Ogtay Ahmadli

AI Facial Recognition System

Metropolia University of Applied Sciences

Bachelor of Engineering

Degree Programme in Electronics

Bachelor's Thesis

29 January 2022

Abstract

Author:	Ogtay Ahmadli
Title:	AI Facial Recognition System
Number of Pages:	43 pages + 1 appendix
Date:	29 January 2022
Degree:	Bachelor of Engineering
Degree Programme:	Degree programme in Electronics
Professional Major:	Electronics
Supervisor:	Matti Fischer, Principal Lecturer

Nowadays, facial recognition is one of the widely used categories of biometric security that distinguish itself by its security and speed from other categories such as fingerprint recognition and eye retina or iris recognition. This technology is mainly used in electronics devices, airport control, banking, health care, marketing, and advertising. This thesis project aimed to build a facial recognition system that could recognize people through the camera and unlock the door locks. Recognized results were sent to the database and could be analyzed by users after the successful login.

The project consists of building a facial recognition, electronics operation, and webpage design for the database. Firstly, machine learning and deep learning algorithms were used to recognize faces. In the second step, AI data is transmitted to the electronics components and sensors to make a smart lock system. Finally, the last step was to design a user interface that requires a login and displays the attendance list according to the database.

The prototype could successfully recognize human faces and activate the electronics components. It has fast performance and could log information about recognized humans in the Google database.

With further advancements, the prototype would implement more extensive algorithms to distinguish the pictures and real faces through a camera. These algorithms would make the prototype faster, secure, and suitable for commercial purposes.

Keywords: Facial Recognition, Artificial Intelligence, Machine Learning, Deep Learning, Neural networks, Computer Vision

Contents

List of Abbreviations

AI:	Artificial Intelligence
ML:	Machine Learning
DL:	Deep Learning
CNN:	Convolutional Neural Networks
ANN:	Artificial Neural Networks
RNN:	Recurrent Neural Networks
2D:	2-dimensional
3D:	3-dimensional
HOG:	Histogram of the gradients
SVM:	Support Vector Machines
CV:	Computer Vision
LED:	Light Emitting Diode
OLED:	Organic Light Emitting Diode

1 Introduction

Facial recognition is an immensely powerful technology that recognizes human faces through the camera based on facial features. Nowadays, this technology exists in electronic devices, industries, airports that perform facial recognition instantly without human intervention. Facial recognition distinguishes itself by its preciseness in terms of data collection and verification. It is also more suitable for identification uses, as other biometric security categories are unique to each person.

The goal of the thesis project was to build a facial recognition system that could recognize people through the camera and unlock the door locks. The prototype would be fixed to the doors and use the camera to operate the whole circuitry. The results would be logged in the google database and analyzed by users after the successful login.

The implementation of the project was accomplished in three steps. Initially, the facial recognition system was built using machine learning and deep learning algorithms. In the second step, the data from the facial recognition system were transmitted to the electronics circuitry to make a smart lock system. Finally, the last step was to design a user interface for the google database that displays the attendance list.

This paper provides the required knowledge to build a facial recognition system and the necessary mathematical formulas of the algorithms used in the project. After the theory, the practical work is explained where those algorithms were implemented into practice, which was the major stage of the project to recognize faces and control the whole electronics circuitry.

2 Artificial Intelligence

Artificial Intelligence, also identified as AI, is a branch of computer science that empowers machines to learn from experience and perform specific tasks intelligently like a human brain without any interference [1]. AI challenges problems of modern life and tries to solve them using intelligent algorithms. It contains many theories, methods, technologies, and the following significant subfields as shown in figure 1. [2.]

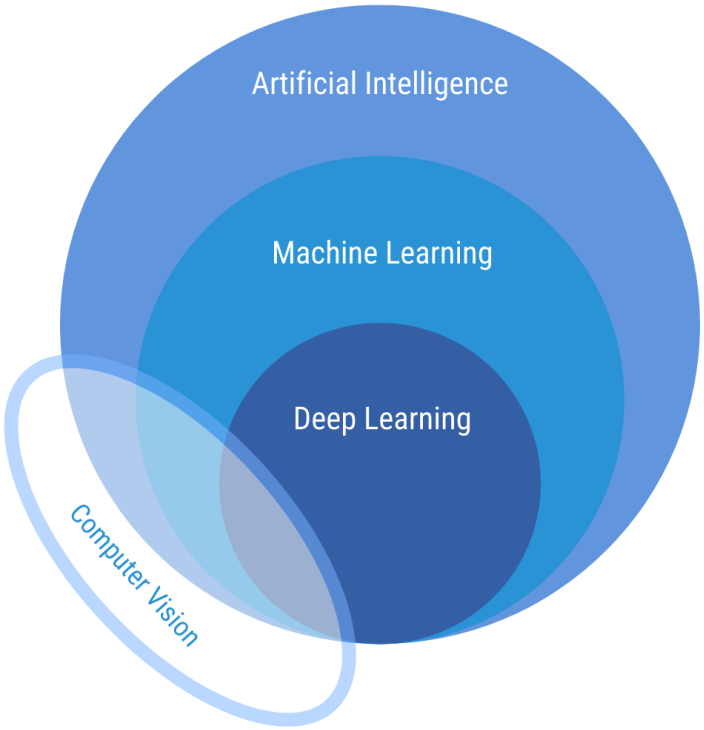

Figure 1. Major subfields of AI [3]

As Figure 1 shows, the significant subfields of AI are machine learning, deep learning, and computer vision which were used in the project for different purposes.

Artificial intelligence divides into two categories: strong AI and weak AI. Weak AI is a narrow application, and it is suitable for specific tasks, for instance, virtual assistants. On the other hand, strong AI is a broader application and has human-level intelligence. It is mainly used in advanced robotics and automation. [4.]

2.1 Machine Learning

Machine learning is a subfield of AI that can automatically learn through experience and data. It has algorithms that run on data to create mathematical models to make predictions or decisions. There are three primary machine learning methods: supervised, unsupervised, and reinforcement learning. [5.]

2.1.1 Supervised Learning

Supervised learning is one of the three primary methods of machine learning. It uses various algorithms that train using datasets to classify data or predict outputs, as illustrated in figure 2. [6.]

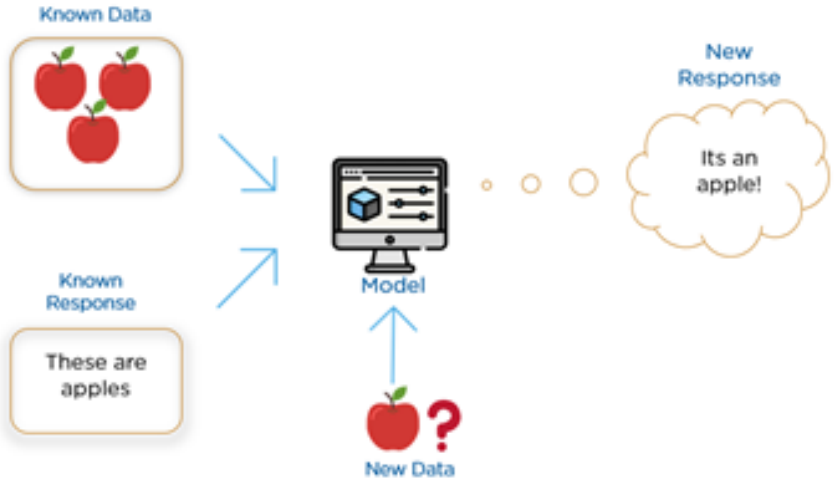

Figure 2. Visual Illustration of working principle of supervised learning algorithms [6]

The supervised learning algorithms begin the operation by feeding the data and adjusting the weights until the model fits appropriately. This process is used to ensure that the model prevents overfitting and underfitting. Over time, the algorithms learn to approximate the connection between the input data and labels. Once the algorithms are fully trained, they can observe the new objects and predict proper labels. [6.]

Supervised learning can be divided into regression and classification. Regression is used to understand the connection between dependent and independent variables. The popular regression algorithms include linear regression, logistical regression, and polynomial regression. [7.]

Classification uses an algorithm to assign test data into classes or groups. It identifies particular entities contained by the dataset and attempts to label those entities. The most familiar classification algorithms are support vector machines, decision trees, k-nearest neighbor, naïve bayes, and random forest. Support vector machine is one of the widely used algorithms that give a high test accuracy. Therefore, this was decided to be used in the project and is described in detail below. [7.]

Support Vector Machine

A support vector machine or SVM is a classification algorithm. The primary function of SVM is to find a hyperplane where the distance between two classes of data points is at its maximum. This hyperplane is the decision boundary, splitting the classes of data points on each side of the plane as shown in figure 3. [7.]

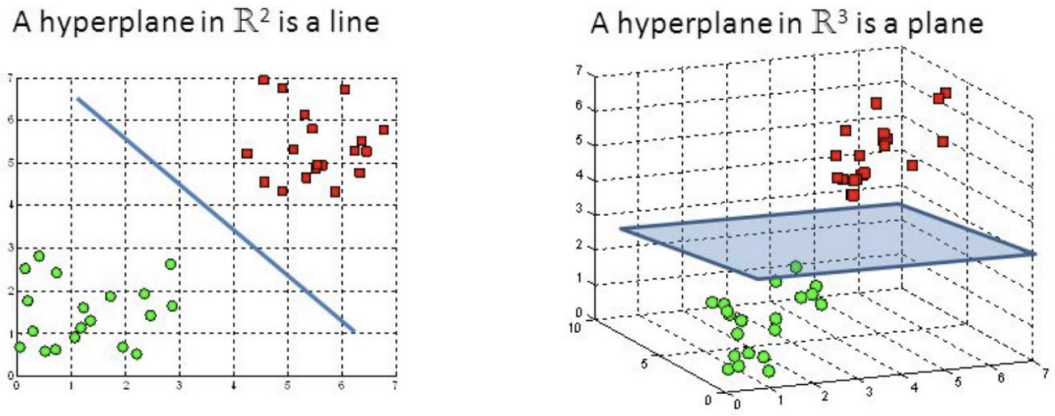

Figure 3. Hyperplanes in 2D and 3D space [8]

The dimension of the hyperplane depends on the space dimension. If the space is two-dimensional, then the hyperplane is simply a straight line. If it is three-dimensional, then the hyperplane becomes a two-dimensional plane.[8.]

There are a lot of possible hyperplanes that can be found in a plane, as shown in figure 4. In order to find the optimal hyperplane, among others, mathematical computation of the margin is needed, which is described below.

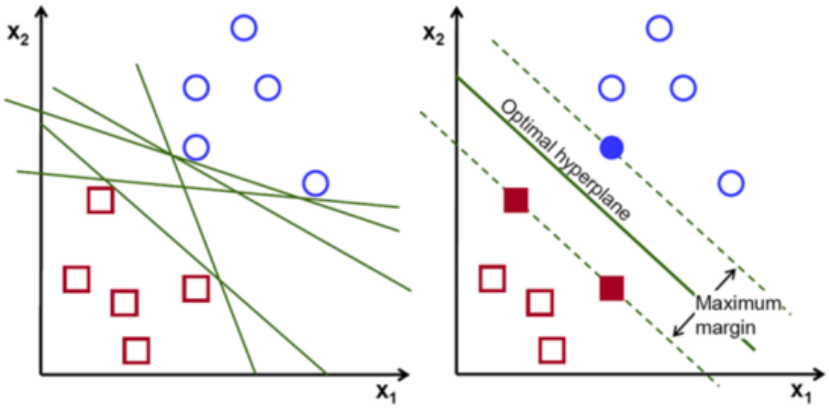

Figure 4. Illustration of optimal hyperplane [8]

The notation shown in equation (1) is used to define the hyperplane:

$$f(x) = w^T x + b \tag{1}$$

In the equation, w and b are the weight vector and the bias, respectively [9].

The optimal hyperplane can be represented in an infinite number by scaling of the weight and the bias, known as the canonical hyperplane:

$$|w^T x + b| = 1 \tag{2}$$

Where x is data points closer to the hyperplane called support vectors that are used to increase the margin and help to build a classifier. [9.]

