Babbar, 2019)**.
The use of Haar cascade with CNN is proposed for face detection. Haar cascade is a method for detecting faces quickly and in real time. CNN also uses the convolution process by moving a convolution kernel of a certain size from one picture to the next based on what happens when the current picture is multiplied by the filter being used **(Asmara, 2021)**. Deep learning is now the most significant technology in computer vision. It can extract more critical facial characteristics automatically compared to conventional face recognition systems. A face recognition system is developed using the neural computing paradigm and the neural network concept. The results of the experiments show that the suggested method has a high rate of detection and a quick processing time **(Yu and Pei, 2021)**. The roles of convolutional neural network-based deep learning approaches for object detection are explained. Object identification frameworks and services based on deep learning are also described. Modern approaches to deep learning for object identification systems are evaluated **(Kadhim, 2023; Pathak, 2018)**.
The main contribution of this work is proposing a CNN cloud-based system that can be used to share computational resources for ANN to reduce redundant computation to train and test the real data set created from our camera system. Also, it is used in entering it into one of the two detectors; the first is called the Haar cascade detector and the second is called the multitask cascaded convolutional neural networks (MTCNN) detector. The test process is running by using AWS cloud (elastic computation (EC2) and simple service storage (S3)). In addition to enhancing the test prediction time using the AWS paid cloud and comparing it with the AWS free cloud. Access the test prediction from any location by AWS cloud, and save many captured images on the AWS S3 camera. Show the effect of the image capture size on the test execution time.
The following is how this study is set up: The research background for the face recognition system employing IOT (AWS cloud) and a literature review of face recognition methods that use enhanced convolutional neural networks are both provided. The architecture of the

proposed cloud classification model is described in two parts, these are: a description of the proposed cloud infrastructure; and a description of the Haar cascade and MTCNN detectors. The permission and non-permission classes' three subsections make up Section 3's experimental results and discussion: training the real data set; classification performance; and testing the predictions system as a whole, utilizing the free AWS cloud service and comparing it to a paid service of the same provider.

## 2. PROPOSED CLASSIFICATION MODEL BASED ON AWS CLOUD

An actual data set acquired from stationary cameras was subjected to the proposed categorization model. The suggested low-complexity CNN offline augmentation model of face recognition and classification, divided into the categories of "permitted person" and "non-permitted person," must be built from scratch. This procedure is designed to identify people permitted to enter high-security locations like airports or tourist sites. The Amazon Web Services (AWS) cloud may be used to test the complete system, and the AWS application allows access to the images from anywhere. IoT is currently quite significant. A bucket in Amazon S3 is a publicly available cloud storage resource that may be accessed through the object storage service Simple Storage Service (S3) from Amazon Web Services (AWS). Amazon S3 buckets store objects made up of data and associated identifying information, much like file folders do. IaaS is well-exemplified by AWS EC2. EC2 offers a scalable infrastructure for hosting cloud-based applications. AWS is an Amazon corporation that provides individuals, companies, and government organizations with on-demand cloud computing platforms and APIs. Through AWS server farms, these cloud computing web services provide software tools and the capability to perform distributed computing. The proposed model topological structure and architecture are shown in **Fig. 1 and Table 1,** respectively.
To create a convolution, two fundamental functions (f) and (g) must be multiplied by a (n). This expression describes a convolution in one dimension (Eq. (1)) **(Sharma et al., 2020a):**

$$(f * g)(n) = \sum_{m} f(m)g(n-m) \tag{1}$$

Two-dimensional (2D) digital images can be produced using convolution if (A) is a 2D image with (i + j) dimensions, (K) is a filter with (m + n) dimensions, and (F) is a feature map. To obtain the output F, picture A is convolved with the filter K. Eq. (2) defines the operation in this situation because it is commutative, and Eq. (3) below can be used to represent the 2D equation:

$$F(i,j) = (A * K)(i,j) = \sum_{m}\sum_{n} A(m,n)K(i-m, j-n) \tag{2}$$

$$F(i,j) = (A * K)(i,j) = \sum_{m}\sum_{n} A(i-m, j-n)K(m,n) \tag{3}$$

The activation function of RELU ignores negative values. Additionally, adaptive moment estimation (Adam) is an optimization approach that may replace the standard SGD technique to update network parameters depending on training data **(Vinh, 2020)**. Adam converges more quickly than other approaches, according to empirical evidence. A frequent strategy for generalization is a dropout.

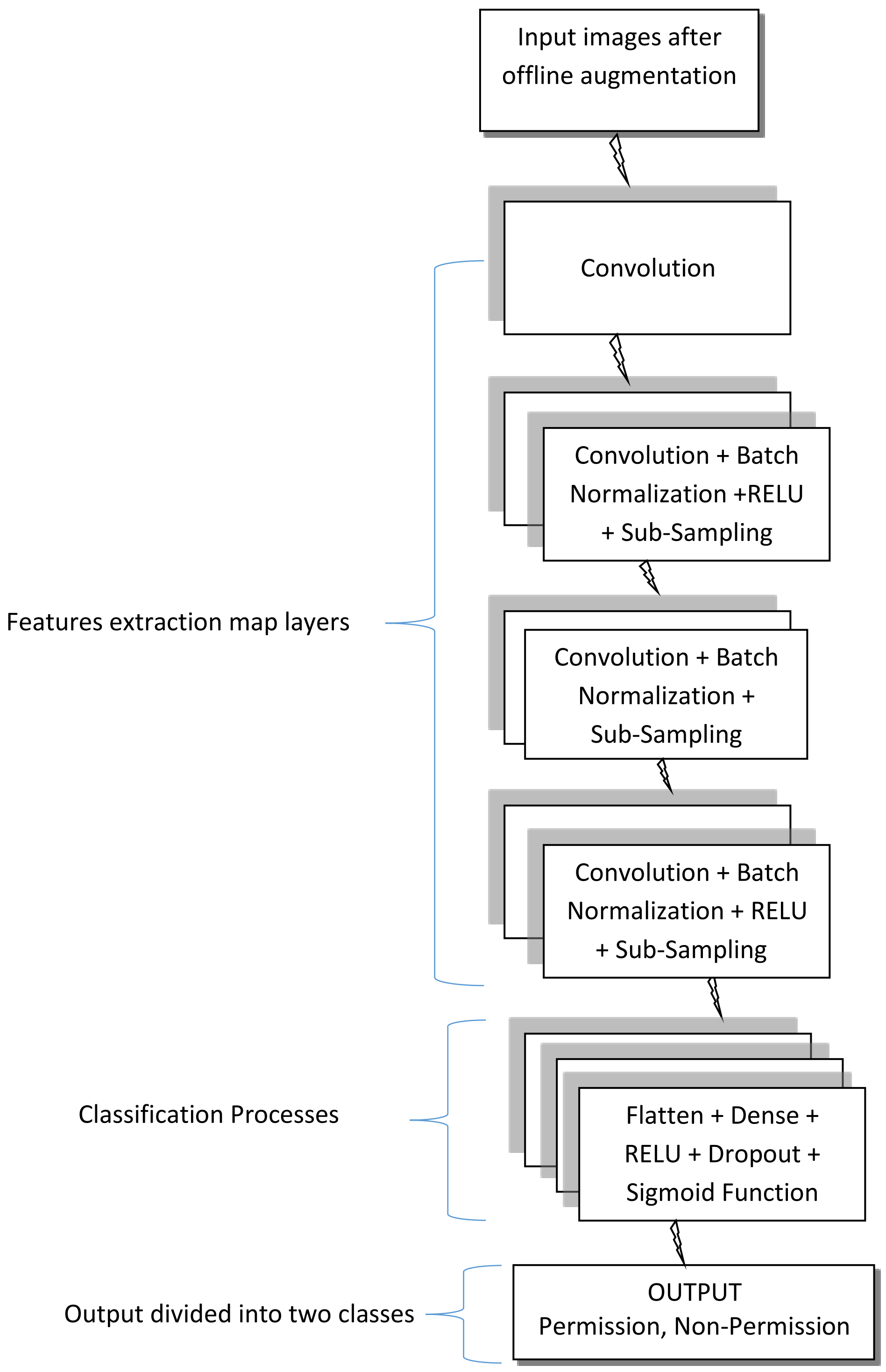

Input images after offline augmentation
Convolution
Convolution + Batch Normalization +RELU + Sub-Sampling
Convolution + Batch Normalization + Sub-Sampling
Convolution + Batch Normalization + RELU + Sub-Sampling
Features extraction map layers
Flatten + Dense + RELU + Dropout + Sigmoid Function
Classification Processes
OUTPUT Permission, Non-Permission
Output divided into two classes

**Figure 1.** The topological structure of the AWS cloud CNN proposed model

**Table 1.** The architecture proposes a CNN model with offline data augmentation applied on the AWS EC2 cloud

| 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 |

Neurons are randomly discharged during each training session. A loss function that measures how far apart two values are from one another in the 0 to 1 is called binary cross-entropy. The loss function shown in Eq. (4) **(Hughes, 2018)** can be considered the definition of binary cross-entropy.

$$loss(\alpha, \acute{\alpha}) = -\left(alog(\acute{\alpha}) + (1-a)log(1-\acute{\alpha})\right) \quad (4)$$

where the true label value is α and ά is the value the model returned. Minimizing (α, ά) several photos at once is common practice.

## 2.1 AWS Cloud infrastructure Description

The proposed cloud architecture is based on Amazon Web Services. Our infrastructure is the two instances of size t2. micro and t3a. large availability in EC2. **Table 2** shows the EC2 free tier performance, **Table 3** shows the EC2 paid tier performance, and **Table 4** shows the hardware resources used in our system.

**Table 2.** AWS EC2 free tier performance

| | |
|---|---|
| Instance type | t2. micro |
| Availability zone | Us-east-2c |
| Architecture CPU | AMD 64 |
| Total memory (RAM) | 1024 MB |
| Network performance | Low to moderate |
| OS | Windows Server 2019, 64 bit |
| Hard Disk (HD) | 30 GB |

**Table 3.** AWS EC2 paid tier performance

| | |
|---|---|
| Instance type | t3a.large |
| Availability zone | Us-east-2c |
| Architecture CPU | AMD EPYC 7571 2.2 GHz |
| Total memory (RAM) | 8192 MB |
| Network performance | Up to 5 Gigabit |
| OS | Windows Server 2019, 64 bit |
| Hard Disk (HD) | 30 GB |

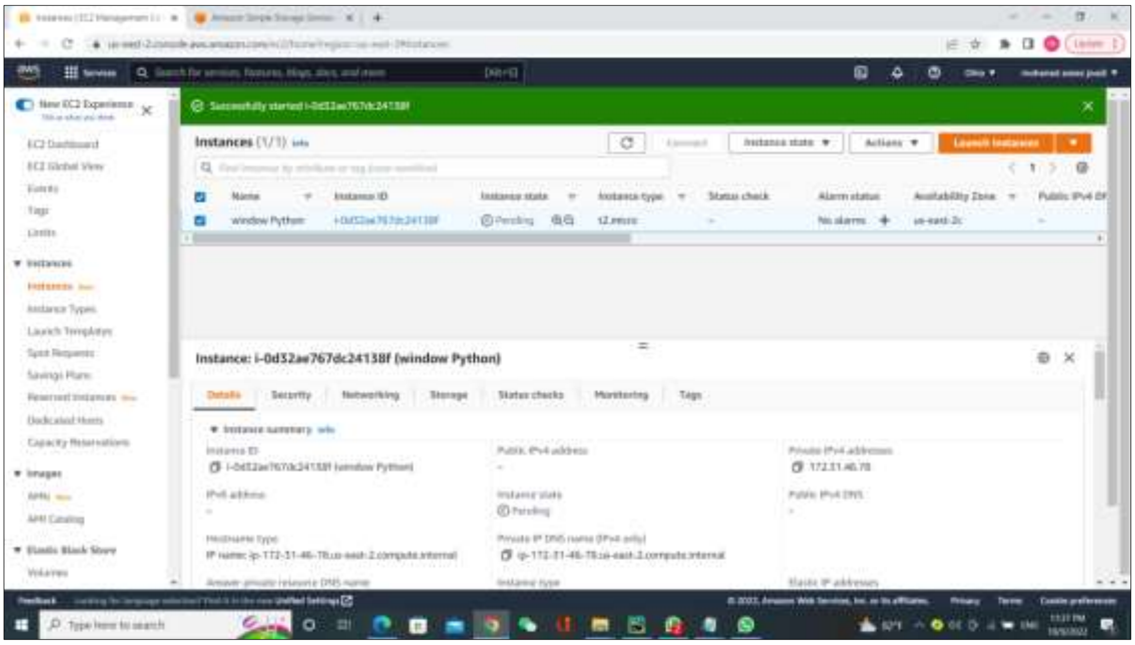
**Table 4.** Hardware equipment's used

| Hardware resources | Quantity | Type |
|---|---|---|
| Camera | 4 | 3.6 mm dhua IP camera |
| network video recorder (NVR) | 1 | DHIT-NVR4108HS-4KS2/L |
| Router | 1 | D-Link router DIR-X1560 |
| Power of Ethernet (POE) switch | 1 | ONV-H1064PLD. |

Elastic Compute Cloud (EC2) Instance from Amazon. It may provide enterprises with almost an infinite number of virtual machines. The goal of this design was to simplify web-scale cloud computing for developers. The Windows Server 2019 operating system is installed on an EC2 Windows instance that hosts a website in the AWS cloud. The Windows instance free tier and paid tier are depicted in **Figs. 2** and **3,** respectively.

