T.C.
KIRŞEHİR AHİ EVRAN ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
İLERİ TEKNOLOJİLER ANABİLİM DALI

GERÇEK ZAMANLI YÜZ MASKESİ ALGILAMA UYARI SİSTEMİ

ALI ABBAS JASIM

YÜKSEK LİSANS TEZİ

KIRŞEHİR / 2022

T.C.
KIRŞEHIR AHI EVRAN UNIVERSITY
INSTITUTE OF SCIENCES
DEPARTMENT OF ADVANCED TECHNOLOGIES

REAL-TIME FACE MASK DETECTION WITH ALERT SYSTEM

ALI ABBAS JASIM

MASTER'S THESIS

Supervised by
Asst. Prof. Dr. MUSTAFA YAĞCI

KIRŞEHİR / 2022

Bu çalışma ……. tarihinde aşağıdaki jüri tarafından …… Anabilim Dalı, ……Programında Yüksek Lisans / Doktora tezi olarak kabul edilmiştir.

Tez Jürisi

Kırşehir Ahi Evran Üniversitesi
……Fakültesi

……
Kırşehir Ahi Evran Üniversitesi
…… Fakültesi

Prof. Dr. …….
Kırşehir Ahi Evran Üniversitesi
…… Fakültesi

Prof. Dr. …….
Kırşehir Ahi Evran Üniversitesi
…… Fakültesi

Prof. Dr. …….
…….. Üniversitesi
……. Fakültesi

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada bana ait olmayan her türlü ifade bilginin kaynağına eksiksiz atıf yapıldığını bildiririm.

ALI ABBAS JASIM

20.04.2016 tarihli Resmi Gazete'de yayımlanan Lisansüstü Eğitim ve Öğretim Yönetmeliğinin 9/2 ve 22/2 maddeleri gereğince; Bu Lisansüstü teze, Kırşehir Ahi Evran Üniversitesi'nin abonesi olduğu intihal yazılım programı kullanılarak Fen Bilimleri Enstitüsü'nün belirlemiş olduğu ölçütlere uygun rapor alınmıştır.

ÖNSÖZ

Yüksek Lisansa / Doktoraya başlamamda ve yüksek lisans / doktora ders sürecinde kendisini tanıdığım günden bu yana gösterdiği sakin ve sabırlı hali ile her zaman bana örnek olmasının yanı sıra bir bilim adamının nasıl çalışması gerektiğini kendisinden öğrendiğim değerli danışmanım Doç. Dr. Mustafa YAĞCI'e büyük bir içtenlikle teşekkür ederim. Tezimin her aşamasında gerek sorularımla gerekse alt ayda bir yapılan tez izleme komitesi sunumlarında tezin şekillenmesinde ve nihai hale gelmesinde katkıları olan değerli jüri üyelerim Doç. Dr. Nezih ÖNAL ve Dr. Öğr Üyesi AKSU'e teşekkürlerimi içtenlikle sunarım.

Son olarak eğitim hayatım boyunca bana her türlü maddi ve manevi desteği sunan çok kıymetli aileme sonsuz şükranlarımı sunarım.

Haziran, 2022 ALI ABBAS JASIM

CONTENTS

Page No

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviations	Explanation
SSD	: Single Shot Multi-box Detector
CNN	: Convolutional Neural Network
WHO	: World Health Organization
CDC	: Centers for Disease Control and Prevention
SIFT	: Scale-Invariant Feature Transform
HOG	: Histogram of Oriented Gradients
YOLO	: You Only Look Once
R- CNN	: Region-based Convolutional Neural Networks
ARL	: Army Research Laboratory
DARPA	: Defense Advanced Research Project Agency
FERET	: Face Recognition Technology
PCA	: Principal Component Analysis
LDA	: Linear Discriminant Analysis
AdaBoost	: Adaptive Boosting
ILSVRC	: ImageNet Large Scale Visual Recognition Challenge
GPU	: Graphics Processing Units
SOM	: Self-Organizing Map
PDBNN	: Probabilistic Decision-Based Neural Networks
SVM	: Support Vector Machine
SNoW	**:** Sparse Network of Winnows
HMM	**:** Hidden Markov Model
MRFs	**:** Markov Random Fields
DNN	: Deep Neural Network
VGG	: Visual Geometry Group
ResNet	: Residual Neural Network
XML	: Extensible Markup Language
ROI	: Regions Of Interest
FC	**:** Fully-Connected Layers
ReLU	: Rectified Linear Unit
RGB	: Red-Green-Blue
CAFFE	: Convolutional Architecture for Fast Feature Embedding
GUI	: Graphical User Interface