The next step is to compute the distance between point x and the hyperplane by using the rule of geometry:

$$d = \frac{|w^T x+b|}{\|w\|} \tag{3}$$

According to the canonical hyperplane, the numerator in equation (2) is equal to one. Therefore the distance to the support vectors is

$$d_{sv} = \frac{|w^T x+b|}{\|w\|} = \frac{1}{\|w\|} \tag{4}$$

The margin is twice the distance between points and the hyperplane:

$$M = 2 * d_{sv} = \frac{2}{\|w\|} \tag{5}$$

The last step is to maximize the margin, which is equivalent to minimizing a function $L(w)$ subject to some constraints. Those constraints model the requirement for the hyperplane to classify all the data points x_i correctly. Formally,

$$\min_{w,w_0} L(w) = \frac{\|w\|^2}{2} \tag{6}$$

In order to find the perpendicular distance between two data points, x and z, the following formula is used.

$$\sqrt{\sum_{i=1}^{P}(x_i - z_i)^2} = \frac{|\mu(z)|}{\|w\|_2} \tag{7}$$

Equation (7) is the Euclidean distance formula and it is used to calculate the distance between two data points and implemented in face recognition, which is described in further sections. [10.]

2.1.2 Unsupervised learning

Unsupervised learning is pretty much the opposite of supervised learning. It uses algorithms to analyze unlabeled raw data, understand the properties, and learn to group them without human intervention, as illustrated in figure 5. [11.]

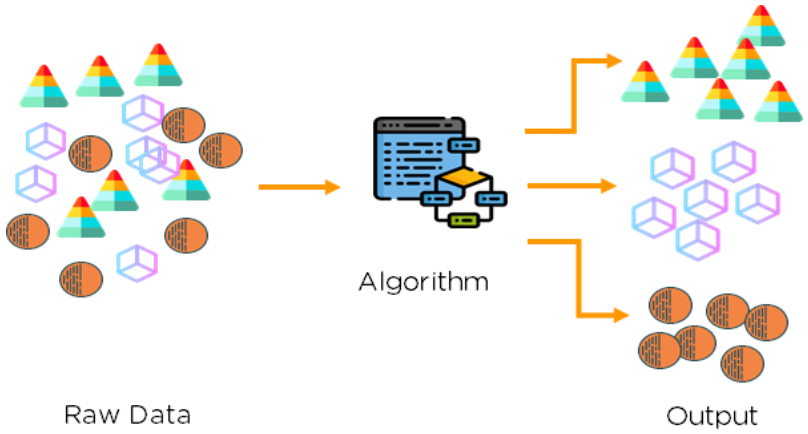

Figure 5. Visual Illustration of working principle of unsupervised learning algorithm [6]

Unsupervised learning algorithms have three main tasks, including clustering, association, and dimensional reduction [11].

Clustering

Clustering is one of the main unsupervised learning tasks. Its algorithms classify a set of unlabeled data so that data in the same cluster are more similar to each other than other clusters. [11.]

Association

An association rule is a widely-used method that explores the dataset and acquires the connections between variables in a dataset. This method is commonly used for market basket analysis, facilitating companies to understand the relationship between products. [11.]

Dimensionality Reduction

Dimensionality reduction is a technique applied when the dataset or dimensions in a given dataset are too large. It reduces input data to a feasible size, besides maintaining the integrity of the dataset. [11.]

2.1.3 Reinforcement Learning

Reinforcement learning is a machine learning model that uses intelligent algorithms. Those algorithms are not trained using data as in supervised learning. Instead, they learn from mistakes and experiences, as shown in the picture below. [5.]

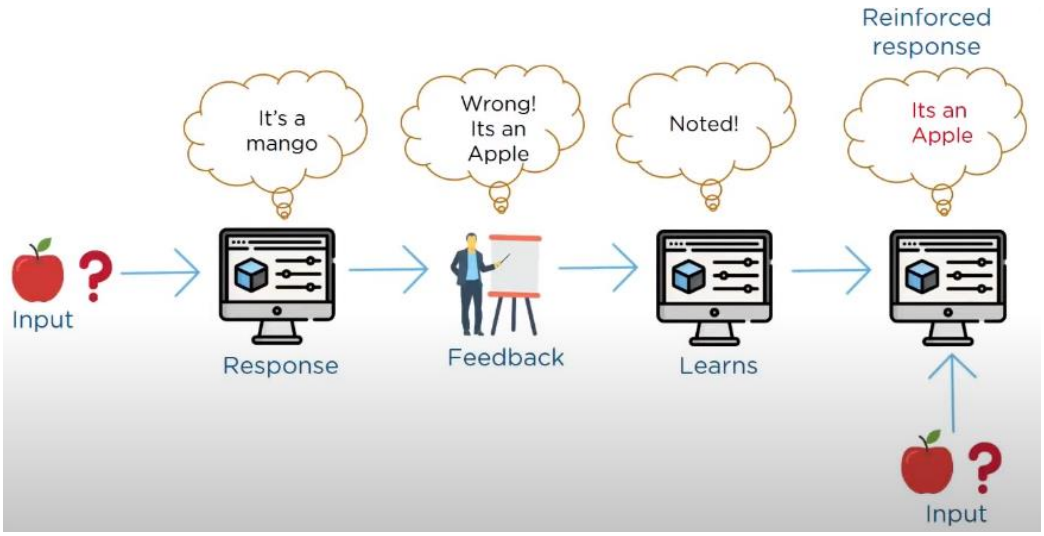

Figure 6. Visual Illustration of working principle of reinforcement learning [6]

The model gives faulty results in the beginning. Despite this, as long as the feedback is provided to the algorithm, it selects correct feedback over incorrect ones and improves itself for the subsequent trial. Over time, the algorithm learns and makes fewer mistakes than it used to. [6.]

2.2 Deep Learning

Deep learning, also called deep neural networks, is a subfield of machine learning, which is essentially a neural network with more than two layers. It is worth saying that the "deep" in deep learning refers to the depth of layers in a neural network. The function of deep learning is to learn from large amounts of data and perform like a human brain. [12.]

Deep learning algorithms process unstructured data, like text and images, and automate feature extraction. For instance, the algorithm processes a set of photos of different animals to categorize by a cat, dog, etc. They can determine which features are most significant to distinguish each animal from another, like ears, nose, etc. [12.]

Neural networks are at the heart of deep learning algorithms. Their name and structure are inspired by the biological neuron. Neuron in neural networks is a mathematical operation and imitates the functioning of a biological neuron, as schematized in figure 7. [13.]

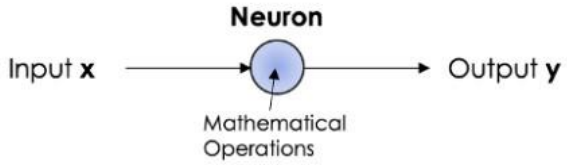

Figure 7. Scheme of the working process of a single neuron [14]

As Figure 7 illustrates, the input feeds into the neuron and produces the output. On the other hand, several input neurons are used to solve complicated problems, as shown in figure 8.

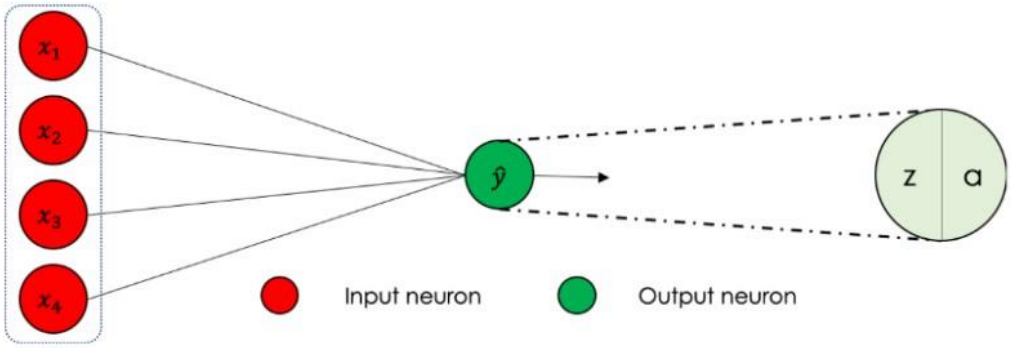

Figure 8. Scheme of the working process of several input neurons [14]

Here, each neuron is divided into two blocks:

- Computation of z using the inputs x_i:

$$z = \sum_i w_i x_i + b \tag{8}$$

- Computation of a, which is equal to y at the output layer, using z:

$$a = \psi(z) \tag{9}$$

Each neuron multiplies its weights w_i to inputs x_i, adds the bias b and passes the sum through the activation function ψ. [14.]

Neural networks have three main types: artificial neural networks, convolutional neural networks, and recurrent neural networks [14]. CNN is one of the widely used neural networks in face recognition, which is a topic of this thesis. Therefore, this was decided to be used during the project and described in detail below.

2.2.1 Artificial Neural Networks

Artificial Neural networks consist of three main layers of interconnected nodes, each building upon the previous layer to optimize the prediction or categorization. Those are input, hidden, and output layers, as shown in figure 9. [13.]

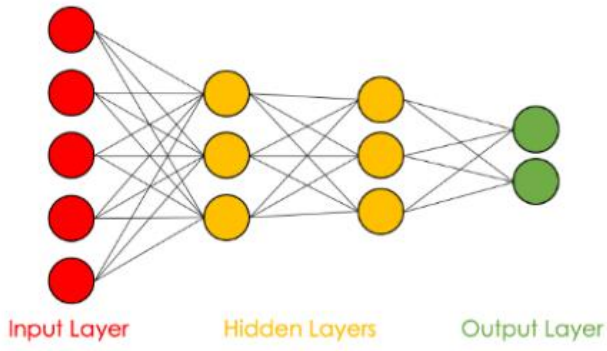

Figure 9. The architecture of neural networks [14]

The architecture of the artificial neural networks starts with the input layer, which ingests the data for processing, and gives the material to hidden layers to do all the mathematical computations. Finally, the output layer produces the result for given inputs. [15.]

2.2.2 Convolutional Neural Networks

CNN is a type of neural network that is very effective in image recognition and classification. They use a mathematical operation on two functions that produce a third function, called convolution, as shown in figure 10. [16.]

Figure 10. Process of a computing convolution function [17]

CNN starts the operation by converting the inputted image into pixels, and forwards it to filter processing. The filters used in image processing are vertical-edge and horizontal-edge filters. The combination of those filters gets the edges of an object in an image. [16.] The vertical edge filter, VEF, is defined as follows:

$$VEF = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = HEF^T \tag{10}$$

This filter slides over the input image to extract the vertical edges, which is the sum of the elementwise product in each block, as shown in figure 11. [16.]

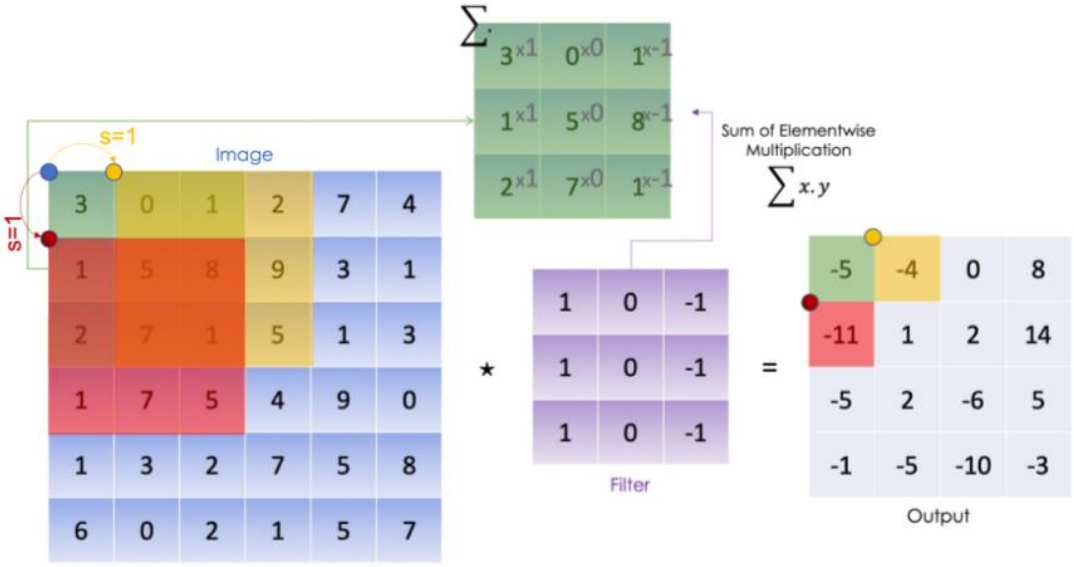

Figure 11. The feature map after filtering the image [16]

The elementwise multiplication is performed starting from the first 3x3 block, slides the block until it covers all possible blocks, and outputs the edges of the image, also called feature map. The parameter s in this figure is the stride parameter in the convolutional product. A large stride produces a smaller feature map and vice versa. [16.]