## 2.2 Describe the Haar cascade and MTCNN Detectors

Face recognition in still photos and live streams is accomplished using the object detection method known as the Haar cascade. Edge or line detection characteristics were created by Viola and Jones **(Viola, 2001)**. Models are stored in XML files in the repository that may be accessed using OpenCV methods. These versions include the face, eye, upper and lower body, license plate, and other detecting features. **Fig. 4** illustrates some of the concepts offered by Viola and Jones. Due to these features of the image, it is easy to spot its edges or lines and places where the pixel intensities quickly change, as seen in **Fig. 5**.



Successfully started i-0d32ae767dc24138f

Instances (1/1)

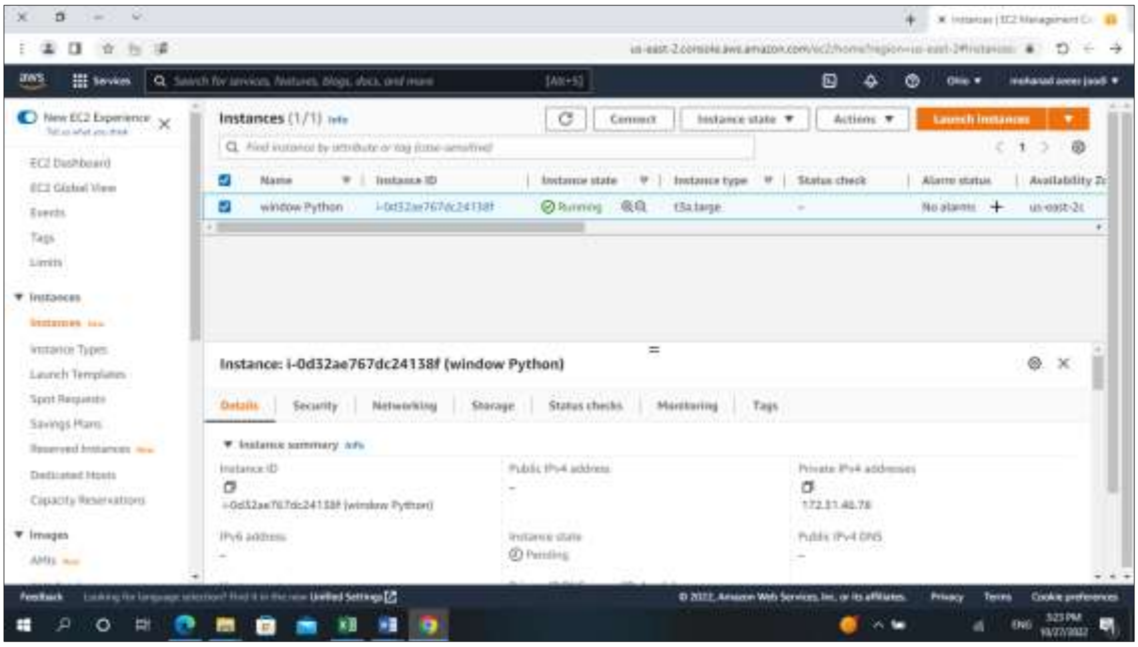
| Name | Instance ID | Instance state | Instance type | Status check | Alarm status | Availability Zone |
|---|---|---|---|---|---|---|
| window Python | i-0d32ae767dc24138f | Pending | t2.micro | – | No alarms | us-east-2c |

Instance: i-0d32ae767dc24138f (window Python)

Instance summary

Instance ID: i-0d32ae767dc24138f (window Python)
Public IPv4 address: –
Private IPv4 addresses: 172.31.46.78
IPv6 address: –
Instance state: Pending
Public IPv4 DNS: –
Hostname type: IP name: ip-172-31-46-78.us-east-2.compute.internal
Private IP DNS name (IPv4 only): ip-172-31-46-78.us-east-2.compute.internal

Figure 2. AWS EC2 free tier



Instances (1/1)

| Name | Instance ID | Instance state | Instance type | Status check | Alarm status | Availability Zone |
|---|---|---|---|---|---|---|
| window Python | i-0d32ae767dc24138f | Running | t3a.large | – | No alarms | us-east-2c |

Instance: i-0d32ae767dc24138f (window Python)

Instance summary

Instance ID: i-0d32ae767dc24138f (window Python)
Public IPv4 address: –
Private IPv4 addresses: 172.31.46.78
IPv6 address: –
Instance state: Pending
Public IPv4 DNS: –

Figure 3. AWS EC2 paid tier

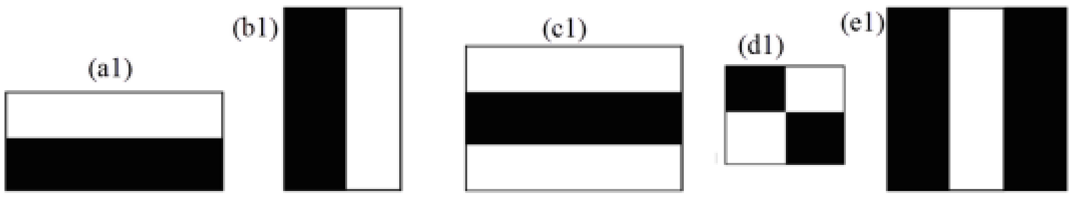


**Figure 4.** Some features of Haar Cascade

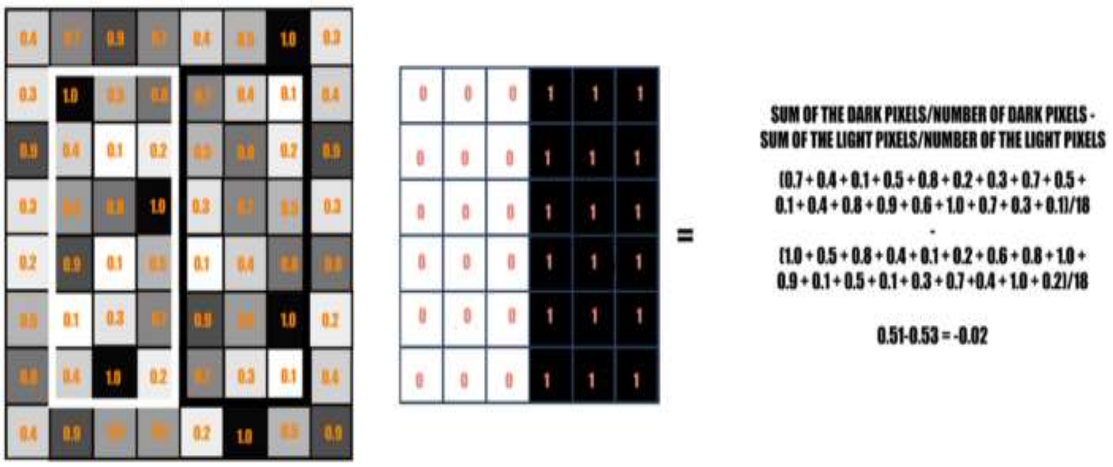


**Figure 5.** Image and Haar cascade kernel

An illustration of an image with pixel values ranging from 0.0 to 1.0 is the rectangle on the left. With all bright pixels on the left and dark pixels on the right, the rectangle in the center is a Haar kernel. The Haar computation calculates the difference between the average pixel values in the brighter and darker regions. Haar identifies an edge if the difference is fairly close to 1. Here is an illustration of calculating the Haar value using a rectangular image slice. Pixels with a value of 1 make up the Haar feature's darker areas, whereas pixels with 0 make up its brighter areas. Each of them is in charge of recognizing a certain visual feature, such as an edge, a line, or any other visual structure with a sudden change in intensity. As the sample image above shows, the Haar feature may detect a vertical border with darker pixels on its right and brighter pixels on its left. This operation aims to calculate the sum of all image pixels situated in the brighter region of the Haar feature and the total of all image pixels situated in the darker area of the Haar feature. Then decide what makes them unique. The Haar value is closer to 1 if the image has an edge separating bright pixels on the left from dark pixels on the right. If the Haar value is near 1, an edge has been found.
A framework called Multitask Cascaded Convolutional Networks (MTCNN) was created as a face alignment and identification solution. Convolutional networks are used in three process stages to recognize faces and facial landmarks such as the eyes, nose, and mouth. The research suggests MTCNN merge both tasks using multitask learning (recognition and alignment). In the initial stage, candidate windows are quickly developed using a shallow CNN. The proposed candidate windows are improved in the second step using CNN. A third,

more complex CNN is employed in the final stage to refine further the output and output of the positions of landmarks **(Zhang, 2016).**

## 3. RESULTS AND DISCUSSION

The third component comprises parts about training the proposed CNN model; the first section prepares the real data set containing images of humans acquired by building-mounted cameras. This division consists of two classes: with permission and without authorization. The second step consists of training the real data set with offline augmentation applied to the real data set. The third phase utilizes performance metrics to assess the quality and accuracy of the model. In the fourth part, permission and non-permission AWS cloud users are classified based on a test prediction of some photographs.

### 3.1 Dataset

For the experiment, a genuine data set of 347 frontal faces with a minimum size of 100 × 100 pixels taken by cameras was employed. The dataset consists of two training and validation sets portions, each containing 263, 84 photos and two classes: permission and non-permission: **Fig. 6** and **Fig.7** display photos from the real dataset.