ÖZET

YÜKSEK LİSANS TEZİ

GERÇEK ZAMANLI YÜZ MASKESİ ALGILAMA UYARI SİSTEMİ

ALI ABBAS JASIM

Kırşehir Ahi Evran Üniversitesi
Fen Bilimleri Enstitüsü
İleri Teknolojiler Anabilim Dalı

Danışman: Doç. Dr. MUSTAFA YAĞCI

Gerçek zamanlı maske tespiti ve tahmini, iki aşamada kurulan derin öğrenmeyi kullanan araştırmaların ana hedefleridir; ilk aşama, görüntü sınıflandırması için önceden eğitilmiş bir MobileNetV2 ağı kullanarak bir modeli eğitmekti. Eğitim verilerimiz iki kategoriye ayrılmış (maskesiz maske) 1785 görüntüydü. İkinci aşamada, 'Single Shot Multi-box Detector' (SSD) dedektörü ve ResNet-10 mimarisini kullanarak gerçek zamanlı video akışlarında yüzleri algılamak için OpenCV kütüphanesini kullanıyoruz. Daha sonra bu sonuçları, birinin maske takıp takmadığını belirlemek için modelimize girdi olarak kullanırız. F1 puanı yüzde 100, model doğruluğu ise yüzde 99.72 idi. Eğitim verileri çeşitli kaynaklardan elde edilmiş ve tamamı internetten indirilmiştir.

Haziran 2022, 50 Sayfa

Anahtar Kelimeler: Yüz Maskesi, COVID-19, CNN, Transfer Öğrenimi, MobileNetV2

ABSTRACT

M.Sc. THESIS

REAL-TIME FACE MASK DETECTION WITH ALERT SYSTEM

ALI ABBAS JASIM

Kırsehir Ahi Evran University
Graduate School of Sciences and Engineering
Advanced Technologies Department

Supervisor: Asst. Prof. Dr. MUSTAFA YAĞCI

Real-time mask detection and prediction are the main objectives of research using deep learning, which was established in two stages; the first stage was training a model using a pre-trained MobileNetV2 network for image classification. Our training data was 1785 images divided into two categories (mask without mask). In the second stage, we use the OpenCV library to detect faces in real-time video streams using the 'Single Shot Multi-box Detector' (SSD) detector and the ResNet- 10 architecture. Then we utilize these results as inputs to our model to determine whether or not someone is wearing a mask. The F1 score was 100 percent, while the model accuracy was 99.72 percent. Training data was obtained from various sources, and all downloaded from the internet.

June 2022, 50 Pages

Keywords: Face Mask, COVID-19, CNN, Transfer Learning, MobileNetV2

1 INTRODUCTION

When we use social networking sites, we often notice that some things happen that surprise us, such as suggesting new friends, analyzing mutual friends, helping users discover new content, communicating with the things they care about most ... etc. These and many other things happen under the Machine learning term.

Machine learning is a discipline of computer science that investigates algorithms that can learn from their data and experience [1]. It's a part of artificial intelligence. Machine learning algorithms develop a model based on sample data, called "training data" [2], it is a preliminary set of data used to help a program understand how to apply techniques like learning neural networks to achieve complex results through which a computer learns how to process information without being explicitly programmed. A training set is often made up of a huge number of photos. Because the machine learning software is so complicated and smart, it performs iterative training on each of these images to eventually recognize characteristics, shapes, and even subjects such as humans, animals, letters, numbers, etc. Training data is critical to this process and can be considered the "fuel" that allows the system to function. The development of conventional algorithms to accomplish the required tasks is difficult or impossible in a broad range of applications, including medicine, email filtering, voice recognition, and computer vision, where machine learning techniques are utilized [3]. One of the scientific topics that have multiple disciplines is the topic of computer vision, which deals with the ability of computers to perceive and understand images and to automate the procedures that the human system can perform [4], which makes the computer deal with how it enables it to acquire a high-level understanding of a digital image or a series of images, analyze it and extract data that allows the computer to reach the right decision like the human mind works in distinguishing objects. video streams, multidimensional data from a scanner, medical scanning devices, etc., are all examples of image data. Engineering, physics, statistical analysis, and learning theory are used to deconstruct symbolic information from visual data and understand the image [5].