When VEF is used, the pixels on the edges are less used than those in the middle. It means that the data from the edges are ignored. In order to solve this problem, padding can be added around the image to consider the edge pixels, as shown in figure 12. [16.]

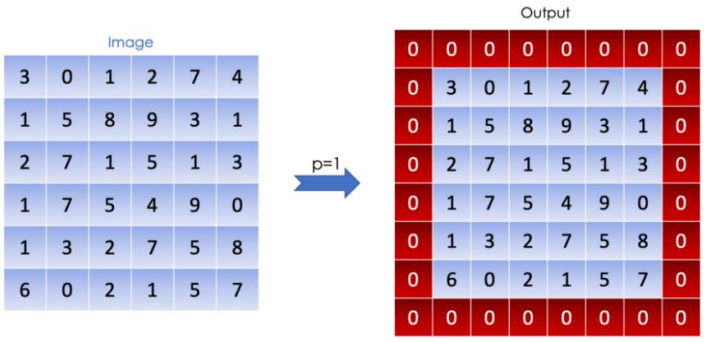

Figure 12. The output, after adding padding around the image [16]

The padding parameter p in figure 12 is the number of elements added to the four sides of an image [16].

Once the stride and the padding are defined, here comes to construct a CNN, layer per layer. CNN consists of three layers: convolution, pooling, and fully-connected layers. [16.]

2.2.2.1 Convolutional Layer

As mentioned above, CNN derives its name from the convolutional operator. The primary goal of the convolutional layer is to extract features from the input image, which can be mathematically represented as a tensor with the following dimensions:

$$\dim(image) = (n_H, n_W, n_C) \tag{11}$$

Here n_H is the height, n_W is the width and n_C is the number of channels, which are the depth of the matrices involved in the convolution. They are used to refer to a specific component of an image. If the image is grayscale, it has only one channel and has pixel values in the range of 0 to 255. On the other hand, if the image is RGB, the number of channels equals three. In this case, the filter can be represented with the following dimensions:

$$\dim(filter) = (f, f, n_C) \tag{12}$$

As described above, the convolutional product between an image and filter is a two-dimensional matrix. In the convolutional layer, each element is the sum of the elementwise multiplication of the filter, which is a cube, as illustrated in figure 13. [16.]

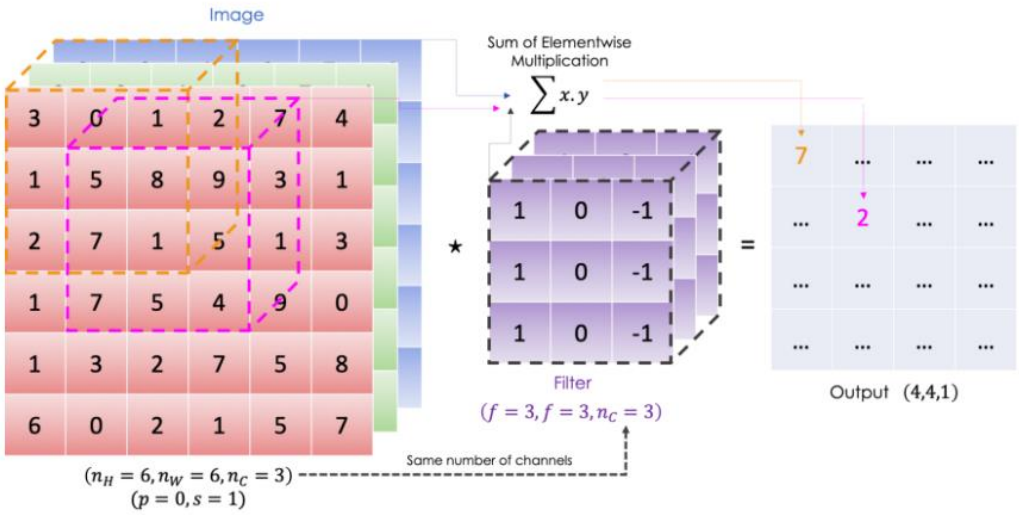

Figure 13. Illustration of a convolutional product on a volume [16]

The filter has the odd dimension f to center each pixel and the same number of channels as the input image [16].

In order to solve complex tasks, the convolutional product is applied using multiple filters and followed by an activation function ψ. The mathematical formula of the convolutional layer at the l^{th} layer is

$$conv\left(a^{[l-1]}, K^n\right)_{x,y} = \psi^{[l]}\left(\sum_{i=1}^{n_H^{[l-1]}} \sum_{j=1}^{n_W^{[l-1]}} \sum_{k=1}^{n_C^{[l-1]}} K_{i,j,k}^{(n)} a_{x+i-1,y+j-1,k}^{[l-1]} + b_n^{[l]}\right) \quad (13)$$

$$\dim\left(conv\left(a^{[l-1]}, K^{(n)}\right)\right) = \left(n_H^{[l]}, n_W^{[l]}\right) \quad (14)$$

Here, $a^{[l-1]}$ is the input image with the dimensions $(n_H^{[l-1]}, n_W^{[l-1]}, n_C^{[l-1]})$, and $n_C^{[l]}$ is the number of filters where each filter $K^{(n)}$ has the dimension of $(f^{[l]}, f^{[l]}, n_C^{[l-1]})$. The bias of the n^{th} convolution is $b_n^{[l]}$ and the activation function indicates as $\psi^{[l]}$. Finally, according to equation (22), the output from the convolutional layer can be written as

$$a^{[l]} = \left[\psi^{[l]}\left(conv\left(a^{[l-1]}, K^{(1)}\right)\right), \psi^{[l]}\left(conv\left(a^{[l-1]}, K^{(2)}\right)\right), \ldots, \psi^{[l]}(conv(a^{[l-1]}, K^{\left(n_C^{[l]}\right)}))\right] \quad (15)$$

$$\dim\left(a^{[l]}\right) = \left(n_H^{[l]}, n_W^{[l]}, n_C^{[l]}\right) \quad (16)$$

with

$$n_{H/W}^{[l]} = \left\lfloor \frac{n_{H/W}^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right\rfloor \quad (17)$$

According to these equations, the convolutional layer with multiple filters can be summarized in figure 14. [16.]

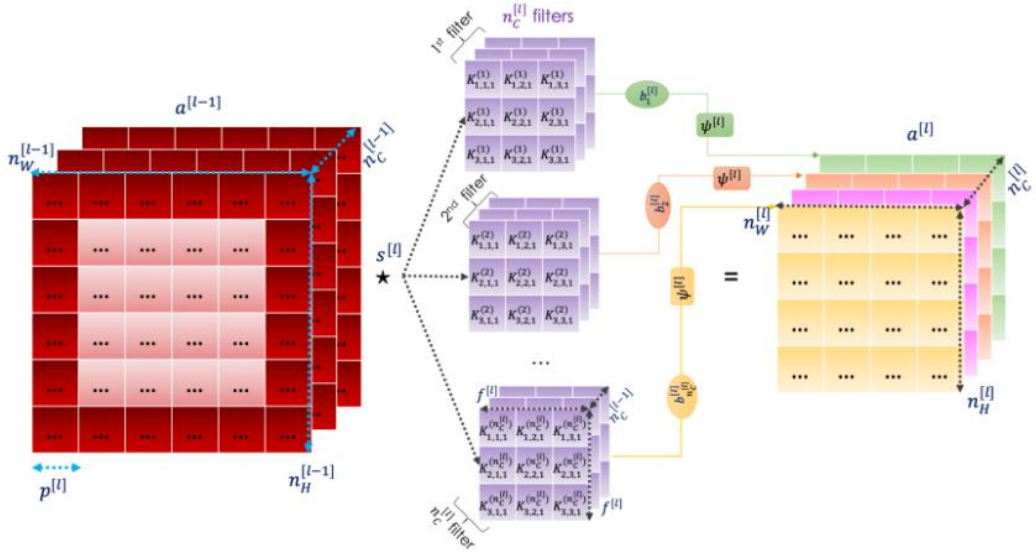

Figure 14. Illustration of the convolutional layer with multiple filters [16]

In figure 14, $p^{[l]}$ and $s^{[l]}$ are the padding and stride parameters, respectively, and the learned parameters from these convolutional layers are filters and the bias [16].

2.2.2.2 Pooling Layer

CNN uses the pooling layer to reduce the training time and the dimensionality of each feature map by applying it to each channel. However, it still maintains the useful information in the image. There are two often-used pooling types: max and average pooling. Max pooling returns the largest element from the feature map. On the other hand, average pooling takes the average of all elements, as illustrated in figure 15, when the stride parameter is equal to two. [16.]

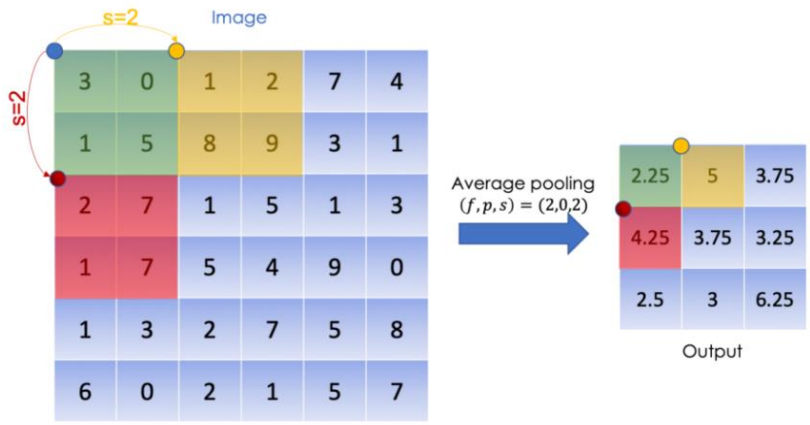

Figure 15. An illustration of average pooling [16]

The formula of the pooling layer at the l^{th} layer is

$$pool(a^{[l-1]})_{x,y,z} = \phi^{[l]}((a^{[l-1]}_{x+i-1,y+j-1,z})_{i,j\in\left[1,2,\ldots,f^{[l]}\right]^2}) \quad (18)$$

Here, $a^{[l-1]}$ is the input image to the pooling layer, which passes through a pooling function $\phi^{[l]}$ to the output $a^{[l]}$ as shown in figure 16. [16.]

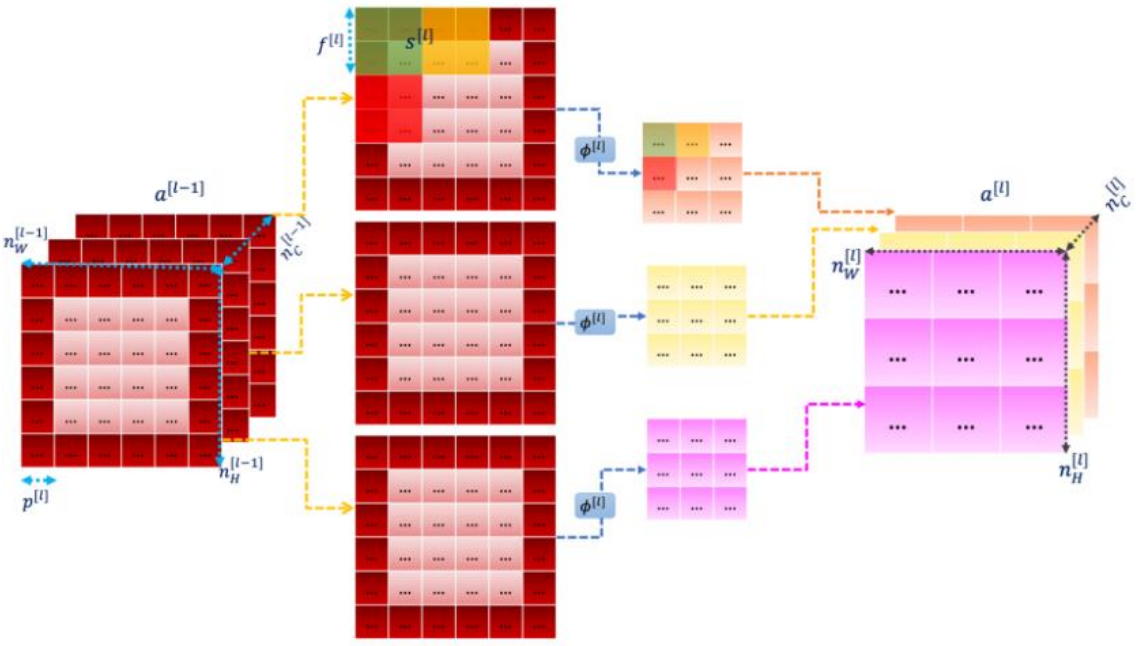

Figure 16. Illustration of the pooling layer [16]

This layer only produces the compressed version of images using the pooling function, and it has no learned parameters [16].