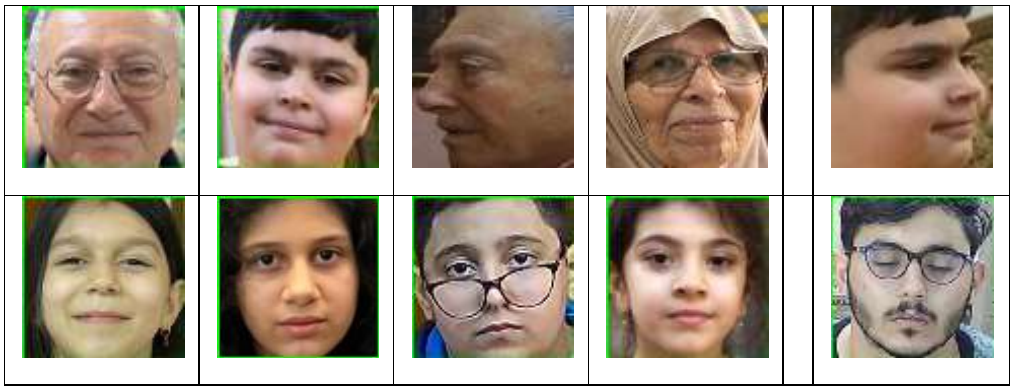


**Figure 6.** Real data set example images representing the permission class.

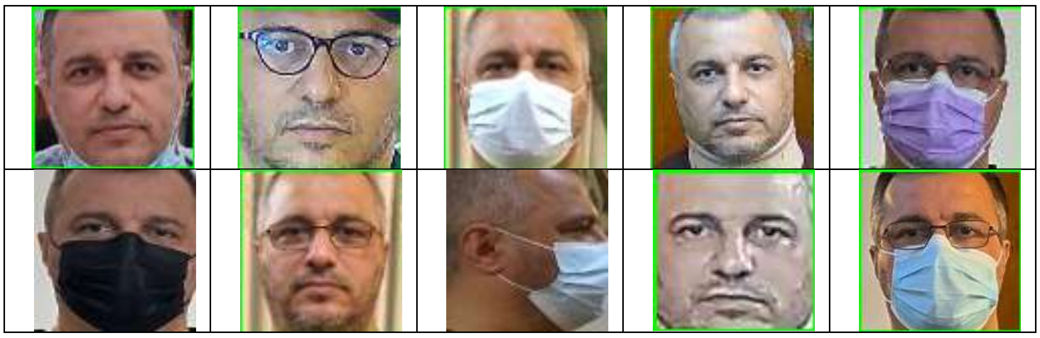
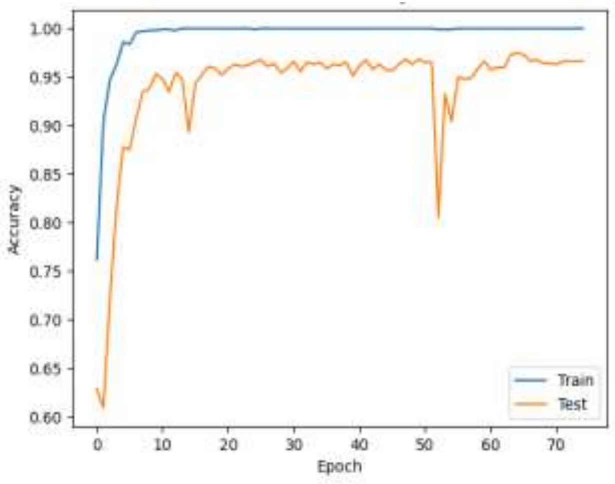


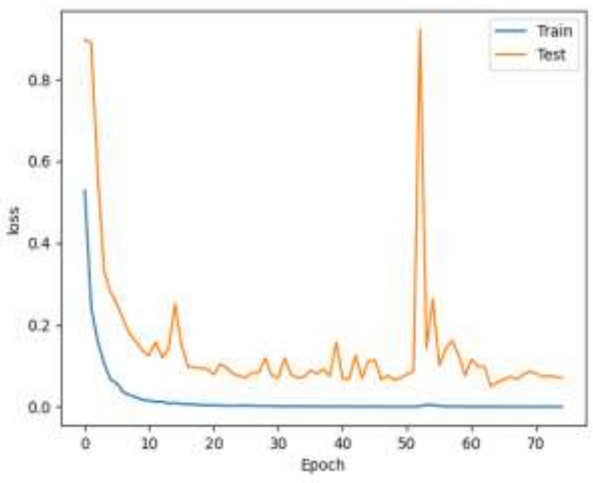
**Figure 7.** Real data set example images representing the non-permission class.

### 3.2 Training

All performance indicators were established for the model, which includes offline augmentation of the real data set. The binary cross-entropy loss across the training and validation sets was evaluated using a batch size of 32. 75 epochs were used throughout the training procedure, and a feature vector location with 300 values was selected. **Fig. 8a** compares the results of offline augmentation for training and validation data accuracy. With offline augmentation, the estimates of binary cross-entropy loss for training and validation data are compared in **Fig. 8b**.



(a)



(b)

**Figure 8.** The accuracy (a) loss and (b) values of real data set with offline augmentation.

### 3.3 Performance

The following five metrics evaluated the suggested model's performance: recall, validation accuracy, training error, and precision or specificity. The percentage of examples in the negative class that are correctly classified is known as "recall." The validation accuracy is calculated by dividing the total number of cases examined by the number of correct predictions. The training error is calculated by dividing the number of wrong predictions by the total number of occurrences, as shown in (5, 6, 7, 8, 9, and 10). The precision and AUC metrics, as a consequence of the classification, represent the quality and accuracy of the model. For data with offline augmentation, **Table 5.** displays the retrieval measure, representing the capacity to locate all relevant components in the original dataset. The feature vector in use chooses 300 locations.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (5)$$

$$\text{TPR (True positive rate)} = \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (6)$$

$$\text{Validation Acc} = \frac{(\text{TP}+\text{TN})}{(\text{TP} + \text{FP} + \text{TN}+\text{FN})} \quad (7)$$

$$\text{Training error} = \frac{(\text{FP} + \text{FN})}{(\text{TP} + \text{FP} + \text{TN} + \text{FN})} \quad (8)$$

$$\text{Specificity} = \frac{\text{TN}}{\text{TN+FP}} \tag{9}$$

$$\text{FPR (False positive rate)} = 1 - \text{Specificity} = \frac{\text{FP}}{\text{TN+FP}} \tag{10}$$

where "true positive" (TP) indicates that both the model prediction and the class are positive. True Negative (TN) states that both the model's prediction and the class are negative. False Positive (FP): The model was incorrectly classified as being in the negative category. False Negative (FN): The model was classified as positive when it was not. The real negative class (N) is FP + TN, while the real positive class (P) is TP + FN. The validation accuracy value given in **Table 5.** which was improved to 98.81 % when using the offline data augmentation. The validation loss was also improved to reach 0.0176, whereas the AUC value was equal to 1.00.

**Table.5** Confusion matrix for the model with offline augmentation, applied on the original real data set.

| Models proposed applied to a 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| With Offline augmentation | 23 | 60 | 0 | 1 | 1 | 0.9583 | 0 | 1 | 98.81% | 0.0176 | 0.0064 |

## 3.4 Test Prediction System Results

As a test, we take captured images from distributed fixed cameras located in the important regions of the high-security buildings to capture images of people entering and then upload them to the S3 Amazon Web Service (AWS). From AWS EC2, the full system is running by downloading all the images from S3 and inputting them into one of the two detectors. The first is the Haar cascade detector, and the second is the MTCNN detector, cutting the face region, inputting them to the classifier model, and classifying them as "permission" or "non-permission" based on a built-from-scratch proposed CNN training model for face recognition. After that, the non-permission classes are uploaded to the S3 AWS cloud. The captured images had a window size of 100 x 100 pixels to recognize the regions of a person's face. **Fig. 9** and **Fig. 10** show the flow chart of the detector accuracy versus images and versus execution time for two detectors: Haar cascade and MTCNN.

**Fig. 11** shows the overall test prediction system. The difference in accuracy and execution time given in **Table 6.** when the Haar cascade detector and the MTCNN detector are applied to the real data images captured by our camera as a test prediction downloaded from AWS S3 to AWS EC2.

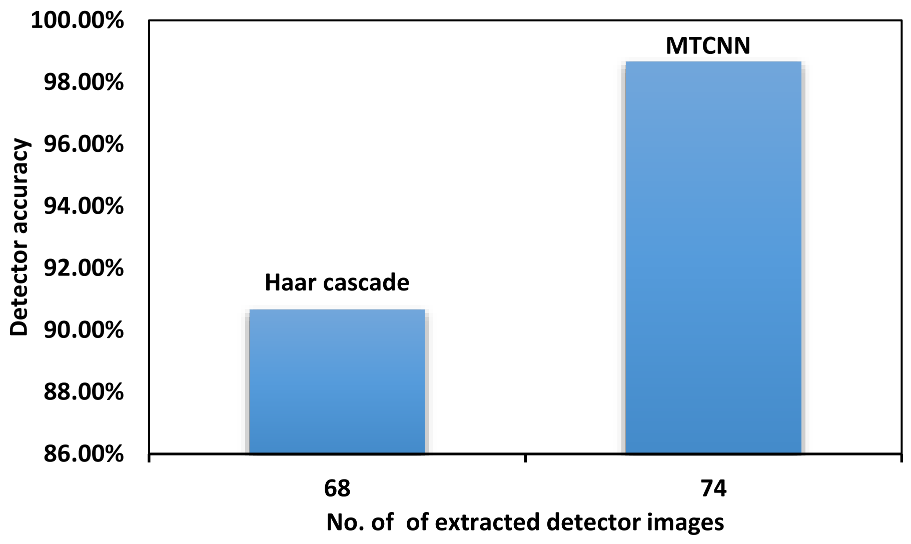


**Figure 9.** Detector accuracy versus no. of extracted detector images

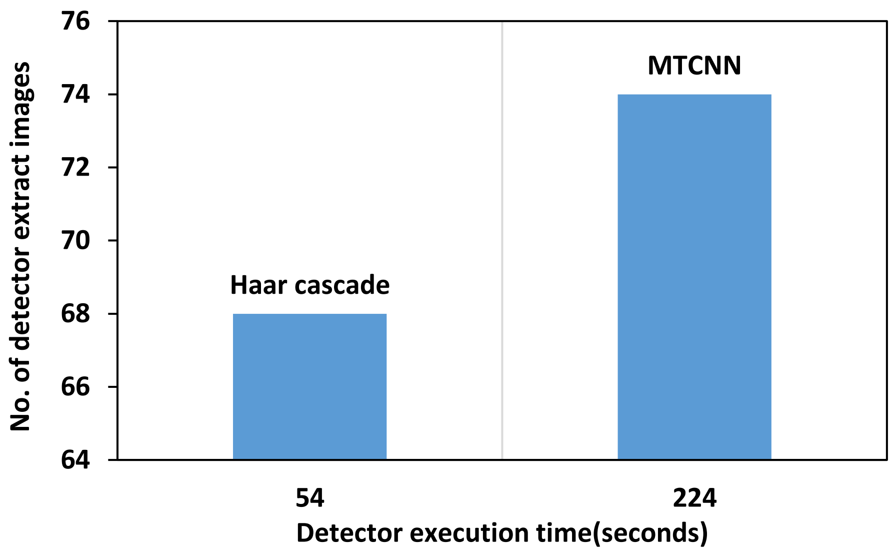


**Figure 10.** Number of detectors extracted images versus detector execution time

**Table 6.** The difference between the two detectors in accuracy and time on the AWS EC2 cloud-free tier

| Detector Type | number of images entered into the detector | The number of detector images extracted | Detector Accuracy | Detector execution time (seconds) |
|---|---|---|---|---|
| Haar cascade | 75 | 68 | 90.66 % | 54 |
| MTCNN | 75 | 74 | 98.66 % | 224 |

**Table 7** illustrates the test prediction for 75 images uploaded to the AWS S3 cloud from our camera system. When saved on the cloud, it is downloaded to the EC2 AWS cloud, inputting the two types of detectors, the Haar cascade detector and the MTCNN detector.