One of the most important of these disciplines is face detection, which has received great attention from studies and development due to its success in many applications, especially in the field of human-computer interaction and the field of security, in addition to many fields. In all face recognition applications, we find a major problem: how to detect the face first, as face detection is an initial step for the applications of faces. In contrast to the human ability to recognize faces effortlessly, the computer considers face detection a difficult task because the human face is dynamic and has a high degree of contrast in its appearance, as well as image resolution, light, face angle, and other variables from the varied nature of the object. As a result, we tried to find many techniques, from simple edge-based algorithms to complex high-level methods that use advanced pattern recognition techniques [6].

In Wuhan, the People's Republic of China, the World Health Organization (WHO) discovered the existence of the Covid-19 virus on December 31, 2019. The Covid-19 virus was later announced on March 11, 2020, as the emerging Coronavirus that causes "Covid-19 disease". Coronavirus spreads from an infected person to a healthy person through airborne droplets generated by sneezing, coughing, hugging, and kissing [7]. Therefore, all countries have taken preventive safety measures to limit the spread of the virus, such as social distancing, hand sanitizers, and wearing face masks. Wearing a face mask is a very important preventive measure, but rather the most important step to take when it is difficult to maintain social distancing, especially for people who suffer from chronic diseases or other health problems that put them at greater risk of infection with the Coronavirus. Research shows that the Coronavirus is mostly transmitted between persons who come into close contact. It may even be transmitted by people who do not display signs of illness and are unaware that they are sick [8]. Therefore, the CDC (Preventative Health Services of the United States) recommends that you continue to wear masks and keep a distance of at least 6 feet from each other, especially when you are indoors around people who do not live in your home [9].

However, manually imposing such a policy in large workplaces or public places and following up on individual violations is difficult, so computer vision is considered the alternative solution. Therefore, I developed an alert system to detect the face mask in real-time, consisting of two stages. The system looks to see whether a human face is there, and if so, it determines whether or not it is wearing a mask and then sounding an alert. The OpenCV Deep Neural Network, TensorFlow, Keras, and MobileNetV2 architectures are used to develop an image classification model.

To limit the spread of the Coronavirus, this classifier can be combined with surveillance cameras to be used to detect faces that are not wearing masks.

In the present suggested face identification methods, the detection of face masks remains a serious difficulty [10,11]. Because it relies on handcrafted feature extractors for non-frontal face recognition, Viola-Jones suffers from weak Haar features on non-frontal faces [12]. Scale-invariant feature transform (SIFT) [13] and histogram of oriented histograms (HOG)[14] are two examples of other methods that extract distinctive features.

Recently, object detectors based on deep learning have shown excellent and dominant performance in developing modern objects detection devices such as autonomous driving [15] and monitoring [16]. Deep learning is based on neural networks, a series of algorithms that seek to learn basic relationships in a set of data using mathematical functions, which collect and classify information according to a certain structure, which is similar to the work of the neural network for the human brain. The input, processing, and output layers are all components of a neural network. There are nodes and connections between these nodes, which are akin to neurons and synapses in the human brain; hence, the processing layer is concealed from view.

Neural networks can learn features in an overall way [17], like a convolutional neural network, it is configured to analyze and identify visual data such as digital images. There are one-stage and two-stage deep learning-based object detectors, one-stage detectors employ a single neural network to identify things, such as the Single-Shot Detector (SSD) [18] , you only look once (YOLO) [19]. and the finding of the two-stage discovery can be accomplished using a variety of networks, including the area-based convolutional neural network (CNN) (R-CNN) [20,21].

1.1 Purpose

In light of the outbreak of the Coronavirus, which caused the suffering of governments around the world to control this disease and limit its spread, according to the World Health Organization, protection from COVID-19 infection is critical. In light of the preventive steps taken to limit the spread of the Coronavirus, wearing a face mask has become one of the most important preventive measures that prevent the spread of the disease among people, as it is spread through the air through sneezing and coughing. A face mask detection warning system using deep machine learning is presented in this paper. There are two parts to this application.