2.2.2.3 Fully Connected Layers

The fully connected layers are the main layers of the CNN, which connects every neuron in one layer to every neuron in the other layer. The primary purpose of these layers is to use convolutional and pooling layers and produce the desired output. They are the layers where the actual neural network starts and takes in a vector $a^{[l-1]}$ and returns a vector $a^{[l]}$. The formula of the fully connected layer on the j^{th} node of the i^{th} layer is

$$z_j^{[i]} = \sum_{l=1}^{n_{i-1}} w_{j,l}^{[i]} a_l^{[i-1]} + b_j^{[i]} \quad (19)$$

$$a_j^{[i]} = \psi^{[i]}(z_j^{[i]}) \quad (20)$$

Here, $w_{j,l}^{[i]}$ is the weight, $b_j^{[i]}$ is the bias, and $a^{[i-1]}$ is the output of the pooling layer with the dimensions $(n_H^{[i-1]}, n_W^{[i-1]}, n_C^{[i-1]})$. [16.]

The fully connected layers can be summarized in the illustration in figure 17.

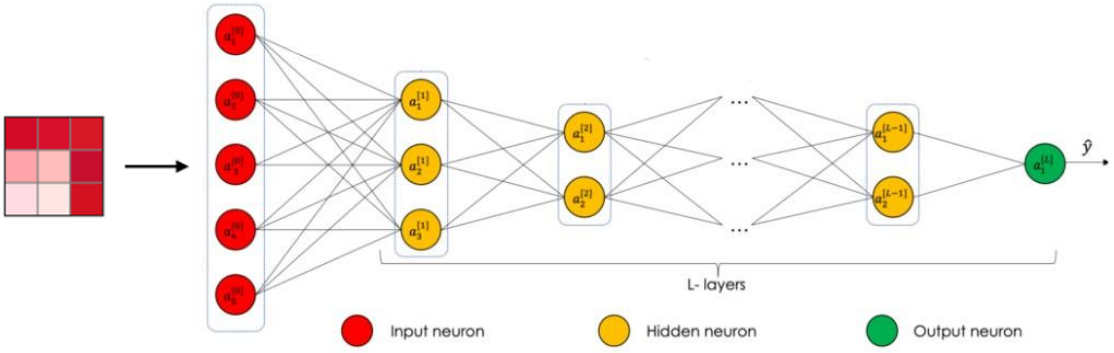

Figure 17. Illustration of the fully connected layer [16]

As can be seen here, the input is flattened to a one-dimensional vector, allowing the fully connected layers to start the operation. The formula of flattening can be expressed as

$$n_{i-1} = n_H^{[i-1]} \times n_W^{[i-1]} \times n_C^{[i-1]} \quad (21)$$

This vector feeds into the fully connected layer and generates the output. The learned parameters from this layer are the weights and the bias. [16.]

2.2.2.4 CNN in Overall

Overall, the convolutional neural network is a sequence of all layers and is illustrated in figure 18.

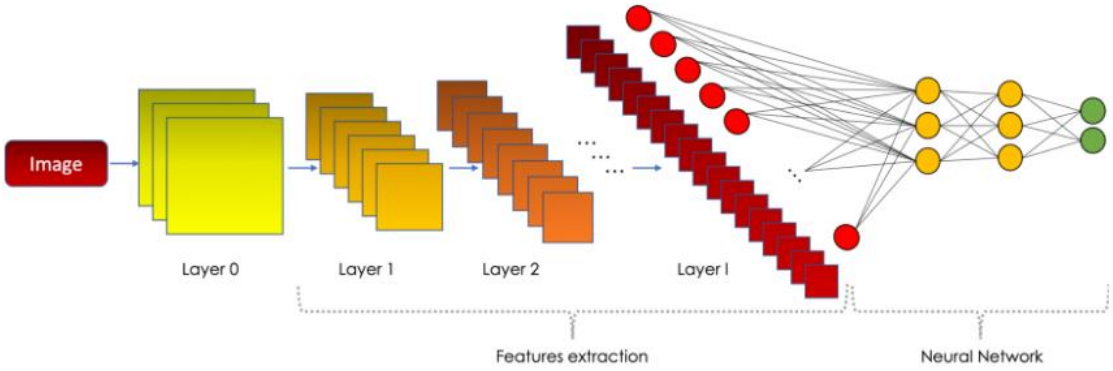

Figure 18. Illustration of the CNN [16]

Initially, CNN extracts features from the input image by performing the convolutional and the pooling layers. These features fed to fully connected layers to produce the output. The output can be the label or other features of the inputted image, like 128 measurements described in further sections.

2.2.2.5 Training the CNN

Data preprocessing

Data preprocessing is the step to transform the data so that the computer can easily read it. It is applied to increase the number of images in a given dataset. There are many techniques used in data preprocessing, such as cropping, rotation, flipping, etc. These techniques enable better learning due to the large size of the training set and allow the algorithm to learn from different conditions.

Before the CNN is trained, the dataset splits into training and test set. The training set is used to train the algorithm and consists of 80% of the dataset. On the other hand, the test set is used to check the algorithm's precision. [14.]

Learning algorithms

Learning algorithms aim to find the best parameters that give the best prediction. For this, the loss function J is defined to measure the distance between the real and the predicted values. The loss function has two steps: forward propagation and backward propagation. [14.]

Forward propagation is basically fully connected layers where the layers receive the input data, processes the information, and generates the predicted output value of x_i through the neural network $\hat{y}_i^{\theta}$ with some errors. In this case, the loss function J is evaluated as

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(\hat{y}_i^{\theta}, y_i) \quad (22)$$

Here, m is the size of the training set, θ is the model parameters, $\mathcal{L}$ is the cost function and y_i is the real values for all $i = (1,2, \dots, N)$. N is the iteration of the same process, called epoch number. [14.]

Backward propagation is the method to train neural networks. This method calculates the gradients of $\mathcal{L}$ for all the network parameters and adjusts those parameters based on the error rate obtained in the previous epoch.

The convolutional neural network is fully trained when the parameters are adjusted, and the training of CNN gives the minimum loss, which makes the model fast and reliable..

2.2.2.6 Activation Functions

Activation functions are an essential part of the neural network. They determine whether a neuron should be activated. The nonlinear functions typically convert the output of a given neuron to a value between 0 and 1 or -1 and 1. The most common activation functions are defined below. [18.]

- ReLU:

$$\psi(x) = x1_{x\geq 0} = \max(0, x) \quad (23)$$

- Sigmoid:

$$\psi(x) = \frac{1}{1+e^{-x}} \tag{24}$$

- Tanh:

$$\psi(x) = \frac{1-e^{-2x}}{1+e^{-2x}} \tag{25}$$

- LeakyReLU:

$$\psi(x) = x1_{x\geq 0} + \alpha x1_{x\leq 0} \tag{26}$$

2.2.3 Recurrent Neural Networks

The RNN is a type of neural network that applies sequential data and is used for natural language processing, speech recognition, language translation, etc. RNNs are derived from feedforward neural networks and can use their memory to take information from previous inputs to impact the current input and output, as shown in figure 19. [18.]

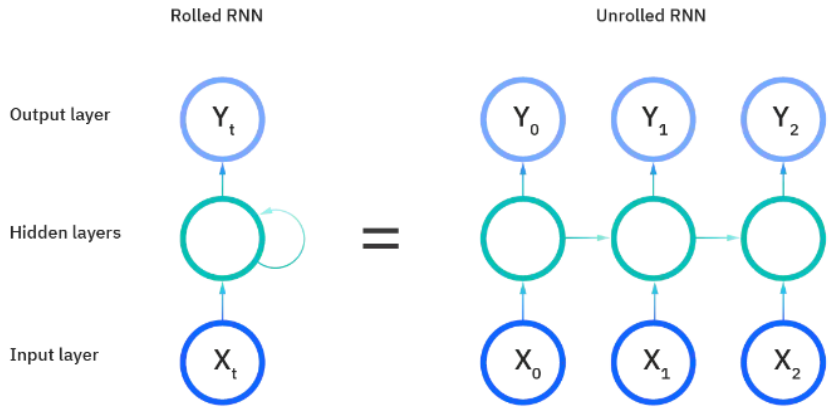

Figure 19. Illustration of the rolled and unrolled RNN [ibm]

The rolled RNN represents the total predicted outputs. On the other hand, the unrolled RNN represents the individual layers of the neural network, and each layer maps to a single output. [18.]

2.3 Computer Vision

Computer vision is a field of AI and works like human vision. It uses deep and machine learning algorithms described in sections 2.1 and 2.2 to enable computers to observe and understand images and videos by feeding lots of data. They run data over and over until they recognize images. [19].

One of the well-known computer vision applications is autonomous vehicles that need to identify people, cars, and lanes on the road in order to navigate [19].

3 Facial Recognition

Facial recognition is a category of biometric security used to identify people from images, videos, or real-time. It generally works by comparing a given face image with others in a database. [20.] This technology is mainly used in marketing, advertising, healthcare, banking, payment verification, and airports control [21].

Face recognition is executed in three stages: face detection, face encoding, and face classification [22].

3.1 Face Detection

The operation of face recognition starts by detecting faces which uses the HOG method to detect the faces in an image. The HOG stands for the histogram of oriented gradients. It starts the operation by converting an image to black and white. For every pixel in an image, surrounding pixels are selected to figure out the darkness of that pixel compared to surrounding pixels. Then the arrow is drawn in the direction of the darkness, as shown in figure 20. [22.]

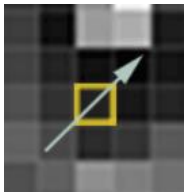

Figure 20. The drawn arrow on the pixel [22]

This process repeats for every single pixel in an image. In the end, every pixel replaces by arrows. These arrows are called gradients, which are obtained by combining

magnitude and angle from the image. First, gradients G_x and G_y are calculated for each pixel using the following formulas. [23.]

$$G_x(x,y) = H(x+1,y) - H(x-1,y) \quad (27)$$

$$G_y(x,y) = H(x,y+1) - H(x,y-1) \quad (28)$$

After these calculations, the magnitude and the direction of the gradient are obtained as

$$G(x,y) = \sqrt{G_x(x,y)^2 + G_y(x,y)^2} \quad (29)$$

$$\theta(x,y) = arg tan(^{G_y(x,y)}/_{G_x(x,y)}) \quad (30)$$

The magnitude and the direction are divided into several cells. For each cell, a 9-point histogram is calculated and each bin produces the intensity of gradient.

Four cells are combined to form a block once the histogram computation is over for all cells. This combining is done in an overlapping manner, as shown in figure 21. [23.]

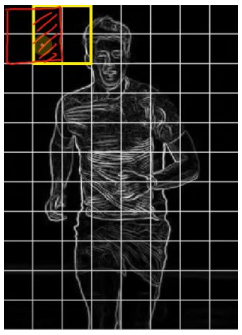

Figure 21. The HOG example with overlapping [23]

For all four cells in a block, 9-point histograms of each cell are concatenated to form a 36-point feature vector. Then the normalization is applied to reduce the effect of changes in the contrast between images of the same face. [23.]

Figure 22 below shows the inputted HOG image extracted from a bunch of other training faces [22].