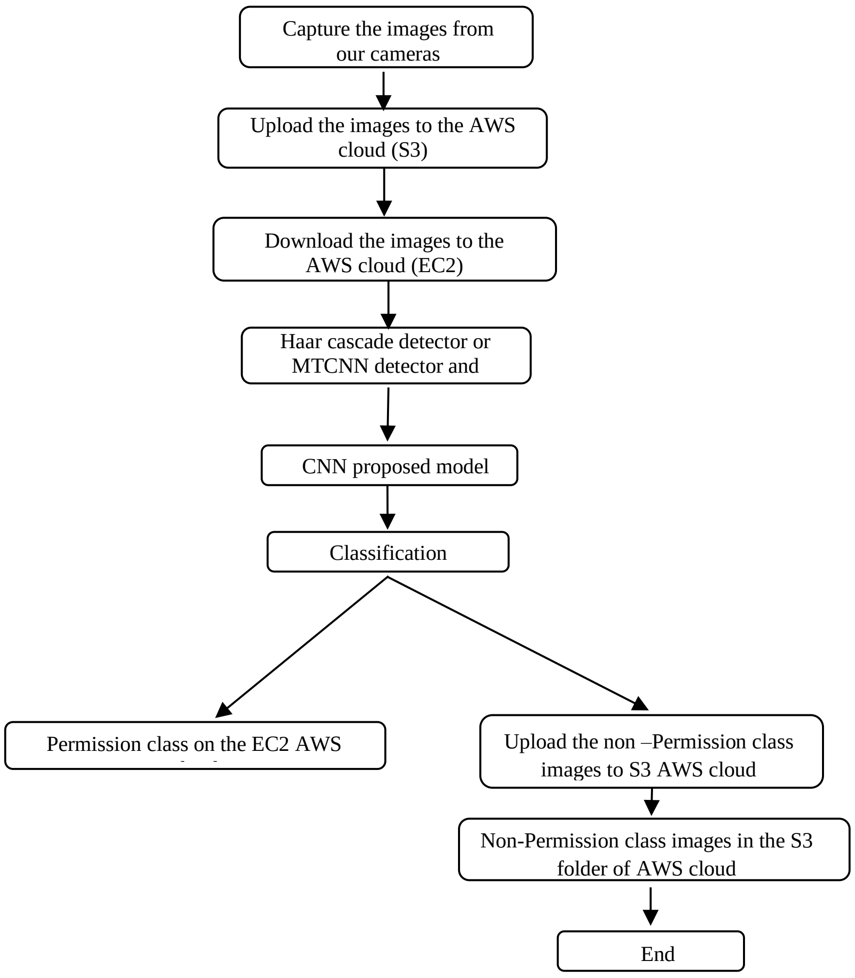

Capture the images from our cameras
Upload the images to the AWS cloud (S3)
Download the images to the AWS cloud (EC2)
Haar cascade detector or MTCNN detector and
CNN proposed model
Classification
Permission class on the EC2 AWS
Upload the non –Permission class images to S3 AWS cloud
Non-Permission class images in the S3 folder of AWS cloud
End

**Figure 11.** The overall test prediction system from S3 to EC2 to S3 (AWS cloud).

**Table 7.** Execution time from S3 to EC2 to S3 AWS cloud-free tier

| Overall system | No. of test images | Total sizes (MB) | The number of detector images extracted | Overall system execution time (seconds) |
|---|---|---|---|---|
| With Haar cascade detector | 75 | 19.7 | 68 | 104 |
| With MTCNN detector | 75 | | 74 | 250 |

The number of extracted images is 68 for the Haar cascade detector and 74 for the MTCNN detector. Execution times using the AWS free tier for the overall system using Haar cascade or MTCNN detectors for 75 images are 104 and 250 seconds, respectively.

**Figs. 12 and 13** show the flow chart using two detectors in the free tier of the AWS cloud: the number of images and the images' total size versus execution time, respectively.

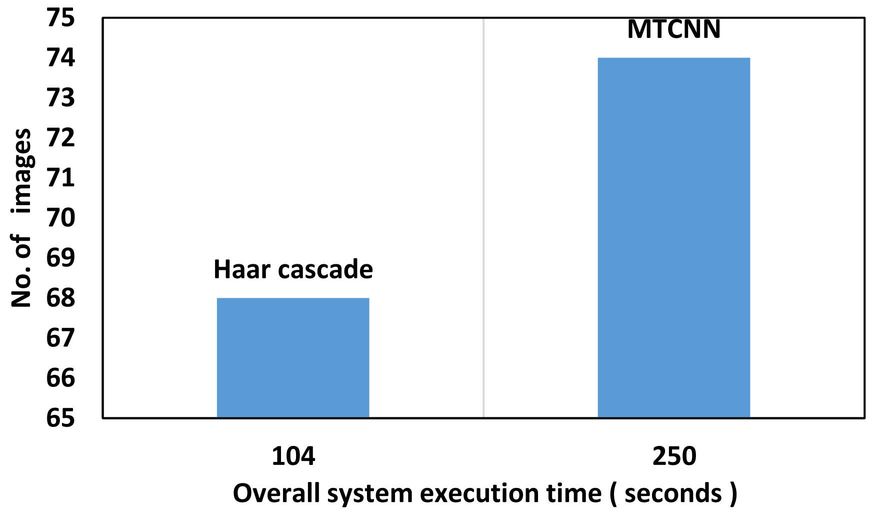


**Figure 12.** No. of images versus execution time

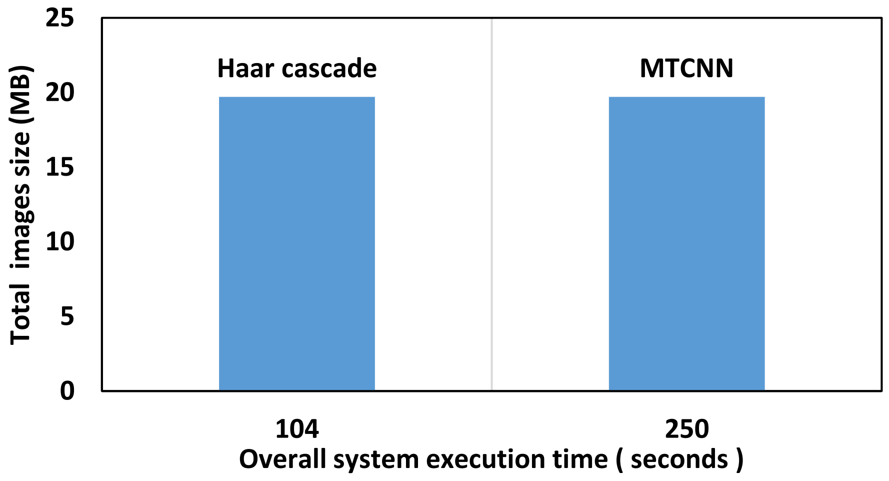


**Figure 13.** Total images size versus the execution time in AWS free tier

**Table 8** illustrates the execution time to run the overall system as a test prediction for one image from S3 to EC2 to S3 AWS cloud leased tier and describes its performance of it in **Table 3.** Also, show the total image size for the test image.

**Table 8.** Execution time for one test image from S3 to EC2 to S3 AWS cloud paid tier

| No. of test images | Size (KB) | Total images size (KB) | Overall system execution time with Haar cascade detector (seconds) | Overall system execution time with MTCNN detector |
|---|---|---|---|---|
| 1 | 200.5 | 200.5 | 3 | 7 |

**Figs. 14 and 15** illustrate the flow chart of all systems being run over for testing one image by using the Haar cascade detector or MTCNN detector for the number of images and the total image size versus the execution time.

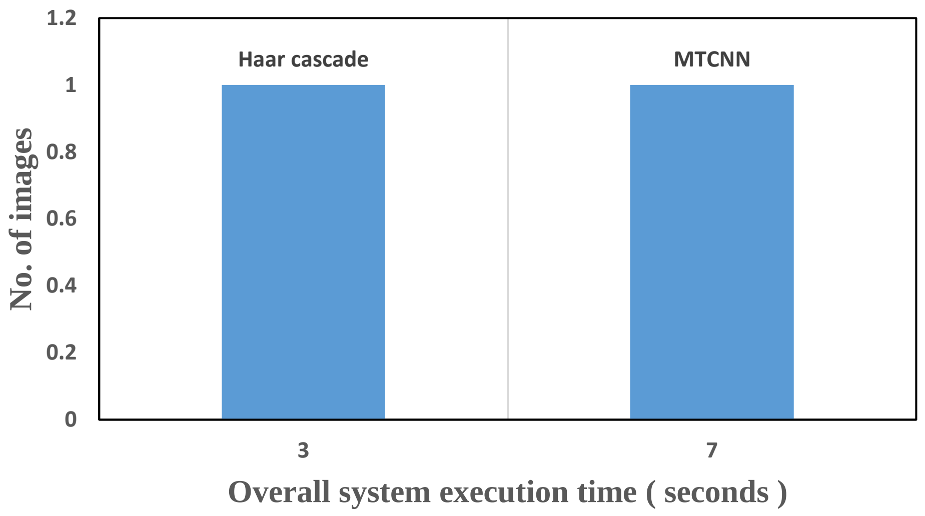


**Figure 14**. One image test versus execution time for running the overall system.

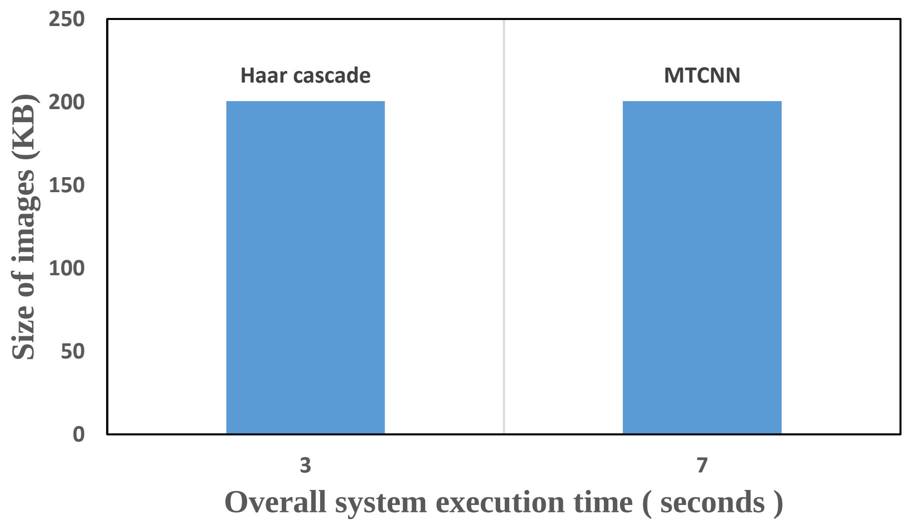


**Figure 15.** The total image size with execution time for testing one image.

**Table 9** illustrates the execution time to run the overall system as a test prediction for two images and the total image size from S3 to EC2 to S3 AWS cloud paid tier.
**Figs. 16 and 17** illustrate the flow chart of all systems that test two images using the Haar cascade detector or the MTCNN detector. They show the number of images and the total image size versus the execution time. **Table 10** illustrates the execution time to run the overall system as a test prediction for four images and the total image size from S3 to EC2 to AWS S3 cloud tier.

**Table 9.** Execution time for two test images from S3 to EC2 to S3 AWS cloud paid tier

| No. of test images | Size(KB) | Total images size (KB) | Overall system execution time with Haar cascade detector (seconds) | Overall system execution time with MTCNN detector (seconds) |
|---|---|---|---|---|
| 1 | 200.5 | 395.5 | 4 | 9 |
| 2 | 195 | | | |