The first section will build a face mask detector model trained on suitable data to perform this purpose. The second section will apply this trained model from the previous stage to a live video camera to identify people in real-time. This app can be used in public and private places by connecting it to surveillance cameras to ensure that the precaution (wearing a mask) is taken to stop the virus from spreading.

2 GENERAL SECTIONS

2.1 Face Detection

Face detection and recognition have led to many studies due to their wide range of uses.

Early attempts at computer facial recognition date back to the 1960s. Three researchers focused on developing a computer program to recognize human faces [22]. "Human and Machine" was the name given to the project where the person had to first identify the facial traits in the image before the computer could utilize them for identification. This is done by drawing facial characteristics such as the centers of the pupil, the inner and outer corners of the eye, the width of the mouth, and the looks on a graphic tablet by a person. With this method, a person can analyze approximately 40 images per hour and create a database of these distances. Then the image distances are automatically compared by a computer, which calculates the difference in lengths and presents the closed records as a possible match [23].

Using facial anatomical traits and a computer algorithm, Takeo Kanade developed the facial matching system in 1970 and was the first to assess the distance ratio between facial features without the assistance of a human. It turns out that it is not always possible to correctly determine the characteristics of the face. In 1977, Kanade [24] published the first complete book on the subject of facial recognition technology, despite this growing interest.

The Defense Advanced Research Project Agency (DARPA) and the Army Research Laboratory (ARL) founded the FERET face recognition technology program in 1993 to develop "automatic face recognition capabilities" that can be used in a "real-life environment produced to assist with security and intelligence." The accuracy of current automatic facial recognition systems varies; some algorithms can identify people in still photographs shot in a controlled environment [25]. The FERET test resulted in the creation of Vision Corporation and Miros Inc. in 1994. For the commercialization of Alex Pentland's facial recognition algorithm from MIT, a military contractor founded Viisage Technology in 1996 [25].

Until the 1990s, the primary inputs to developing facial recognition systems were images of real people's faces. Principal Component Analysis (PCA) was employed in the early 1990s in facial recognition studies to detect a face in a picture that included other items. Mathew Turk and Alex Pentland [26] invented this technology, also known as Eigenface. An Eigenface model was created using the Karhunen-Loève theory and component analysis to identify

Eigenfaces based on the characteristics of the general facial features. Turan and Pentland's PCA facial recognition technology uses fewer Eigenfaces than other methods for representing human faces. In 1994, Pentland made the Eigenface features to help improve the use of PCA in face recognition.

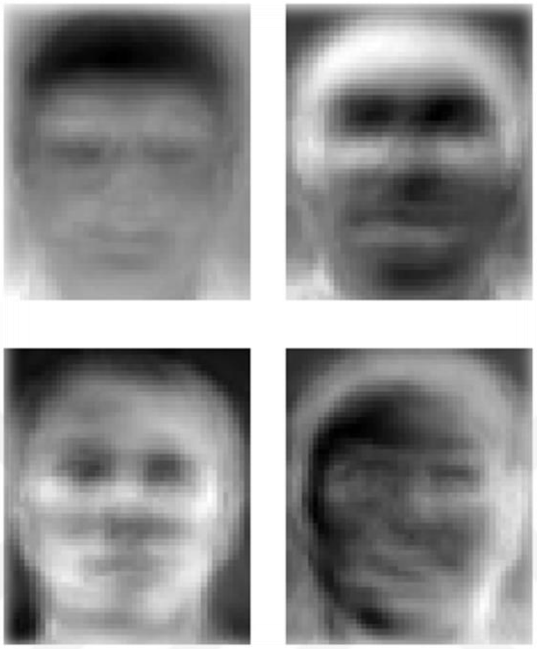

Figure 2-1. Some eigenfaces from AT&T Laboratories Cambridge [27].

The Fisherfaces were created in 1997 by applying linear discriminant analysis (LDA) to improve the PCA Eigenface technique of face identification [28]. PCA feature-based face identification began to rely heavily on LDA Fisherfaces [29]. Face reconstruction techniques that use Eigenfaces do not have a global structure that unites individual facial traits or parts[28].