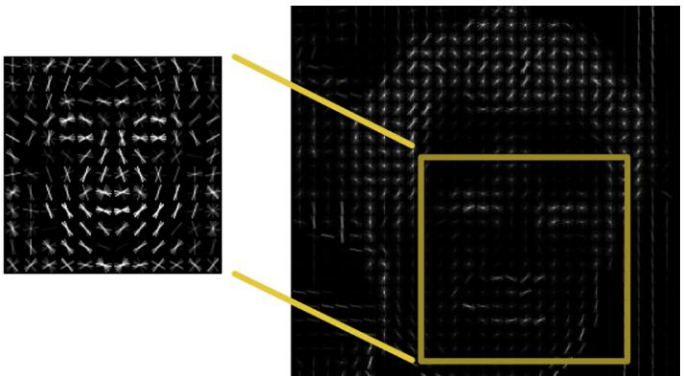

Figure 22. The HOG face pattern of an input image [22]

In this way, the faces can be easily found in any image. If the image size is 128x64, then the total HOG feature is

$$T_f = 7 * 15 * 36 = 3780 \tag{31}$$

Here, 36 is the feature vector, 7 and 15 are the blocks in horizontal and vertical directions, respectively. [23.]

Overall, the HOG method is schematized in figure 23 below.

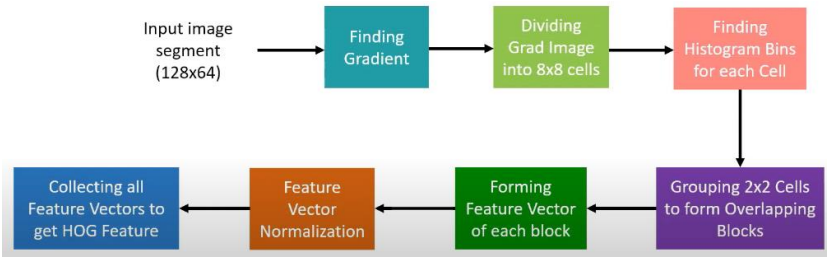

Figure 23. Scheme of the HOG method [23]

The HOG method goes through 8 steps to collect the feature vectors. Those feature vectors obtain the HOG feature according to the input image.

3.2 Face Encoding

After detecting the person's face, FaceNet is used to extract features from that face. It is a convolutional neural network published in 2015 by Google researchers Florian Schroff, Dmitry Kalenichenko, and James Philbin. Generally, the CNN trains to recognize pictures, objects, and digits. However, FaceNet takes an input image of a person's face, extracts the feature from convolutional and max-pooling layers as described in section 2.2.2, and generates a vector of 128 measurements from fully-connected layers, as shown in figure 24. [24.]

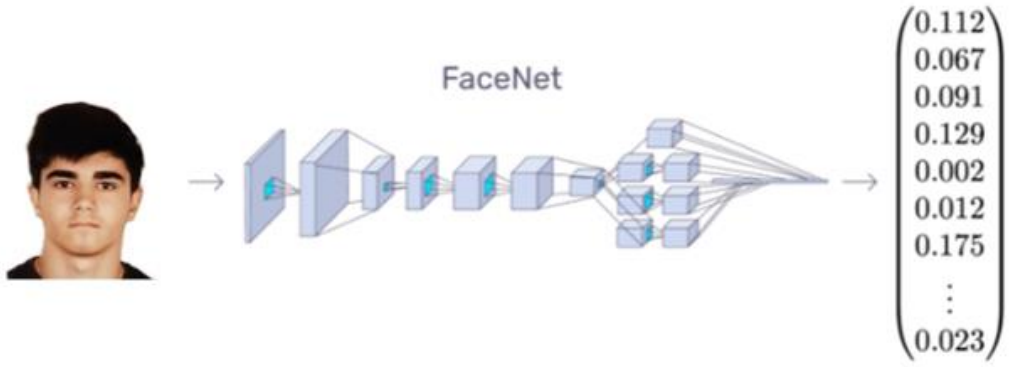

Figure 24. Illustration of the FaceNet (Modified from [24])

These 128 measurements are called embedding, which is a generic representation of a human face. FaceNet inserts this embedding into the triplet loss function to train the accuracy of the neural network classifier. The triplet loss function takes three vector variables as input: an anchor, a positive, and a negative, as shown in figure 25. [25.]

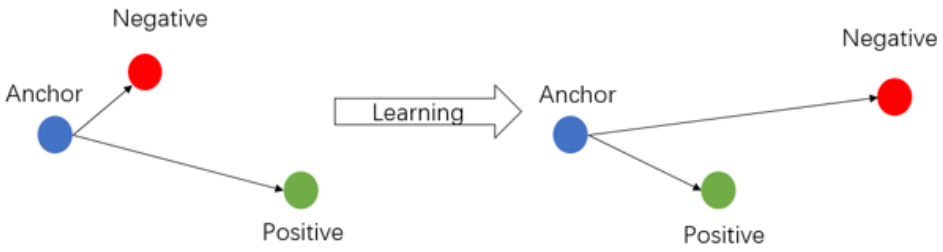

Figure 25. Distances between embeddings of anchor, positive and negative [20]

An anchor is the first known person image, a positive is another image of the same person, and a negative one is an image of a different person. Neural networks are

trained so that the embedding of anchor images should be close to positive embedding and far away from negative embedding. [25.]

When the embeddings give close measurements, the neural network is trained and can generate 128 measurements for any face [22].

3.3 Face Classification

The last step is to compare the embedding of the test image with the embedding of the database image. In this case, the machine learning algorithm SVM can be used to classify the test image with the closest match. As described in section 2.1.1, equation (7) is used to find the distance between two data points. The same technique can be applied to the embeddings of images. If the distance between these embeddings is small, the faces are from the same person and vice versa. [22.]

3.4 Face Recognition in Overall

Overall, the face recognition system can be summarized in the following figure 26.

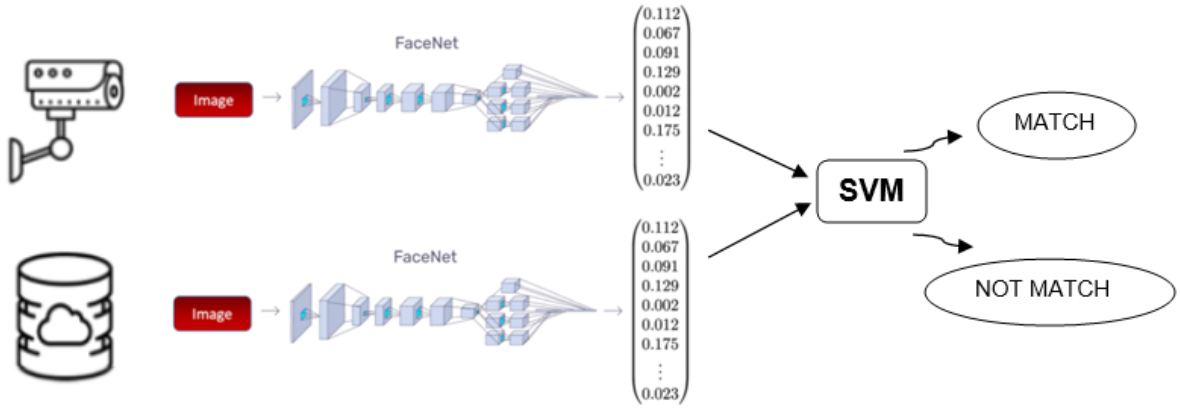

Figure 26. Illustration of the face recognition system (Modified from [24])

After FaceNet is trained, the database and the test images pass through the FaceNet, which generates embeddings. These embeddings feed into the SVM classifier to tell whether they match or not.

4 Implementation

This section of the thesis describes the practical use of the theoretical background, the necessary materials, tools, technologies, and the detailed workflow of the project.

4.1 Tools and Technologies

4.1.1 Python

Python is an object-oriented, high-level programming language released in 1991. It is mainly used for web development, artificial intelligence, machine learning, mathematics, data analysis, etc. Python has a simple syntax, so it is easier to read and understand. This simplicity makes it quicker to build and improve projects. Python supports modules and packages, encouraging program modularity and reuse of the code. [26.]

In this project, Python was used for machine learning, deep learning, mathematics, and computer vision by taking advantage of various Python libraries such as OpenCV, TensorFlow, and Openface.

4.1.2 OpenCV

OpenCV is an open-source computer vision and machine learning library. It was developed to support computer vision applications and accelerate machine perception. OpenCV runs in various operating systems, namely Windows, Mac, and Linux. It mainly focuses on video capturing, image processing, and analysis. [27.]

In this thesis work, the OpenCV library was used to read the path, capture the video, draw the frames, and put the name of the detected face.

4.1.3 TensorFlow

TensorFlow is an open-source platform that was developed by Google for machine learning. It has a complete, flexible ecosystem of tools, libraries, and resources that allows developers to quickly build ML-powered applications. TensorFlow can be used for various tasks but focuses on the training of deep neural networks. [28.]

4.1.4 Openface

OpenFace is an open-source library used in computer vision and deep learning. It is the first library capable of facial landmark detection, pose, eye-gaze estimation, and real-time performance. It can simply run from a laptop camera or webcam. Furthermore, OpenFace utilizes FaceNet for facial recognition, which is described in section 3.2. [29.]

4.1.5 Firebase

Firebase is a Google backend platform that helps to build and run web and mobile applications. This platform provides tools for analytics, reporting, marketing, fixing app crashes, cloud messaging, test lab, authentication, as well as a real-time database, which is used in the project and described in further sections. [30.]

4.1.6 HTML/CSS/JS

HTML, CSS, and JavaScript are the languages to run the web. They all are related but have specific functions. HTML controls the layout of the content, which provides the structure for the web page. Then CSS applies to stylize the web page elements, mainly targets various screen sizes to make web pages responsive. The last step is to use javascript for adding interactivity to a web page. [31.]

4.1.7 Jetson Nano

Jetson Nano is NVIDIA's small and powerful computer for AI purposes such as deep learning and computer vision. Figure 27 illustrates the jetson nano board. [32.]

Figure 27. Jetson nano board [32]

Jetson nano board has four USB ports, an HDMI port, two connectors for the CSI cameras, and 40 GPIO pins expansion header to control electronics components. The operating voltage for this board is 5 Volts using a barrel jack and a micro-USB port. The barrel jack delivers 4 Amps, while the micro-USB port has 2.5 Amps. [33.]

Jetson Nano allows running multiple neural networks in parallel for image classification, segmentation, object detection, speech processing, and face recognition [32].

4.1.8 Arduino

Arduino UNO is a programmable open-source microcontroller board based on the Atmega328p. The board contains six analog input pins, 14 digital I/O pins, a DC power jack, USB connector, as shown in figure 28. [34.]

Figure 28. Arduino UNO board [34]

This board can be integrated into electronic projects to control relays, LEDs, servos, and motors as an output. The operating voltage is 5 Volts, while the input voltage ranges between 6 Volts to 20 Volts. [34.]

4.2 Practical Work and Analysis

This section describes the implementation of the algorithms mentioned in sections 2 and 3, the usage of electronic sensors, and the design of the user interface to make a fully functional facial recognition system.

4.2.1 Hardware

Various components and sensors were used in this project to build the fully functional facial recognition system. Some of these components and sensors are attached to the Arduino UNO board and others to the Jetson Nano board, as illustrated in figure 29.

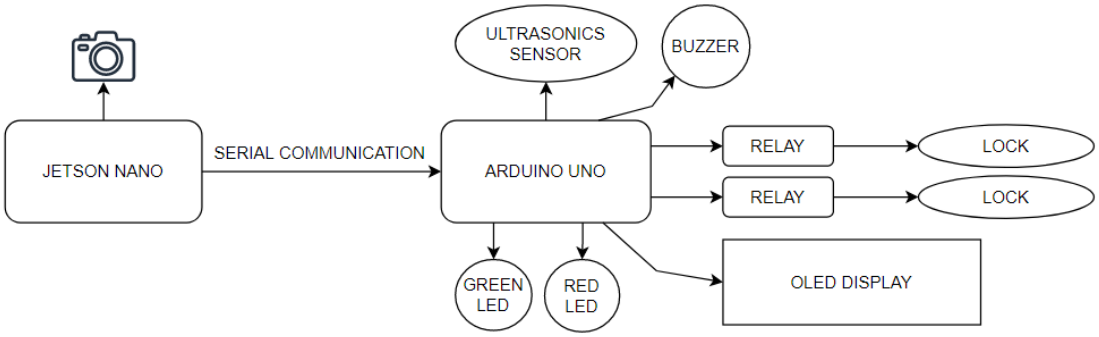

Figure 29. Block diagram of the hardware process

The table 1 below shows the list of all the necessary components, their quantity, and values.