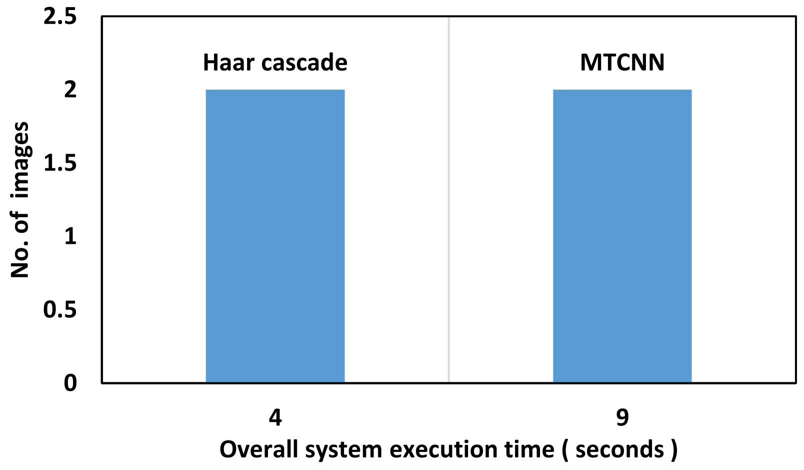


**Figure 16.** Two images test versus execution time for the overall run system.

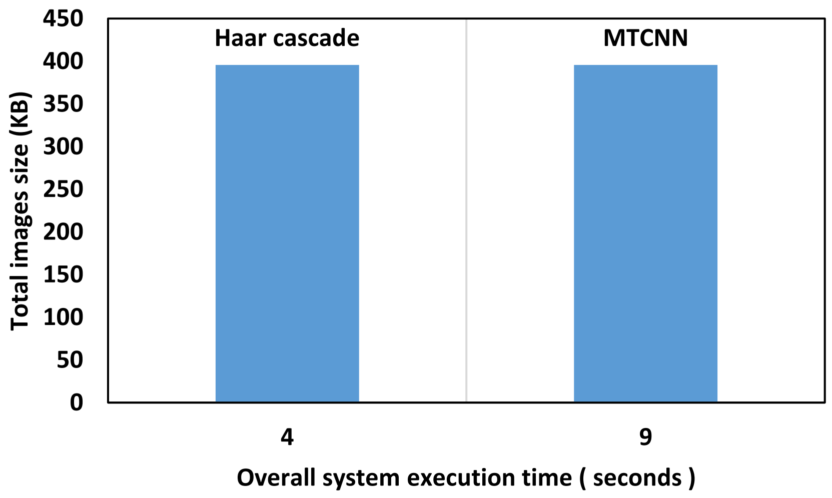


**Figure 17.** The total image size with execution time for testing two images.

**Table 10.** Execution time for four test images from S3 to EC2 to AWS S3 cloud paid tier

| No. of test images | Size(KB) | Total images size (KB) | Overall system execution time with Haar cascade detector (seconds) | Overall system execution time with MTCNN detector |
|---|---|---|---|---|
| 1 | 200.5 | 764.1 | 6 | 10 |
| 2 | 195 | | | |
| 3 | 179.5 | | | |
| 4 | 189.1 | | | |

**Figs. 18 and 19** illustrated the flow chart of run-over all systems for test four images by using the Haar cascade detector or MTCNN detector to show the no. of images and the total images size versus the execution time.

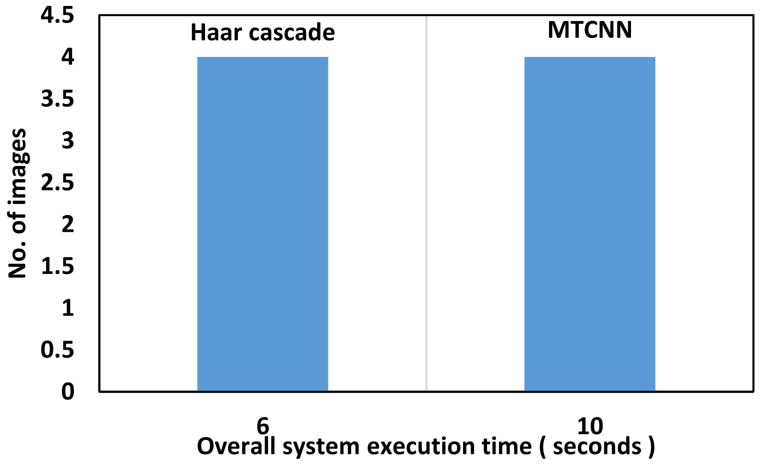


**Figure 18.** Four images test versus execution time for running the overall system

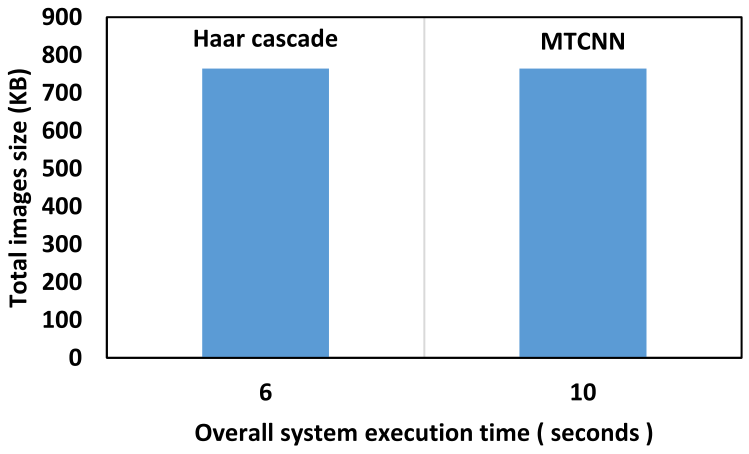


**Figure 19.** The total image size with execution time for test four images.

**Table 11** illustrates the execution time to run the overall system as a test prediction for six images, and the total image size from S3 to EC2 to S3 AWS cloud leased tier.

**Table 11.** Execution time for six test images from S3 to EC2 to S3 AWS cloud Leased tier

| No. of test images | Size (KB) | Total images size (KB) | Overall system execution time with Haar cascade detector (seconds) | Overall system execution time with MTCNN detector (seconds) |
|---|---|---|---|---|
| 1 | 200.5 | 1055.4 | 8 | 24 |
| 2 | 195.0 | | | |
| 3 | 100.3 | | | |
| 4 | 179.5 | | | |
| 5 | 189.1 | | | |
| 6 | 191.0 | | | |

**Figs. 20** and **21** illustrate the flow chart of all systems tested for four images by using the Haar cascade detector or the MTCNN detector.

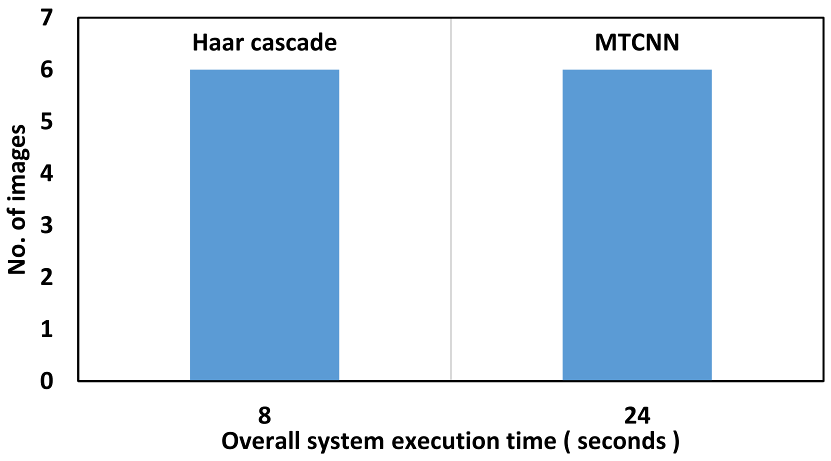


**Figure 20.** Six images test versus execution time for the overall run system.

The number of images and the total image size versus the execution time are presented. **Table 12.** illustrates the execution time to run the overall system as a test prediction for eight images, and the total image size from S3 to EC2 to S3 AWS cloud leased tier. **Figs. 22 and 23** illustrate the flow chart of all systems tested for eight images using the Haar cascade detector or the MTCNN detector. They show the number of images and the total image size versus the execution time.

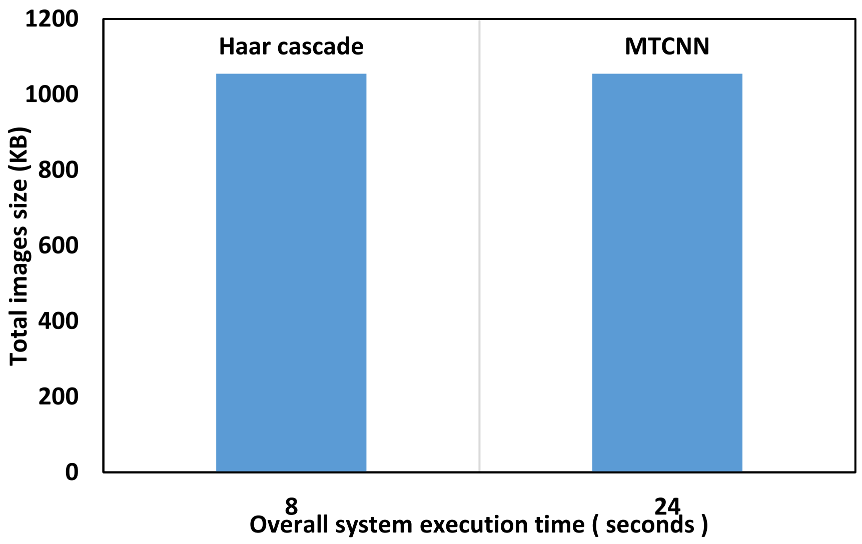


**Figure 21.** The total image size with execution time for test six images

**Table 12.** Execution time for eight test images from S3 to EC2 to AWS S3 cloud paid tier.

| No. of test images | Size(KB) | Total images size (KB) | Overall system execution time with Haar cascade detector (seconds) | Overall system execution time with MTCNN detector (seconds) |
|---|---|---|---|---|
| 1 | 202.0 | 1364.9 | 9 | 32 |
| 2 | 197.5 | | | |
| 3 | 88.3 | | | |
| 4 | 120.0 | | | |
| 5 | 179.5 | | | |
| 6 | 189.1 | | | |
| 7 | 191.0 | | | |
| 8 | 197.3 | | | |

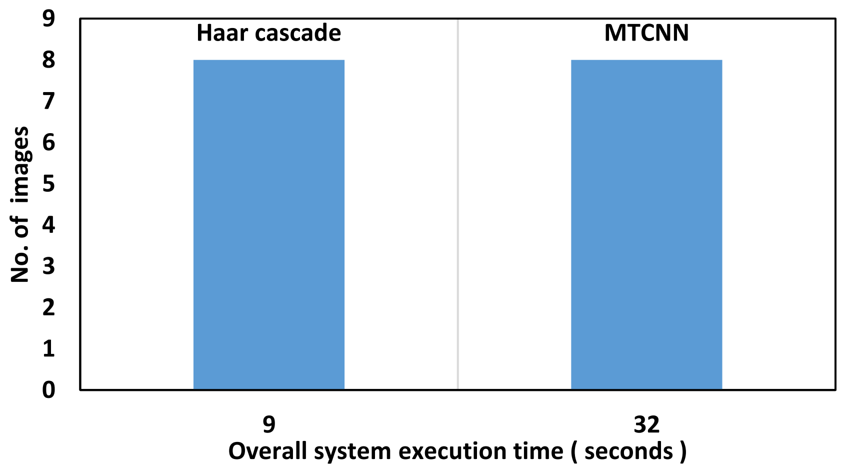


**Figure 22.** Test eight images versus execution time for all systems.

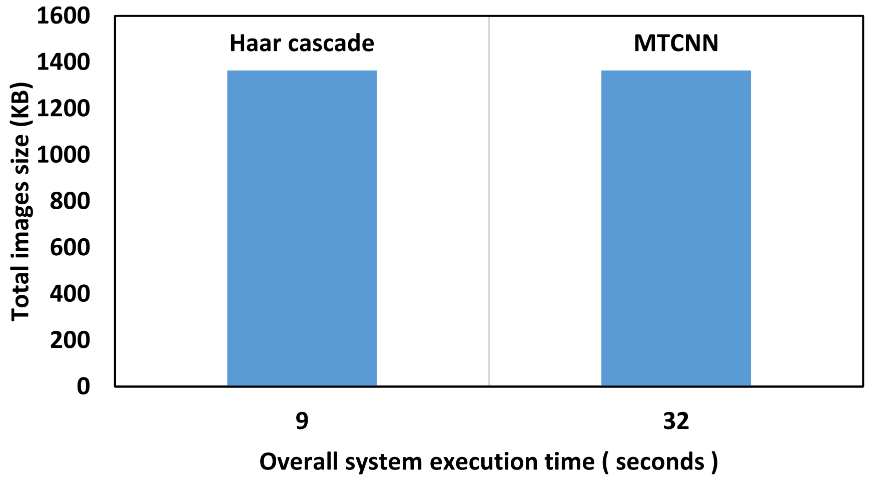


**Figure 23.** The total image size with execution time for test eight images.

**Table 13** illustrates the execution time to run the overall system as a test prediction for ten images and the total image size from S3 to EC2 to AWS S3 cloud paid tier.