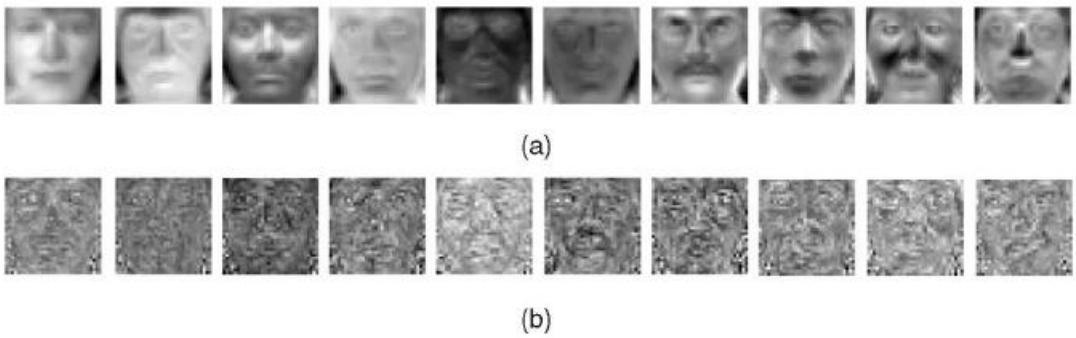

Figure 2-2. (a) Eigenface, (b) Fisherface [30].

By using Gabor filters to gather face data, the Bochum system was able to generate face structures that linked features together [28]. In the mid-1990s, Christoph von der Malsburg and his colleagues at the University of Bochum developed Elastic Bunch Graphic Matching to recover a face from a picture using skin segmentation [26]. Operators of airports and other congested sites purchased the face detection "Bochum system" under the trade name ZN-Face. The program proved "robust enough to produce identifications from less-than-perfect facial images," according to the author. "Mustaches, beards, new haircuts, and even sunglasses may all be seen through this technology" [31].

Real-time facial identification in video data was made possible in 2001 by the Viola-Jones facial-detector framework [26]. Paul Viola and Michael Jones have developed the first real-time face detection, AdaBoost. For object detection in digital photos, they merged face detection technology with a Haar-like feature approach [32]. As recently as 2015, the Viola-Jones approach was applied to mobile devices and embedded systems employing small, low-power detectors. Face recognition systems have been given a practical boost by incorporating this technology into their user interfaces and teleconferencing capabilities [33].