Table 1. List of Components

Component	Quantity	Value
Resistor	2x	330 Ω
Green LED	1x	-
Red LED	1x	-
Solenoid lock	2x	12 Volts

Relay	2x	5 Volts
Buzzer	1x	-
Ultrasonic sensor	1x	-
OLED display	2x	-
Fan	1x	5 Volts
Webcam	1x	-
Wi-Fi Dongle	1x	-
USB cable	1x	-
Raspberry Pi adapter	1x	5V 2.5A
LiPo battery	1x	11.1V 1300mAh

In this project, the ultrasonic sensor was used to measure the distance. When the distance is less than 30 centimeters, then the buzzer buzzes, and the OLED display outputs the message “Please, Look at the camera,” as shown in figure 30.

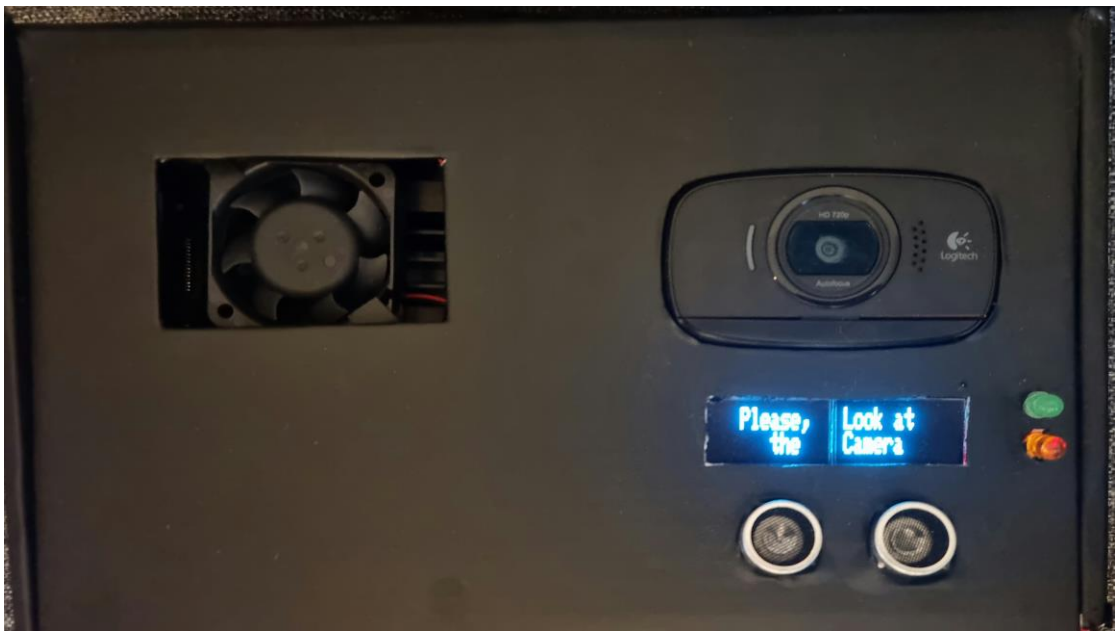

Figure 30. Top view of the project

Resistors were used to limit the current through the green and red LEDs. These LEDs were connected to the Arduino UNO. The green LED burns when the face is recognized, and the red LED burns when the access is denied, as shown in figure 31.

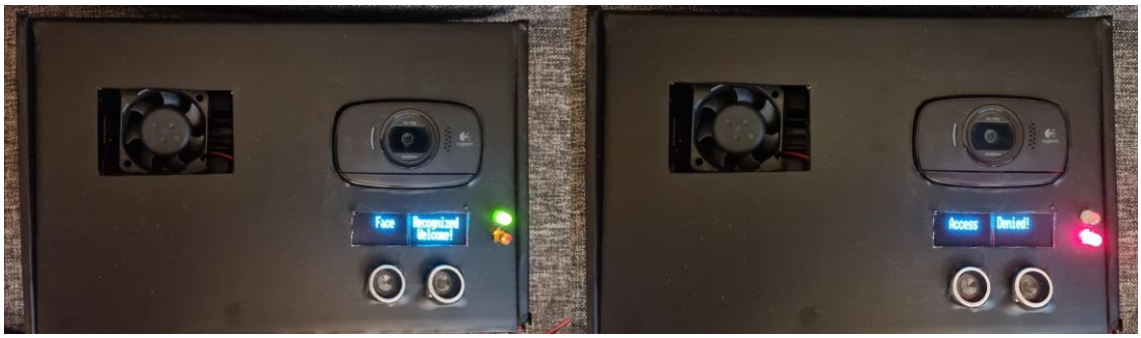

Figure 31. The action of the green and red LEDs

As figure 31 illustrates, the OLED display outputs messages “Face Recognized, Welcome!” and “Access Denied” according to the data.

The relays were used to send the power to solenoid locks in figure 32 below, which lock and unlock the door.

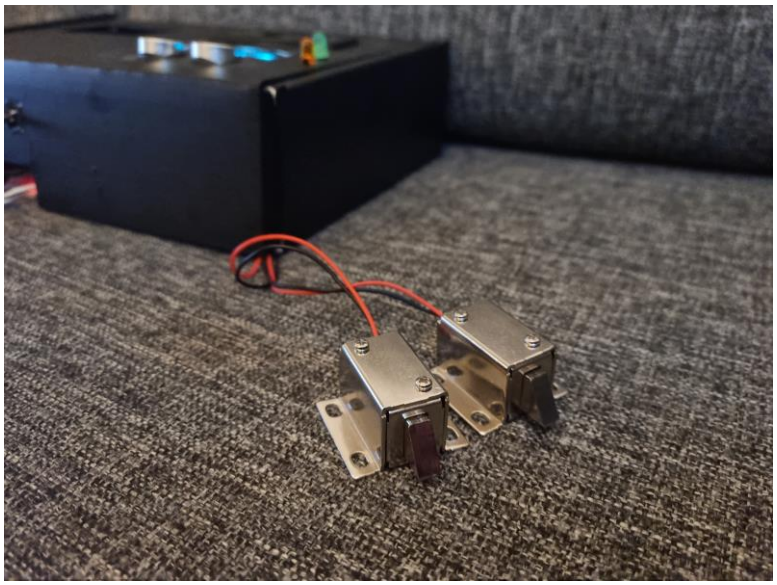

Figure 32. The solenoid locks

These locks work on 9 to 12 Volts. Therefore, an 11.1V Lipo battery was connected to supply the appropriate amount of voltage to the solenoid locks.

The fan was attached to the Jetson Nano heat sink to cool the processor during the training process, and the webcam was used to capture the video. The Wi-Fi dongle was plugged into the USB port of the jetson nano to access the internet since the Jetson Nano does not have built-in Wi-Fi. The board was powered using the 5V 2.5A

Raspberry Pi adapter and shared that power with Arduino using the USB cable. This USB cable was also used to make a serial communication between these two boards.

4.2.2 Software

In this section, the implementation of face recognition stages, database connection, user interface, serial communication, and transmitter and receiver codes were carried out. The block diagram in Figure 33 summarizes all the software stages below to understand the general idea of the working process of the facial recognition system.

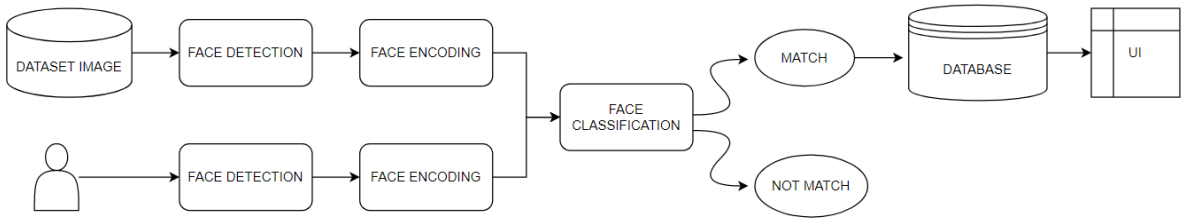

Figure 33. Block diagram of the software process

The dataset image and the real-time face pass through the facial recognition stages. When the embedding gives the close measurement in the face classification section, it means that the faces match, and the data is sent to the google database. All these steps in the block diagram are explained in further sections.

4.2.2.1 Implementation of the HOG method

In this project, AI operates to recognize faces. It starts the process by detecting the faces using the HOG method described in section 3.1. After inputting the face image, the HOG function was used to generate a face pattern, as shown in listing 1.

```
import matplotlib.pyplot as plt
from skimage.feature import hog
from skimage import data, feature, exposure
import cv2

image = cv2.imread('image1.jpg')
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
fd, hog_image = hog(image, orientations = 8, pixels_per_cell =
(16,16),cells_per_block = (1,1), visualize = True, multichannel =
True)
```

Listing 1. A python code that generates the face pattern using the HOG function [36]

Here, the HOG function was applied to 16x16 pixels per cell and 1x1 cells per block with eight vector orientations. The output from this HOG function can be plotted using the matplotlib library, as shown in listing 2 below.

```
fig, (ax1,ax2) = plt.subplots(1,2, figsize = (8,4), sharex = True,
sharey = True)
ax1.axis('off')
ax1.imshow(image, cmap = plt.cm.gray)
ax1.set_title('Input Image')
hog_image_rescaled = exposure.rescale_intensity(hog_image, in_range =
(0,10))
ax2.axis('off')
ax2.imshow(hog_image_rescaled, cmap = plt.cm.gray)
ax2.set_title('Histogram of Oriented Gradients')
plt.show()
```

Listing 2. A python code that plots the output from the HOG function [36]

The following figure 34 shows the output from the HOG function.

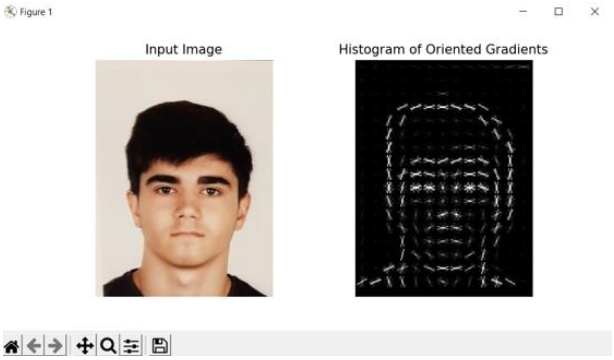

Figure 34. The output of the HOG function

This HOG image was inputted to the function in the face recognition library to detect the face, as shown in the following python code in listing 3.

```
import face_recognition

img = "Ogtay_Ahmadli.jpg"
color=(0,0,255)
faceLocationCurrentImage = face_recognition.face_locations(hog_image)
y1,x2,y2,x1 = faceLocationCurrentImage
cv2.rectangle(img,(x1,y1),(x2,y2),color,1)
```

Listing 3. A python code that draws a rectangle to the detected face

As listing 3 illustrates, face_locations() was used to extract the four points of the detected image. Then these points were applied to the OpenCV library to draw a rectangle on a face, as illustrated in figure 35.

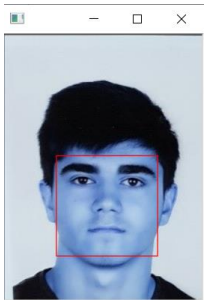

Figure 35. Detected face

4.2.2.2 Implementation of the Face Encodings

After the successful detection in figure 35, the new python subroutine called findEncoding() was created to find the encodings for each face image in the dataset. Firstly, this subroutine goes through the dataset, and for each image in the dataset, the FaceNet method was used to generate those encodings. When the encoding process is completed, the subroutine returns two lists. The first list is the encoding list of each image in the dataset, as illustrated in Appendix 1. The second list is the list of the names in the dataset, as shown in figure 36.

```
Encoding Completed
['Mehman Ahmadli', 'Ogtay Ahmadli', 'Telman Ahmadov']
```

Figure 36. The returned name list from the subroutine

4.2.2.3 Implementation of the Face Classification

Once the face images were encoded, the subroutine called recognizeFaces() was created to recognize faces using the support vector machine algorithm. This subroutine takes the returned lists from the previous subroutine as inputs along with the image. The process of the subroutine starts by generating the encodings of the real-time face image detected from a webcam. Next, the encodings are looped through to calculate the face distance and the result. The result is the list that compares the dataset faces with the real-time face using the compare_faces() function of the face recognition library and outputs the following list in figure 37.