**Table 13.** Execution time for eight test images from S3 to EC2 to AWS S3 cloud paid tier

| No. of test images | Size(KB) | Total images size (KB) | Overall system execution time with Haar cascade detector (seconds) | Overall system execution time with MTCNN detector (seconds) |
|---|---|---|---|---|
| 1 | 102.8 | 1720.2 | 11 | 40 |
| 2 | 100.3 | | | |
| 3 | 177.6 | | | |
| 4 | 179.5 | | | |
| 5 | 189.1 | | | |
| 6 | 191.0 | | | |
| 7 | 177.7 | | | |
| 8 | 185.1 | | | |
| 9 | 197.3 | | | |
| 10 | 219.8 | | | |

**Figs. 24 and 25** illustrate the flow chart of all systems tested for ten images using the Haar cascade detector or the MTCNN detector. They show the number of images and the total image size versus the execution time.

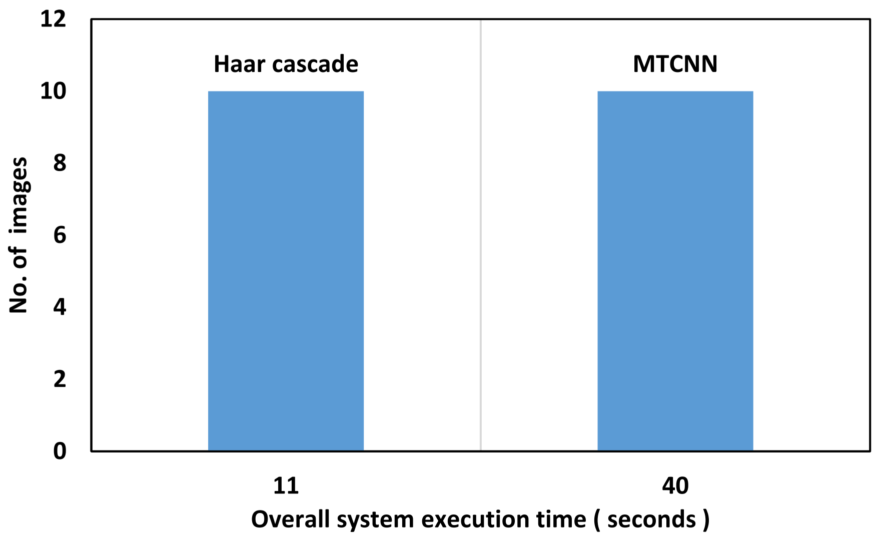


**Figure 24.** Ten images test versus execution time for running overall system

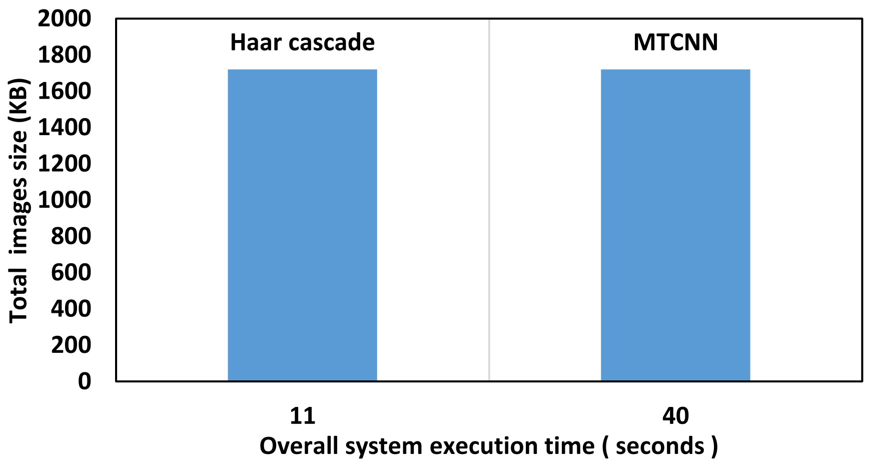


**Figure 25.** The total image size with execution time for test eight images

**Table 14** illustrates the execution time to run the overall system as a test prediction for 75 images and the total image size from S3 to EC2 to AWS S3 cloud paid tier.

**Table 14.** Execution time for 75 test images from S3 to EC2 to AWS S3 cloud paid tier

| Overall system | No. of test images | Total sizes (M Bytes) | The number of detector images extracted | Overall system execution time (seconds) |
|---|---|---|---|---|
| With Haar cascade detector | 75 | 19.7 | 68 | 68 |
| With MTCNN detector | 75 |  | 74 | 230 |

**Figs. 26 and 27** illustrate the flow chart of all systems tested for 75 images using the Haar cascade detector or the MTCNN detector. They show the number of images and the total image size versus the execution time.

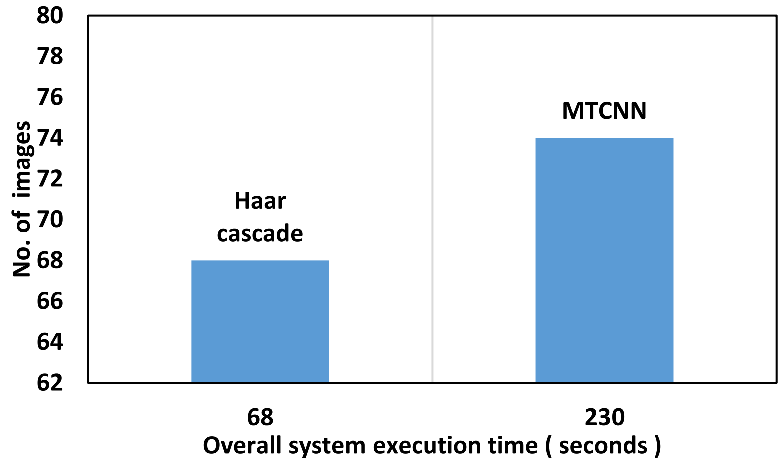


**Figure 26.** Test 75 images versus execution time for running the overall system

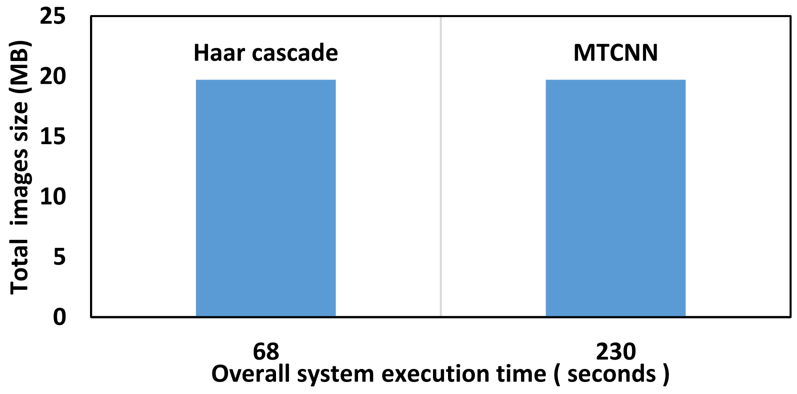


**Figure 27.** The total image size with execution time for test eight images

**Table 15.** Compares the free-tier and paid-tier cloud for 75 images as a test prediction by applying different detectors to the proposed model classification on the AWS EC2 cloud.

**Table 15.** Compare the result of Execution time for 75 test images from S3 to EC2 to S3 AWS cloud free tier and Leased tier

| Overall system | No. of test images | Total sizes (M Bytes) | The number of detector images extracted | Overall system execution time (seconds) free tier AWS cloud | Overall system execution time (seconds) leased tier AWS cloud |
|---|---|---|---|---|---|
| With Haar cascade detector | 75 | 19.7 | 68 | 104 | 68 |
| With MTCNN detector | 75 | | 74 | 250 | 230 |

**Figs. 28 and 29** illustrate the flow chart of overall systems run for 75 images by using the Haar cascade detector using two AWS cloud performance-free tiers and paying for them.

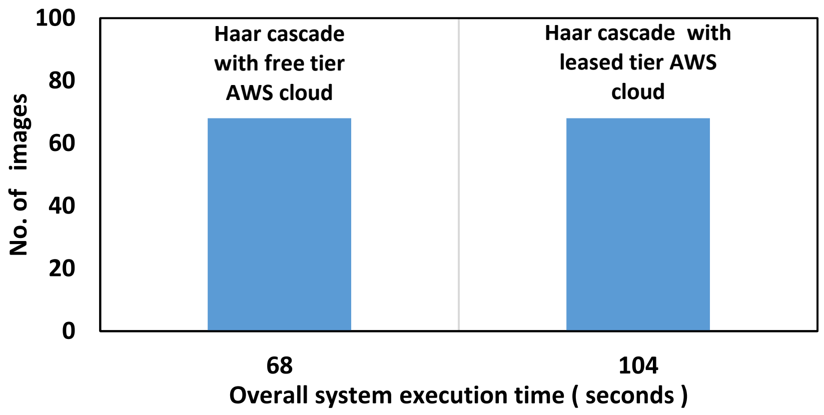


**Figure 28.** Overall system execution time by using Haar cascade detector of different cloud performance

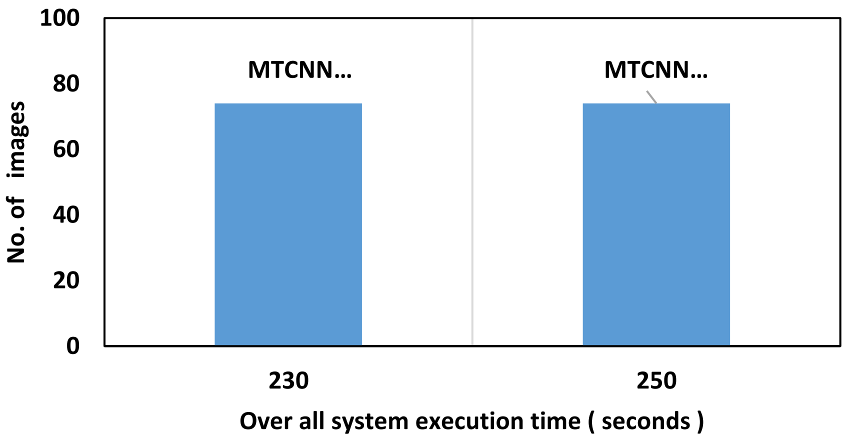
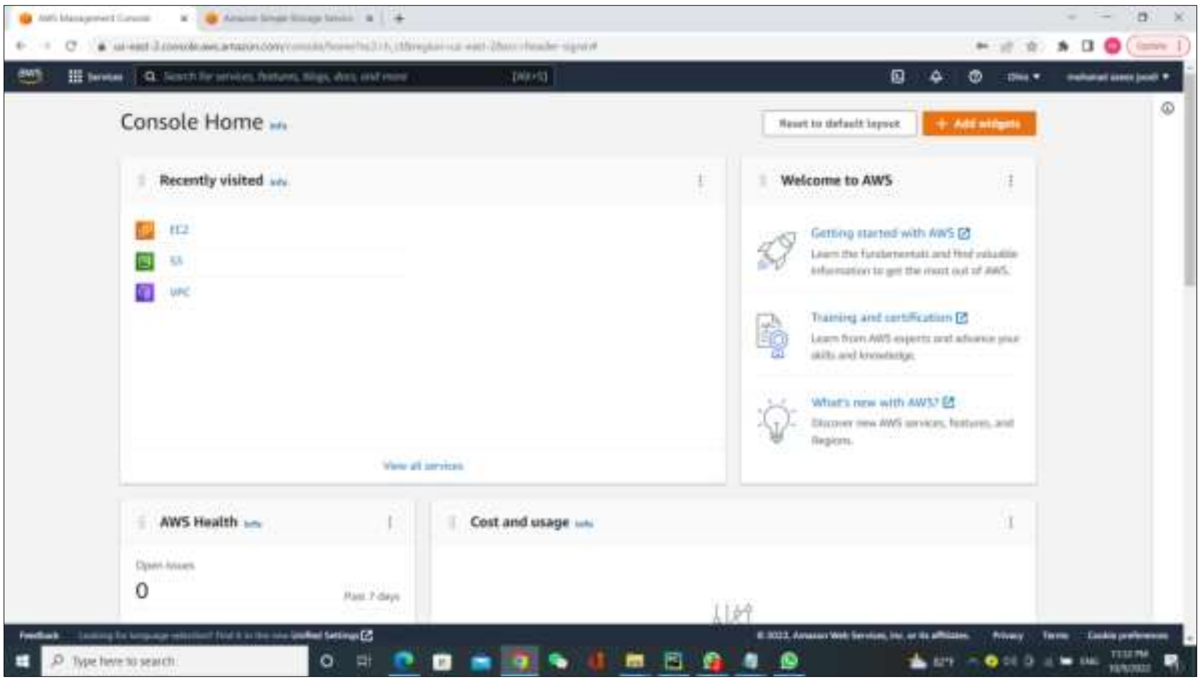


**Figure 29.** Overall system execution time by using MTCNN detector of different cloud performance

**Figs. 30 to 33** represent the web pages of our AWS account, an AWS EC2 instance, an AWS S3 camera, and a not-permitted class upload from AWS EC2. **Fig. 34** shows the non-permission class on the AWS S3 uploaded from AWS EC2 using the Haar cascade detector, and **Fig. 35** shows the permission class saved on the AWS EC2 using the Haar cascade detector. **Fig. 36** and **Fig. 37** show the non-permission and permission classes, respectively, using the MTCNN detector.