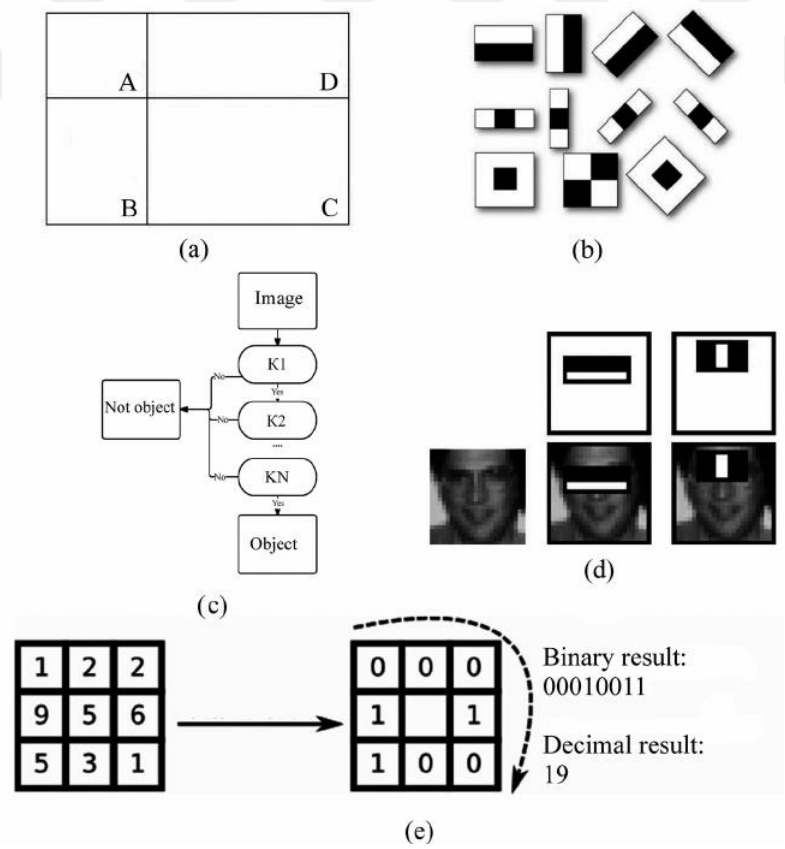

Figure 2-3. Viola-Jones algorithm parts: (a) combination of regions, (b) Haar Features, (c) cascade classifier, (d) Haar feature application to the image, and (e) LBP feature[34].

Requests from academics and industry executives to challenge ImageNet Large-scale Visual Recognition (ILSVRC) between 2010 and 2017 resulted in some of the most notable advancements. An annual computer vision competition, the ILSVRC, was built using a subset of the publicly available ImageNet data set. To encourage the development of improved computer vision technologies and to measure the most recent technology ,

Image classification challenges include labeling images based on the primary item in the image and detecting objects within images [35].

In the first five years of ILSVRC, the rate of development was surprising to the computer vision community. Deep learning has gotten a boost from the development of substantial convolutional neural networks (CNNs) running on graphics processing units (GPUs) [35].

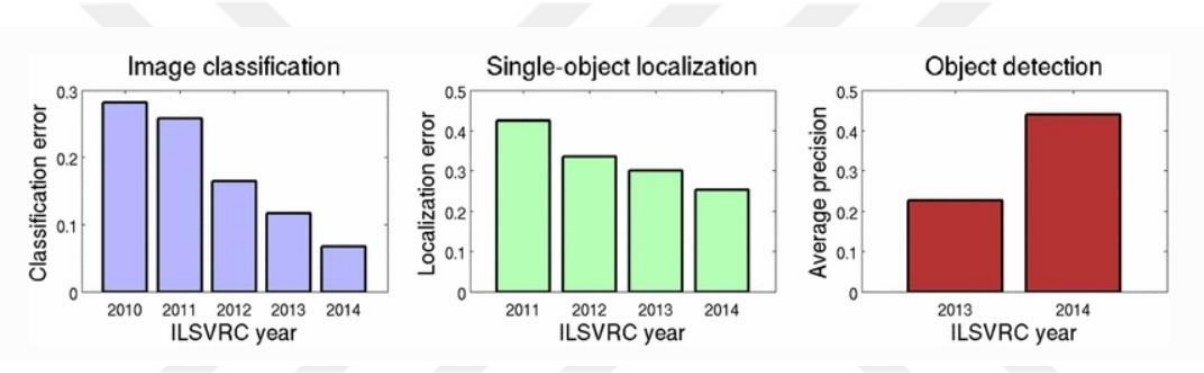

Figure 2-4. Summary of the Improvement on ILSVRC Tasks Over the First Five Years of the Competition.Taken from ImageNet Large Scale Visual Recognition Challenge, 2015 [36].

2.1.1 Face Detection Methods

Yan, Kriegman, and Ahuja [37] grouped face detection systems into four categories: The algorithm may fall under more than one of these categories.

2.1.1.1 Knowledge-Based Top down Methods

It is controlled by a set of principles. Humans' understanding of faces and how to distinguish them as a set of rules is included in this system. For example, the eye area is darker than the cheeks. This, in addition to the difference in color intensity between the eye area and the lower area and the distance between the eyes, is all considered features of the face. The problem with these approaches is that they make it challenging to develop an appropriate set of rules because general rules lead to many false positives. In contrast, precise rules lead to many false negatives.

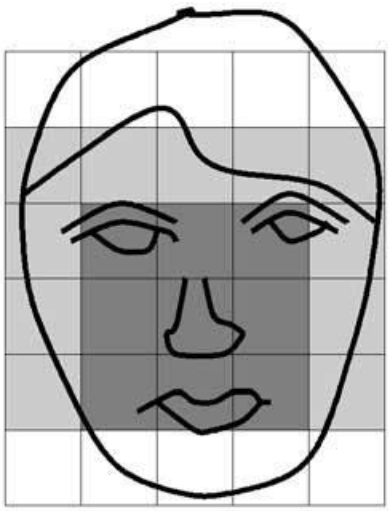

Figure 2-5. Typical face used in knowledge-based methods [38].