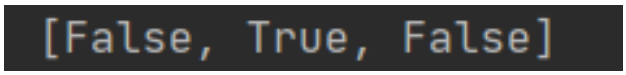

Figure 37. The output of the result list

As figure 37 illustrates, the recognized face is labeled as true and others as false, corresponding to figure 36.

The face distance is computed using equation (7) in section 2.1.1, which is the Euclidean formula to find the sum of the distance between encodings of the dataset and real-time faces, as shown in listing 3.

```
faceDistance = distance.euclidean(encodingList,encodingFace)
```

Listing 3. A python code to calculate the distance between encodings

The output from this calculation can be seen in figure 38.

Figure 38. The output from the euclidean formula

As figure 38 shows, the Euclidean distance of the recognized face is small compared to others. Then the NumPy library was applied to get the index of the minimum value of a list using the argmin() function, as shown in listing 4.

```
matchIndex = np.argmin(faceDistance)
```

Listing 4. The python code to get the minimum value of a list

The output from this line is equal to one, which is the index of the second element in a list in figure 38.

The following listing 5 checks whether the result in figure 38 is true or false at the minimum value.

```
names = []
if result[matchIndex]:
    name = classNames[matchIndex]
    color = (0,255,0)
    sm.sendData(ser,[0,0,1,0], 1)
```

```
else:
    name = 'unknown'
    color = (0, 0, 255)
    sm.sendData(ser,[1,1,0,1], 1)
names.append(name)
```

Listing 5. A python code to recognize faces.

Here, If the result is true, it means that the face is recognized. The name is labeled according to the list in figure 38 and the match index. Then the data is sent to the Arduino UNO to unlock the solenoid locks and turn on the green LED.

On the other hand, If the result is false, the name is labeled as ”unknown,” and the Arduino UNO receives the data to keep the locks closed and turn on the red LED.

After successful decisions, listing 3 in section 4.2.2.1 was slightly modified according to recognized and unrecognized faces, as shown in listing 6.

```
y1,x2,y2,x1 = faceLocation
y1,x2,y2,x1 = int(y1/0,25), int(x2/0,25), int(y2/0,25), int(x1/0,25)
cv2.rectangle(imgFaces,(x1,y1),(x2,y2),color,2)
cv2.putText(imgFaces, name, (x1+6, y1-6),
cv2.FONT_HERSHEY_COMPLEX,1,color,2)
```

Listing 6. A python code to draw a rectangle and put text to the recognize face [36]

Due to the image size in Figure 35, the face locations are increased four times to get the proper face frame from the webcam. Then a rectangle and a text were added around the face using the computer vision library.

4.2.2.4 Database

In this project, firebase was used to keep the data in google's real-time database. First, the firebase database was created, and then the following python module (listing 7) was designed to get the communication with firebase.

```
from firebase import firebase
import datetime
fb = firebase.FirebaseApplication('https://face-rec-dd032-default-
rtdb.firebaseio.com',None)
def postData(name, time):
    data = {'Name': name,'Time': time}
    dateToday = datetime.date.today().strftime('%Y-%m-%d')
    fb.post(f'/{dateToday}',data)
```

Listing 7. Firebase Module

After importing the firebase library, the URL of the firebase database was copied to the code. Then the postData() subroutine was created to post the name and the time to the database.

The next step was to create a markAttendance() subroutine, as shown in listing 8.

```
import FirebaseModule as fbm

def markAttendance(name):
    with open('Attendance.csv','r+') as f:
        myDataList = f.readlines()
        nameList = []
        for line in myDataList:
            entry = line.split(',')
            nameList.append(entry[0])
        if name not in nameList:
            now = datetime.now()
            dateString = now.strftime('%H:%M:%S')
            f.writelines(f'{name},{dateString}\n')
            fbm.postData(name,dateString)
```

Listing 8. The python subroutine that marks the name and the date [36]

As Listing 8 illustrates, an empty CSV file called Attendance was created to check whether the name is in the list or not. If the name is not in the list, then the subroutine posts the name and the time to the real-time database using the postData() function of the firebase module.

4.2.2.5 Transmitter Function

The transmitter function is the combination of all the subroutines mentioned above. It activates the webcam and uses the returned values of subroutines to generate the desired output, as illustrated in listing 9.

```
def main():
    encodingList, classNames = findEncodings("ImageAttendance")
    cap = cv2.VideoCapture(0, cv2.CAP_DSHOW)
    sm.sendData(ser,[1,1,0,0],1)

    while True:
        success, img = cap.read()
        imgFaces, names = recognizeFaces(img, encodingList,
classNames)
        for name in names:
            if name == "unknown":
                sleep(0.2)
            else:
                markAttendance(name)
        cv2.imshow("Image", imgFaces)
        if cv2.waitKey(1) & 0xFF == ord("q"): break
```

Listing 9. The transmitter function

The function starts the operation by taking the returned values of the findEncodings() function according to the images in the dataset called “ImageAttendance.” Then it activates the camera and sends the initial lock and LED values to the Arduino UNO board.

Then the webcam captures and inputs the image to the recognizeFaces() function. Here, the for loop was used to loop through the names of the captured faces. If the

face is not recognized, the program does not publish anything. Otherwise, the name and the time are sent to the database, as shown in Figure 39.

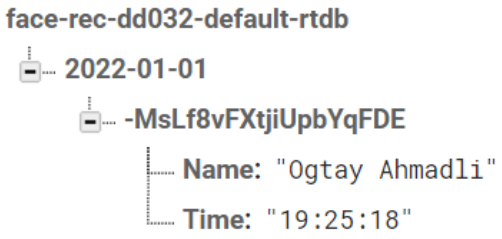

Figure 39. Data in the database

As Figure 39 illustrates, the data contains the name of the recognized person and the time it is recognized.

In the end, the function displays the output, which can be seen in figure 40.

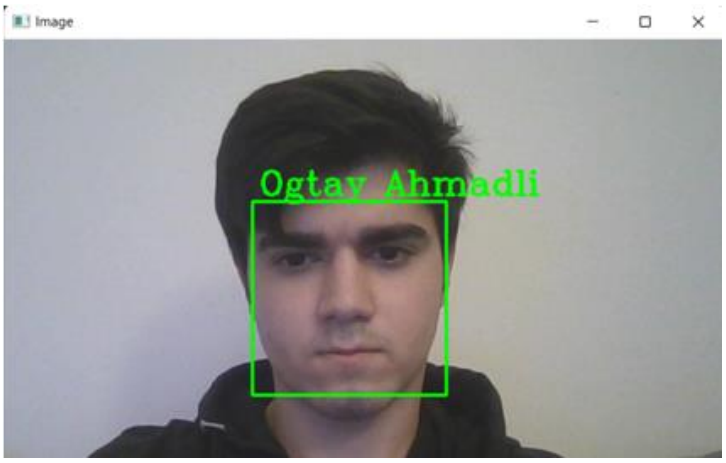

Figure 40. The output of the transmitter function

4.2.2.6 Serial Communication

In this project, the Jetson Nano is responsible for AI, and Arduino UNO is responsible for Electronics operation. The Jetson Nano board is in serial communication with Arduino UNO to transmit the desired data and make the components operate, as shown in figure 41.

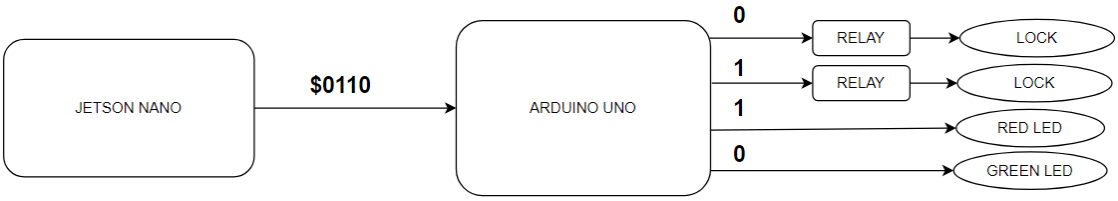

Figure 41. Illustration of serial communication

As figure 41 illustrates, Jetson Nano sends four-digit data to relays and LEDs. Here, the dollar sign was used to split the data in vertical order while looping, which avoids any confusion and defines the start and end digit of the signal. This sign was included in both transmitter and receiver codes.

When the Jetson Nano connects to Arduino UNO with the USB cable, the python subroutine shown in listing 10 checks if the boards are connected.

```
import serial
def initConnection(portNo, baudRate):
    try:
        ser = serial.Serial(portNo, baudRate)
        print("Device Connected ")
        return ser
    except:
        print("Not Connected ")
        pass
```

Listing 10. The python subroutine that tests the connectivity

Here, the subroutine checks the port number and the baud rate of the Arduino UNO using the serial library and returns those initialized serial objects. When the Arduino UNO is connected, the subroutine prints "Device Connected" and vice versa.

After the successful connection, the new subroutine was created to send the data to Arduino UNO, as shown in listing 11 below.

```
def sendData(ser, data, digits):
    myString = "$"
    for d in data:
        myString += str(int(d)).zfill(digits)
    try:
        ser.write(myString.encode())
        print(myString)
    except:
        print("Data Transmission Failed ")
```

Listing 11. The python subroutine that sends the data

This subroutine takes the initialized serial object, data, and digits per data value as inputs. The subroutine starts looping through the data. For each data, it inserts the dollar sign and sends that data to the relevant port. If some issues occur in the connection, the subroutine prints "Data Transmission Failed."

The next step was to create a receiver function for Arduino UNO to control the components. This subroutine starts the operation by checking the dollar sign, as shown in listing 12 below.

```
#define numOfValsRec 4
#define digitsPerValRec 1

int valsRec[numOfValsRec];
int stringLength = numOfValsRec * digitsPerValRec + 1;
int counter = 0;
bool counterStart = false;
String receivedString;

void receiveData() {
   while (Serial.available()) {
    char c = Serial.read();
    if (c == '$') {
      counterStart = true; }
    if (counterStart) {
```

```
      if (counter < stringLength) {
        receivedString = String(receivedString + c);
        counter++;}
      if (counter >= stringLength) {
        for (int i = 0; i < numOfValsRec; i++){
          int num = (i * digitsPerValRec) + 1;
          valsRec[i] = receivedString.substring(num, num +
digitsPerValRec).toInt();}
        receivedString = "";
        counter = 0;
        counterStart = false; }}}}
```

Listing 12. The Arduino C function that receives data [35]

As Listing 12 shows, when the dollar sign is detected and the counter is less than a string length, then the function gets the data and increments the counter. Following this, it loops through the received data elements. For each element, an array was utilized to get and use them in the code independently.

4.2.2.7 Receiver Function

Firstly, the Arduino pin of each component was defined and set up as input or output. Then the new function was created to pass the received data to solenoid locks and LEDs, as shown in listing 13.

```
void unlock_solenoid() {
    digitalWrite(solenoid1Pin, valsRec[0]);
    digitalWrite(solenoid2Pin, valsRec[1]);
    digitalWrite(greenLed, valsRec[2]);
    digitalWrite(redLed, valsRec[3]);}
```

Listing 13. The Arduino subroutine that sends digital values to the components

As listing 13 shows, the array was used to get each signal element and assigned to the components using the function in listing 3.

Overall, there are three main functions in the code that loops all the time, as shown in listing 14.

```
void loop() {
  receiveData();
  unlock_solenoid();
  oled();
}
```

Listing 14. The Looping process of the functions

The first function is to receive the data from the Jetson Nano. The second one is the function above to pass data to the components. Finally, the last function is to display the status message on the OLED display according to the data and the distance from the ultrasonic sensor.

4.2.3 User Interface

The web page was created using HTML, CSS, and javascript. The first step was to create a login interface for the webpage, which can be seen in figure 42.