Console Home

Recently visited: EC2, S3, VPC

Welcome to AWS

Getting started with AWS — Learn the fundamentals and find valuable information to get the most out of AWS.

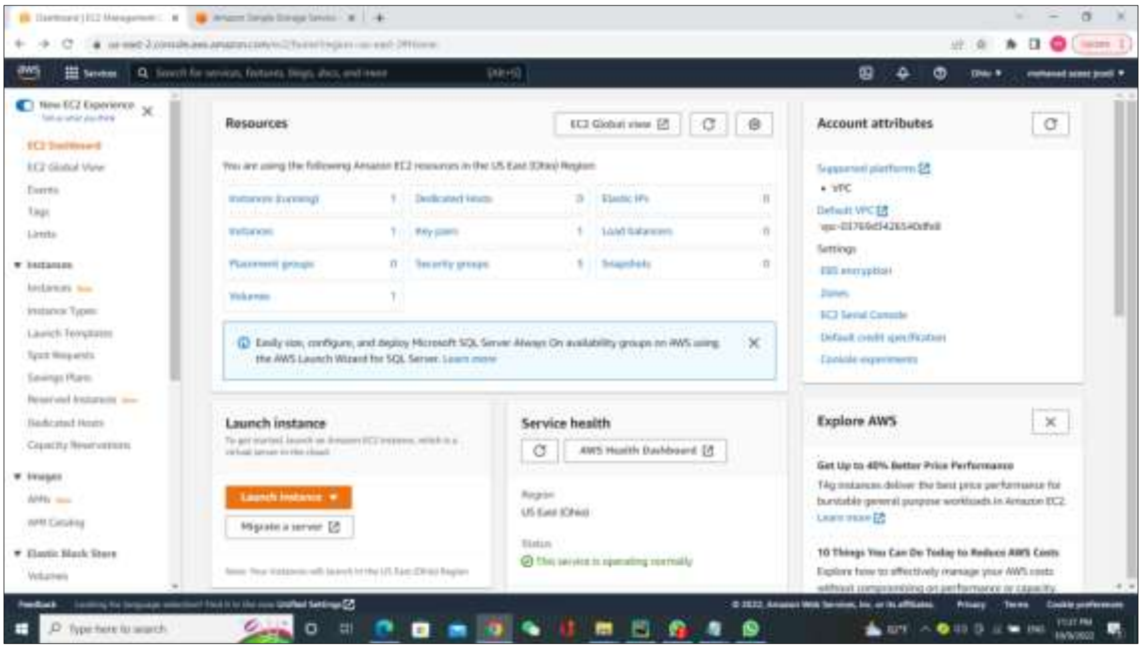
Training and certification — Learn from AWS experts and advance your skills and knowledge.

What's new with AWS? — Discover new AWS services, features, and Regions.

AWS Health — Open issues 0 — Past 7 days

Cost and usage

**Figure 30.** AWS our account


Resources

You are using the following Amazon EC2 resources in the US East (Ohio) Region

| | | | | | |
|---|---|---|---|---|---|
| Instances (running) | 1 | Dedicated Hosts | 0 | Elastic IPs | 0 |
| Instances | 1 | Key pairs | 1 | Load balancers | 0 |
| Placement groups | 0 | Security groups | 1 | Snapshots | 0 |
| Volumes | 1 | | | | |

Easily size, configure, and deploy Microsoft SQL Server Always On availability groups on AWS using the AWS Launch Wizard for SQL Server. Learn more

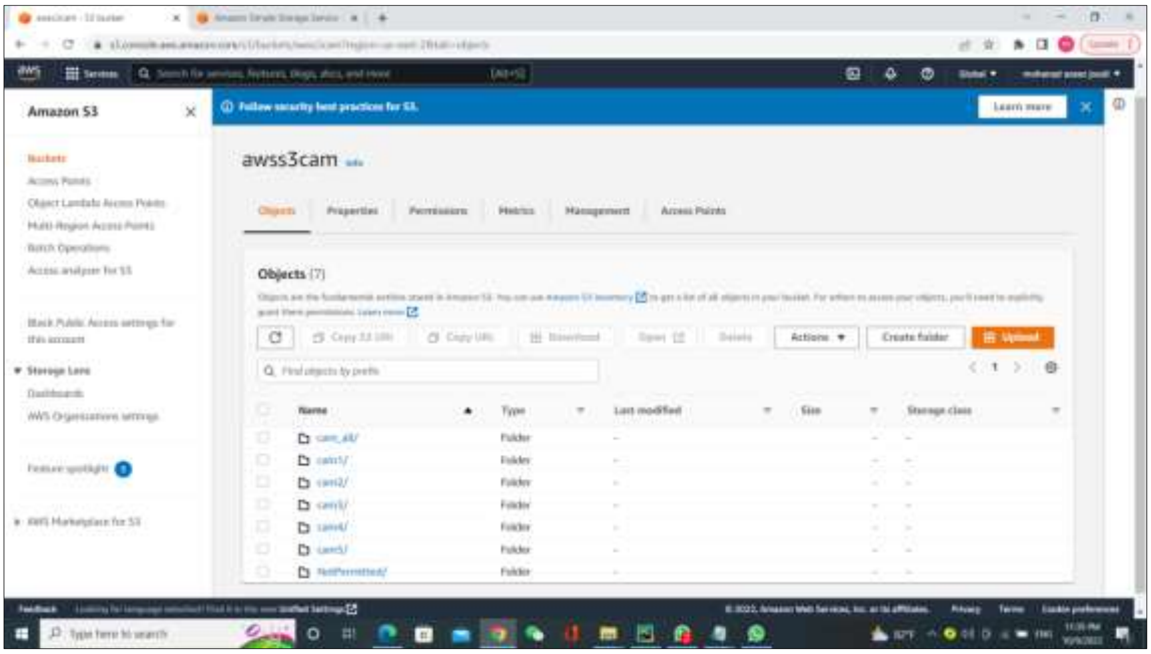
Account attributes: Supported platforms • VPC; Default VPC vpc-0376d42654 0dfe8; Settings: EBS encryption, Zones, EC2 Serial Console, Default credit specification, Console experiments

Launch instance — To get started, launch an Amazon EC2 instance, which is a virtual server in the cloud.

Service health — Region: US East (Ohio); Status: This service is operating normally

Explore AWS — Get Up to 40% Better Price Performance; 10 Things You Can Do Today to Reduce AWS Costs

**Figure 31.** AWS EC2 our instance



Amazon S3

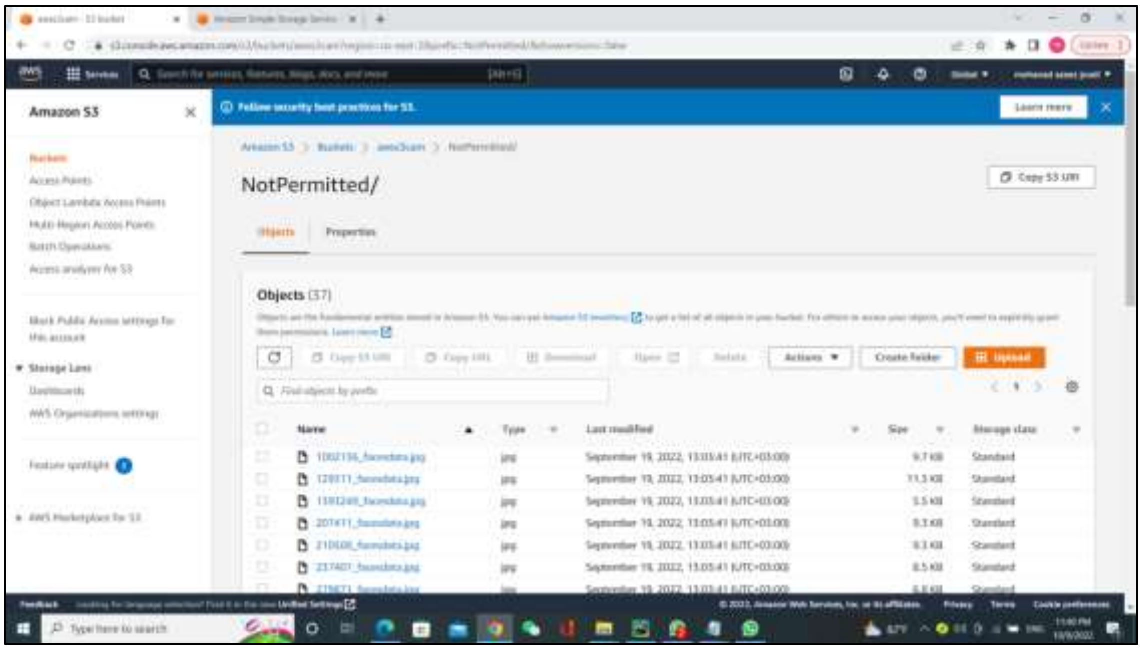
Follow security best practices for S3.

awss3cam

Objects (7)

| Name | Type | Last modified | Size | Storage class |
|---|---|---|---|---|
| cam_all/ | Folder | - | - | - |
| cam1/ | Folder | - | - | - |
| cam2/ | Folder | - | - | - |
| cam3/ | Folder | - | - | - |
| cam4/ | Folder | - | - | - |
| cam5/ | Folder | - | - | - |
| NotPermitted/ | Folder | - | - | - |

**Figure. 32.** AWS S3cam web page uploaded images captured from our cameras



Amazon S3 > Buckets > awss3cam > NotPermitted/

NotPermitted/

Objects (37)

| Name | Type | Last modified | Size | Storage class |
|---|---|---|---|---|
| 1002136_facedata.jpg | jpg | September 19, 2022, 13:03:41 (UTC+03:00) | 9.7 KB | Standard |
| 128311_facedata.jpg | jpg | September 19, 2022, 13:03:41 (UTC+03:00) | 11.5 KB | Standard |
| 1393249_facedata.jpg | jpg | September 19, 2022, 13:03:41 (UTC+03:00) | 5.5 KB | Standard |
| 207411_facedata.jpg | jpg | September 19, 2022, 13:03:41 (UTC+03:00) | 9.3 KB | Standard |
| 210508_facedata.jpg | jpg | September 19, 2022, 13:03:41 (UTC+03:00) | 9.3 KB | Standard |
| 237407_facedata.jpg | jpg | September 19, 2022, 13:03:41 (UTC+03:00) | 8.5 KB | Standard |
| 279871_facedata.jpg | jpg | September 19, 2022, 13:03:41 (UTC+03:00) | 6.8 KB | Standard |