For solving these issues, a hierarchical strategy based on knowledge is required. This is a somewhat constrained strategy—the inability to locate multiple faces in a detailed image.

To find faces, Yang and Huang [36] apply an approach based on hierarchical knowledge. There are three stages to their process. A series of criteria are used to identify possible faces in the input picture, as indicated in the figure (2-8).

Figure 2-6. Multi resolution hierarchy to locate the face [39].

After performing a local graph equation on the first-level candidates, the second set of criteria is used to assess the remaining candidate areas that react to characteristics like the mouth and eyes [37].

2.1.1.2 Bottom-Up Feature-Based Methods

This strategy, unlike the knowledge-based method, is based on our natural understanding of faces. A person can recognize the face and other objects in various situations, and whatever the lighting conditions, this technology goes beyond the limits of this understanding. So she

uses an edge detection approach to extract facial features, including eyes, eyebrows, mouth, and nose, in this method. Because of things like lighting and noise, these algorithms could hurt the quality of image features [37].

2.1.1.3 Template Matching

The face can be divided into areas such as the contour of the face, eyes, lips, and nose. A standard template or function parameters are used to try to define this method's parameters for all faces. The face can be figured out from the picture by calculating the correlation values for the eyes, mouth, and nose separately and then comparing the results.

Despite its simplicity, this strategy cannot be used for face identification because this method is confined to the front and undistorted faces, and its inaccuracies when there are variances in form, location, or size. On the other hand, Deformable templates are being considered a possible solution for this problem [40].

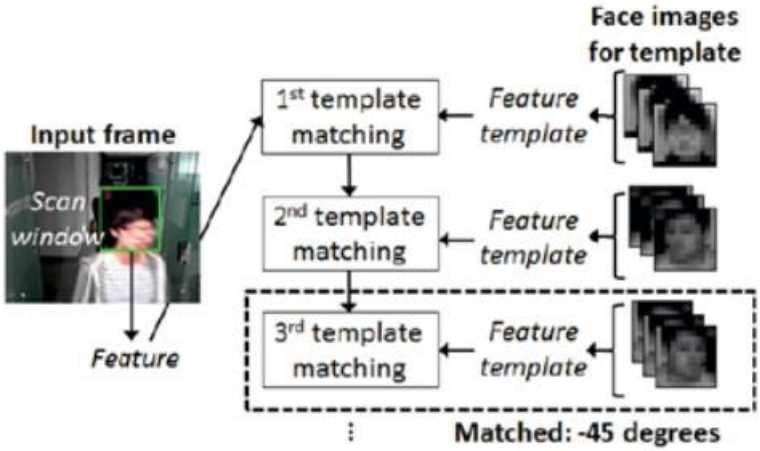

Figure 2-7. Template Matching [41].

2.1.1.4 Appearance-Based Methods

This performance method is better than the others. In contrast to template matching strategies, which use pre-defined templates, this strategy teaches templates through examples from photographs. This approach works on features acquired in models or discriminatory functions, which are then used for facial recognition using statistical analysis and machine learning. The most suitable methods and tools will be mentioned below:

2.1.1.4.1 Eigenface-Based

Eigenvectors for facial identification were shown by Kohonen [42] using a neural network that approximates the image autocorrelation matrix by approximating the eigenvectors (Eigenfaces) are the names given to these vectors.

As proven by Sirovich and Kirby [43], utilizing PCA, face pictures may be encoded using a small number of basis images (Principal Component Analysis). Figure 2.10 shows an example of an eigenface, which was encoded using just 50 Eigen pictures in experiments using a collection of 100 images. A recognition method based on eigenfaces was later developed by Turk and Pentland [44].

Figure 2-8. Eigenfaces [39].

2.1.1.4.2 Distribution-Based

Sung and Poggio [45,46] initially proposed a distribution-based approach to face detection, the method consisting of distributed models of face patterns and a multi-layer classifier trained to reliably assess the status of the target pattern category based on distance measurements.

2.1.1.4.3 Neural Networks

Many pattern recognition applications, including object identification, optical character recognition, and robotic self-driving, have been successfully implemented using neural networks. Several different neural networks have been presented to deal with face detection as a pattern recognition issue with two classes. It is important to use face detection neural networks to try a system to recognize complicated patterns in the density of facial features.