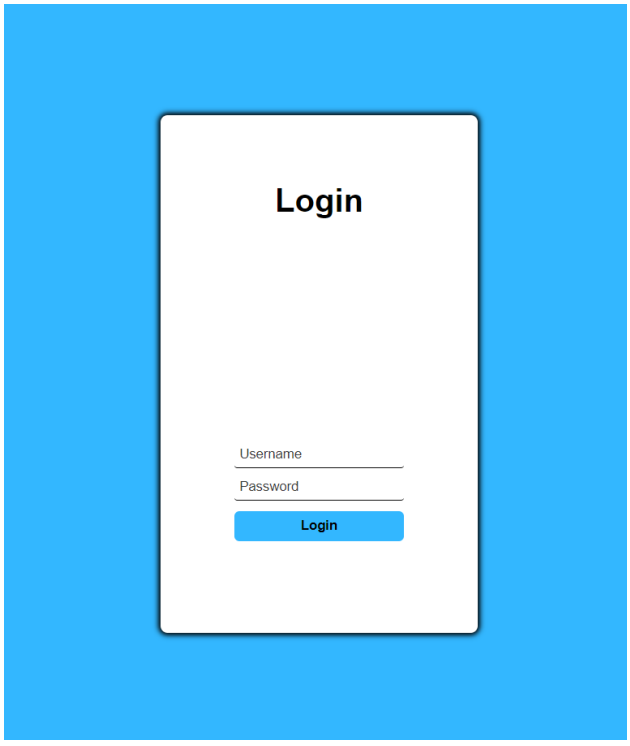

Figure 42. Login Interface

After a successful login, The firebase configurations were used to access the data, and the webpage displays it, as the following figure 43.

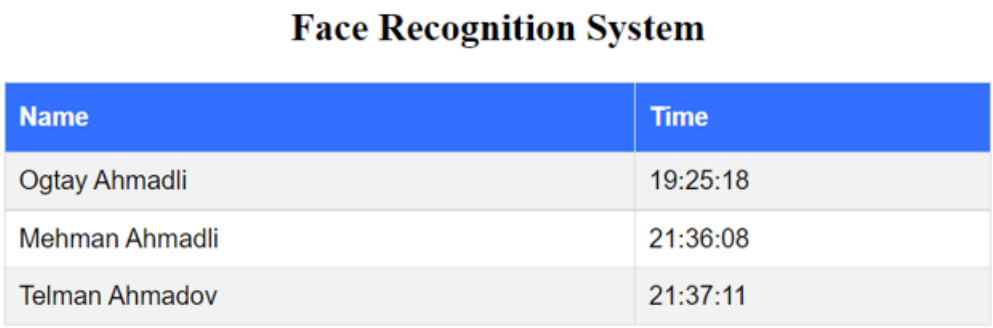

Figure 43. List of the recognized people

5 Conclusion

The goal of the project was to build a facial recognition system that could recognize human faces, log information into the database, and unlock the door.

The thesis project was executed in three steps. During the first step, the machine learning and deep learning algorithms were used to recognize faces and send the data to the google database. In the second step, AI data is transmitted to the electronics components and sensors to make a smart lock system. Finally, the last step was to design a webpage that requires a login and displays the attendance list.

The project's result was accomplished as expected, and the prototype could successfully recognize human faces and activate the electronics components. It has fast performance and could log information about recognized humans in the Google database.

This prototype can be used for office doors to identify employees, open the door and send the boss an attendance list, which displays the employee's name and entry time. A future improvement of the prototype could be implementing more extensive algorithms to distinguish the pictures and real faces from a camera. These algorithms would make the prototype faster, secure, and suitable for commercial purposes.

References

1 Silke Otte [online] How does Artificial Intelligence work?
URL: https://www.innoplexus.com/blog/how-artificial-intelligence-works/
Accessed on: 14.10.2021

2 Sas [online] Artificial Intelligence
URL: https://www.sas.com/en_us/insights/analytics/what-is-artificial-intelligence.html
Accessed on: 14.10.2021

3 Resquared [online] What is AI?
URL: https://www.resquared.com/blog/what-is-ai
Accessed on 14.10.2021

4 IBM [online] Strong AI
URL: https://www.ibm.com/cloud/learn/strong-ai
Accessed on: 14.10.2021

5 IBM [online] Machine Learning
URL: https://www.ibm.com/cloud/learn/machine-learning
Accessed on: 15.10.2021

6 Towards Data Science [online] What are the types of machine learning?
URL: https://towardsdatascience.com/what-are-the-types-of-machine-learning-e2b9e5d1756f
Accessed on: 15.10.2021

7 IBM [online] Supervised learning
URL: https://www.ibm.com/cloud/learn/supervised-learning
Accessed on: 16.10.2021

8 Rohith Gandhi [online] Support Vector Machine – Introduction to Machine Learning Algorithms
URL: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47
Accessed on: 16.10.2021

9 OpenCV [online] Introduction to Support Vector Machines
URL: https://docs.opencv.org/3.4.15/d1/d73/tutorial_introduction_to_svm.html
Accessed on: 16.10.2021

10 Yeng Miller – Chang [online] The mathematics of Support Vector Machines
URL: https://www.yengmillerchang.com/post/svm-lin-sep-part-1/
Accessed on: 16.10.2021

11 IBM [online] Unsupervised Learning
URL: https://www.ibm.com/cloud/learn/unsupervised-learning
Accessed on: 20.10.2021

12 IBM [online] Deep Learning
URL: https://www.ibm.com/cloud/learn/deep-learning
Accessed on: 22.10.2021

13 IBM [online] Neural Networks
URL: https://www.ibm.com/cloud/learn/neural-networks
Accessed on: 23.10.2021

14 Ismail Mebsout [online] Deep Learning's mathematics
URL: https://towardsdatascience.com/deep-learnings-mathematics-f52b3c4d2576
Accessed on: 23.10.2021

15 Gavril Obnjanovski [online] Everything you need to know about Neural networks and backpropagation
URL: https://towardsdatascience.com/everything-you-need-to-know-about-neural-networks-and-backpropagation-machine-learning-made-easy-e5285bc2be3a
Accessed on: 23.10.2021

16 Ismail Mebsout [online] Convolutional Neural Network's mathematics
URL: https://towardsdatascience.com/convolutional-neural-networks-mathematics-1beb3e6447c0
Accessed on: 25.10.2021

17 Wikipedia [online] Convolution
URL: https://en.wikipedia.org/wiki/Convolution
Accessed on: 25.10.2021

18 IBM [online] Recurrent Neural Networks
URL: https://www.ibm.com/cloud/learn/recurrent-neural-networks#toc-what-are-r-btVB33l5
Accessed on: 26.10.2021

19 IBM [online] Computer Vision
URL: https://www.ibm.com/se-en/topics/computer-vision
Accessed on: 26.10.2021

20 Satyam Kumar [online] Face Recognition with OoenFace
URL: https://medium.com/analytics-vidhya/face-recognition-using-openface-92f02045ca2a
Accessed on: 30.10.2021

21 Kaspersky [online] What is Facial Recognition
URL: https://www.kaspersky.com/resource-center/definitions/what-is-facial-recognition
Accessed on: 01.11.2021

22 Adam Geitgey [online] Modern Face Recognition with Deep Learning
URL: https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78
Accessed on: 01.11.2021

23 Mrinal Tyagi [online] Histogram of the oriented gradients
URL: https://towardsdatascience.com/hog-histogram-of-oriented-gradients-67ecd887675f
Accessed on: 03.11.2021

24 Luka Dulcic [online] Face Recognition with FaceNet and MTCNN
URL: https://arsfutura.com/magazine/face-recognition-with-facenet-and-mtcnn/
Accessed on: 03.11.2021

25 Ismail Mebsout [online] Object Detection and Face Recognition algorithms
URL: https://towardsdatascience.com/object-detection-face-recognition-algorithms-146fec385205
Accessed on: 03.11.2021

26 Python [online] What is Python?
URL: https://www.python.org/doc/essays/blurb/
Accessed on: 10.11.2021

27 OpenCV [online] About
URL: https://opencv.org/about/
Accessed on: 11.11.2021

28 TensorFlow [online] Introduction to TensorFlow
URL: https://www.tensorflow.org/learn
Accessed on: 11.11.2021

29 MultiComp Lab [online] OpenFace
URL: http://multicomp.cs.cmu.edu/resources/openface/
Accessed on: 11.11.2021

30 Google Firebase [online] Firebase
URL: https://firebase.google.com/
Accessed on: 15.11.2021

31 Interneting is hard [online] Introduction to HTML, CSS and Javascript
URL: https://www.internetingishard.com/html-and-css/introduction/
Accessed on: 15.11.2021

32 Nvidia Developer [online] Jetson Nano Developer kit
URL: https://developer.nvidia.com/embedded/jetson-nano-developer-kit
Accessed on: 16.11.2021

33 Nvidia Developer [online] Getting started with Jetson Nano Developer kit
URL: https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-devkit#intro
Accessed on: 17.11.2021

34 Arduino [online] Overview
URL: https://www.arduino.cc/en/pmwiki.php?n=Main/arduinoBoardUno
Accessed on: 20.11.2021

35 Arduino [online] Code
URL: https://forum.arduino.cc
Accessed on. 20.11.2021

36 Python [online] Code
URL: https://www.analyticsvidhya.com/blog/2021/11/build-face-recognition-attendance-system-using-python/
Accessed on: 21.11.2021

Appendix 1. The encodings of an image in the dataset

```
[array([-9.27751213e-02,  8.37971792e-02, -6.66356785e-03, -1.46875679e-01,
        -9.73482653e-02,  3.65512632e-02, -9.97424722e-02, -2.10721940e-02,
         1.75890714e-01, -7.38910511e-02,  2.33546436e-01, -6.36117905e-03,
        -2.00898841e-01, -7.43874684e-02, -1.00705978e-02,  3.98356169e-02,
        -1.19481899e-01, -8.14753920e-02, -1.06310442e-01,  1.28049031e-02,
         3.79619263e-02,  1.25408098e-01,  8.88260454e-03,  6.64873868e-02,
        -1.95187896e-01, -4.03097898e-01, -9.16480124e-02, -7.87187070e-02,
        -5.37221842e-02, -1.76542819e-01,  5.34996949e-02,  1.33711800e-01,
        -1.93653062e-01, -4.09115031e-02,  8.10934324e-03,  5.30757718e-02,
        -3.19512188e-02,  1.43463295e-02,  1.45343810e-01,  6.96208775e-02,
        -1.68422952e-01,  2.22997051e-02,  2.28786953e-02,  3.18699539e-01,
         2.15458140e-01,  2.10151225e-02, -1.93308480e-03, -6.44823164e-02,
         1.02172017e-01, -2.13793740e-01,  6.70833737e-02,  1.64675027e-01,
         5.51297963e-02,  3.68565209e-02,  1.00353450e-01, -3.87370288e-02,
         9.76922642e-03,  1.46807730e-01, -1.88594982e-01, -1.63083263e-02,
         4.79167374e-03, -1.25910863e-02, -2.43160352e-02, -4.93273288e-02,
         1.41232491e-01,  2.51120310e-02, -8.95359069e-02, -1.41534001e-01,
         6.81141838e-02, -1.39034495e-01, -6.37964830e-02,  1.11667015e-01,
        -1.57510370e-01, -1.94006741e-01, -3.29888165e-01,  8.70602503e-02,
         3.06037366e-01,  1.47416800e-01, -1.34641767e-01,  5.91222346e-02,
        -5.06316982e-02, -7.98835307e-02, -1.73722790e-03, -1.67507418e-02,
        -1.21616773e-01, -1.03466092e-02, -3.37315798e-02,  1.45569876e-01,
         2.03683898e-01,  4.46848273e-02, -6.92581385e-03,  2.47036919e-01,
         2.20097601e-02, -1.86577216e-02,  4.82001975e-02,  1.17500089e-02,
        -8.06881785e-02, -3.14362682e-02, -1.51013613e-01,  1.42218377e-02,
         4.11931500e-02, -5.18183876e-03, -7.74545129e-03,  1.72503248e-01,
        -2.14133263e-01,  1.70534998e-01,  7.78212864e-03, -8.68732333e-02,
         1.40019832e-03,  2.81511340e-04, -1.34194031e-01, -9.09589455e-02,
         1.43464655e-01, -2.40742475e-01,  7.77156502e-02,  1.19973890e-01,
         1.31654933e-01,  1.09744161e-01,  7.83238113e-02, -2.35875268e-02,
        -4.35248241e-02,  1.23253968e-02, -1.48052365e-01, -2.72081383e-02,
         1.44324109e-01,  1.40274726e-02,  4.97319102e-02, -3.97315472e-02])]
```

Figure 44. The encodings of the image