**Figure 33.** Non-permitted class web page uploaded images from AWS EC2 instance

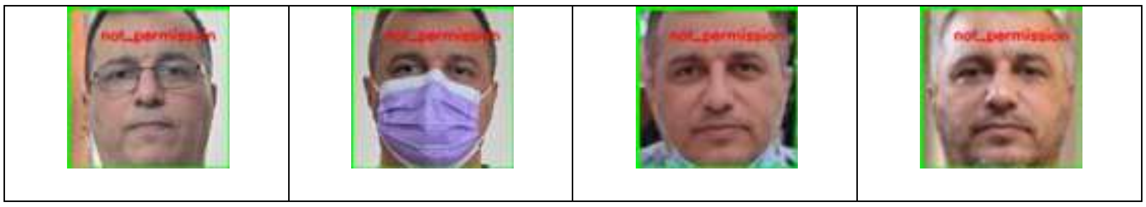


**Figure 34.** Non-permission class by using haar cascade detector only frontal faces

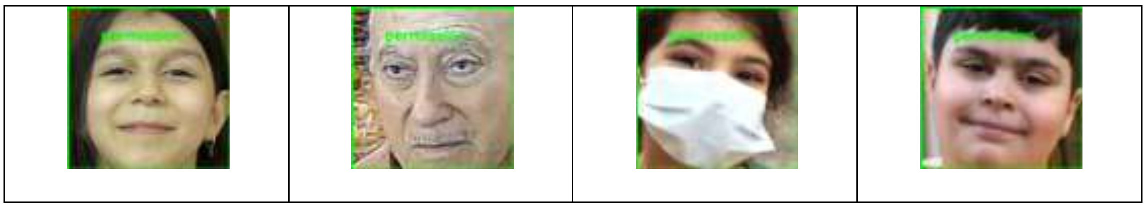


**Figure 35.** Permission class by using haar cascade detector only frontal faces

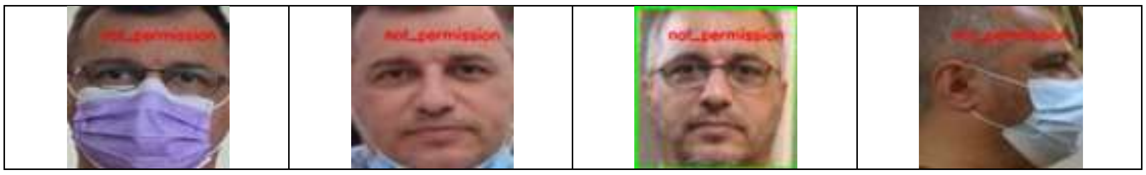


**Figure 36**. Non- permission class by using MTCNN detector (frontal and non-frontal faces)

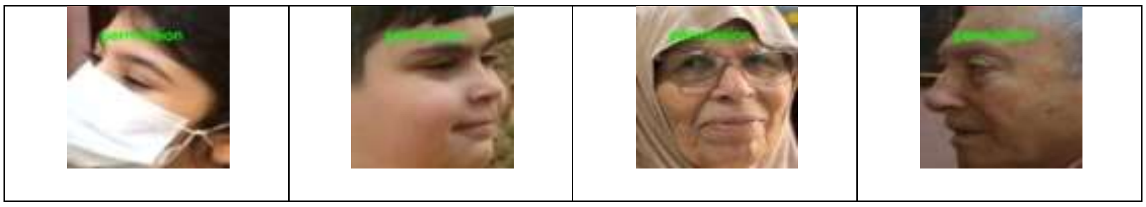


**Figure 37.** Permission class by using MTCNN detector (frontal and non-frontal faces)

## 4. CONCLUSION

This work presented a Haar cascade detector, the MTCNN detector, within a proposed CNN cloud-based system with two classes, the permission class and the non-permission class, applied to a real data set collected by fixed cameras. The main idea is to enable the system to distinguish if a person has permission not to enter the building by using AWS cloud services when applied to RGB images with a size of 100x100 pixels. Compare the accuracy and execution time of all systems using two detectors to detect faces on the captured images: the Haar Cascade only detects frontal faces, while the MTCNN detects both frontal and non-frontal faces. AWS S3 cloud and an AWS EC2 instance are used to save many images captured from cameras and image analysis. Compare the performance of the AWS EC2 free tier, and the AWS EC2 paid tier, and make the execution time shorter compared to the EC2 free tier for testing one image to ten images using two detectors. Running and accessing the programmed software from any location by logging into the AWS account and accessing the non-permission class from any location

## REFERENCES

Asmara, R.A., Ridwan, M., and Budiprasetyo, G., 2021. Haar Cascade and Convolutional Neural Network Face Detection in Client-Side for Cloud Computing Face Recognition. *2021 International Conference on Electrical and Information Technology (IEIT)*, pp. 1-5. doi:10.1109/LGRS.2018.2799232.

Babbar, S., Dewan, N., Shangle, K., Kulshrestha, S., and Patel, S., 2019. Cross-Age Face Recognition using Deep Residual Networks., *2019 Fifth International Conference on Image Information Processing, India, IEEE Xplore*, pp. 257-262. doi:10.1109/ICIIP47207.2019.8985765

Chen, Z., and Liu, Y., 2020. Application of Face Recognition in Smart Hotels., *2nd IEEE Eurasia Conference on IOT, Communication and Engineering 2020*, pp. 180-182. doi:10.1109/ECICE50847.2020.9302014.

Dersingh, A., Charanyananda, S., Chaiyaprom, A., Domsrifah, N., and Liwsakphaiboon, S., 2016 Customer Recognition and Counting by Cloud Computing, pp. 1-4. doi:10.1109/ITC-CSCC.2019.8793318.

Feng, X., Liu, Y., and Zhang, C., 2021. A Class Attendance System Based on Cloud Face Recognition for Multi-users., *2021 7th International Conference on Computer and Communications (ICCC)*, pp. 927-931. doi:10.1109/ICCC54389.2021.9674578.

Gregorius Rafael, H.K., and Tasripan 2020. The Utilization of Cloud Computing for Facial Expression Recognition using Amazon Web Services., 2020 International Conference on Computer Engineering, Network and Intelligent Multimedia, pp. 366-370. doi:10.1109/CENIM51130.2020.9297974.

Hughes, L., Schmitt, M., Mou, L., Wang, Y., and Zhu, X., 2018. Identifying Corresponding Patches in SAR and Optical Images With a Pseudo-Siamese CNN. *IEEE Geoscience and Remote Sensing Letters, 15(5),* pp., 784-788. doi:10.1109/LGRS.2018.2799232.

Jabbar, S.Q., and Kadhim, D. J., 2021. A Proposed Adaptive Bitrate Scheme Based on Bandwidth Prediction Algorithm for Smoothly Video Streaming. *Journal of Engineering 27(1),* pp., 112-129. doi:10.31026/j.eng.2021.1.08.

Jabbar, S.Q., Kadhim, D. J., and Yu, L, 2018. Developing a video buffer framework for video streaming in cellular networks. *Wireless Communications Mobile Computing* 2018. doi:10.1155/2018/6584845

Jha, S., Arora, M., Sharma, Y., Anand, A., and Sharma, D., 2022. Comparative Analysis of Cloud Computing Based Face Recognition Services. *2022 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COM-IT-CON)*, pp. 567-572. doi:10.1109/COM-IT-CON54601.2022.9850927.

Joodi, M.A., Saleh M. H., Kadhim, D. J., 2023. Increasing validation accuracy of a face mask detection by new deep learning model-based classification. *Indonesian Journal of Electrical Engineering and Computer Science 29,* pp., 304-314. doi: 10.11591/ijeecs.v29.i1.pp304-314.

Kadhim, D.J., Saleh, M. H. , Abou-Loukh, S. J., 2023. Evaluation of massive multiple-input multiple-output communication performance under a proposed improved minimum mean squared error precoding, presented at the precoding (PIMPIAES International Journal of Artificial Intelligence (IJ-AI). *IAES International Journal of Artificial Intelligence (IJ-AI) 12(2),* pp., 984-994. doi:10.11591/ijai.v12.i2.pp984-994.

Lin, F.-C., Ngo, H.-H. and Dow, C.-R., 2020. A cloud-based face video retrieval system with deep learning. *The Journal of Supercomputing, 76(11),* pp.,8473-8493. doi:10.1007/s11227-019-03123-x

Mehra, M., Saai, V., Chowdhury, P., and Dsouza, E., 2020. Home Security System using IOT and AWS Cloud Services. *IEEE Xplore*IEEE Xplore, pp. 1-6. doi: 10.1109/ICAC347590.2019.9089839.

Masud, M., Muhammad, G., Alhumyani, H., Alshamrani, S.S., Cheikhrouhou, O., Ibrahim, S., and Hossain, M.S., 2020. Deep learning-based intelligent face recognition in IoT-cloud environment. *Computer Communications, 152*, pp., 215-222. doi:10.1016/j.comcom.2020.01.050.

Neha,N.S., Amita, K., and Enakshi, K.S., 2016. Face Recognition Using Cloud Hopfield Neural Network. *IEEE WiSPNET 2016 conference*, pp. 416--419. doi: 10.1109/WiSPNET.2016.7566167.

Pathak, A.R., Pandey, M. , Rautaray, S., 2018. Application of Deep Learning for Object Detection. *Procedia computer science*, *132 (2018),* pp.,1706-1717. doi: 10.1016/j.procs.2018.05.144.

Pattnaik, P.a.M., and Kumar, K., 2020. AI-Based Techniques for Real-Time Face Recognition-based Attendance System- A comparative Study. Fourth International Conference on Electronics, Communication and Aerospace Technology (ICECA-2020), IEEE Xplore, pp. 1034-1039. doi: 10.1109/ICECA49313.2020.9297643.

Saad, S.Z., and Saleh, M. H., 2018. Seismic attributes selection and porosity prediction using modified artificial immune network algorithm. *Journal of Engineering Science Technology, 13(3)*, pp. 755-765. doi: 10.13140/RG.2.2.23437.38880.

Sharma, D., Sharma, H., and Panchal, D., 2020a. Automatic Office Environment System for Employees Using IoT and Computer Vision. *2020 IEEE 17th India Council International Conference (INDICON)*, pp. 1-6. doi: 10.1109/INDICON49873.2020.9342455.

Sharma, H., Sharma, D., Bhatt, K. and Shah, B., 2020b. Facial Emotion Based Review Accumulation System. *2020 IEEE 17th India Council International Conference (INDICON)*, pp. 1-6. doi: 10.1109/INDICON49873.2020.9342467.

Su, P., 2021. Immersive online biometric authentication algorithm for online guiding based on face recognition and cloud-based mobile edge computing. *Distributed and Parallel Databases*, pp.,1-22. doi: 10.1007/s10619-021-07351-0.

Vinh, T.Q., and Anh , N. T. N., 2020. Real-Time Face Mask Detector Using YOLOv3 Algorithm and Haar Cascade Classifier. International Conference on Advanced Computing and Applications (ACOMP), pp. 146-149. doi: 10.1109/ACOMP50827.2020.00029.

Viola, P., and Jones, M., 2001. Rapid object detection using a boosted cascade of simple features. in 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, Hawaii, pp. I-I. doi : 10.1109/CVPR.2001.990517.

Xue, Y., Zhang, H., and Ma, H., 2018. Performance Evaluation of Image and Video Cloud Services. *2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*. pp. 733-741. doi: 10.1109/HPCC/SmartCity/DSS.2018.00126.

Yong, B., Zhang, G., Chen, H., and Zhou, Q., 2016. Intelligent monitor system based on cloud and convolutional neural networks. *The Journal of Supercomputing*, *73(7),* pp., 3260-3276. doi:10.1007/s11227-016-1934-1.

Yu, C., and Pei, H., 2021. Face recognition framework based on effective computing and adversarial neural network and its implementation in machine vision for social robots. *Computers & Electrical Engineering, 92*, pp., 107-128. doi:10.1016/j.compeleceng.2021.107128.

Zhang, K., Zhang, Z., Li, Z., and Qiao, Y., 2016. Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks. *IEEE Signal Processing Letters, 23(10),* pp., 1499-1503 doi:10.1109/LSP.2016.2603342.