As a result, optimal performance necessitates a wide range of configuration options (number of layers, learning rates, nodes... etc.). According to Agui et al., an early approach combining two stages of neural networks can recognize faces when all test pictures have the same size [46]. Propp and Samal [47] created one of the first neural networks for face recognition. The early approach of scanning an input picture using a time-delayed neural network was later proposed by Souli et al. [48,49]. Vaillant et al. [50] utilized 20 x 20-pixel pictures with and without faces to train their convolutional neural network to recognize faces. With the help of Kohonen's SOM algorithm [42], Burel and Carel[51] presented a neural network for face identification that compresses and reduces the number of training instances. An associative neural-network-based detection approach was reported by Feraud and Bernier [52,53,54]. Lin and his colleagues have developed a decision-based probabilistic neural network (PDBNN) to extract trait vectors from density and edge information in the facial area, including the eyebrows, eyes, and nose [55]. Rowley et al [56,57,58] work is the most significant of all face detection systems. Face and non-face images were analyzed using a multi-layer neural network. First, the neural network obtains an area of 20 x 20 pixels from the image and produces values from -1 to 1. This means that for non-face samples, the value is -1, and for face samples, the value is 1. Secondly, the logical operators (and/or) and voting are simple arbitration procedures. Here is a look at the algorithm design, as shown in Figures (2-11).

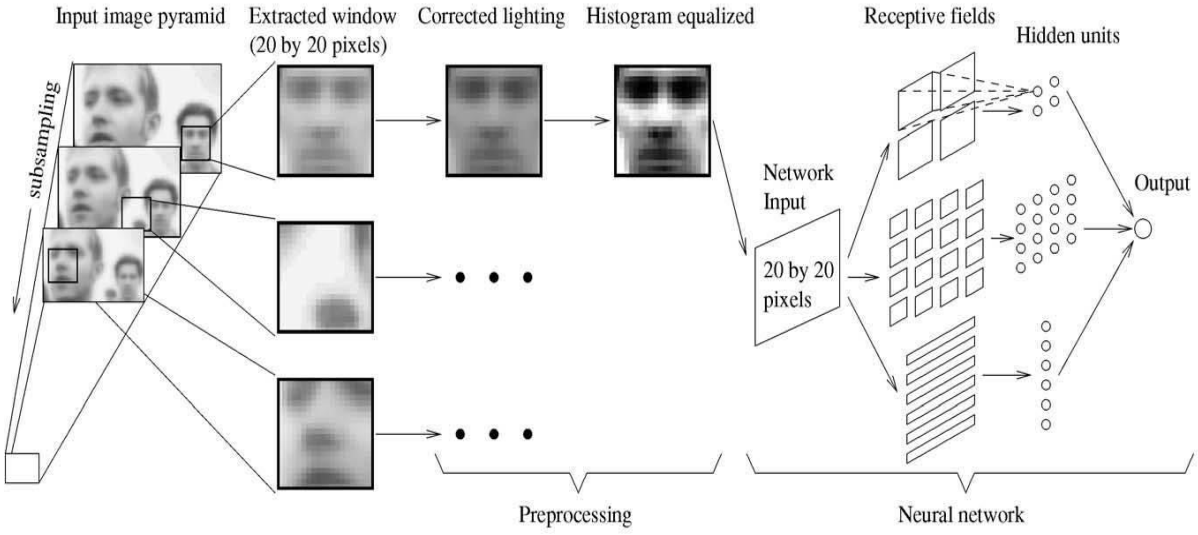

Figure 2-9. Diagram of how Rowley's method works [58].

2.1.1.4.4 Support Vector Machines

Statistical classification and regression analysis may be performed using SVMs, which are linear classifiers. The method begins with the sorted (encoded) data. Training then proceeds

to discover a linear framework. For the first time, SVMs were used to identify faces by Osuna et al.[59].

2.1.1.4.5 Sparse Network of Winnows

Using the SNoW (Sparse Network of Winnows) learning architecture for the first time, Yang et al. [60] introduced a new approach to recognizing faces with diverse expressions and features in various lighting circumstances and the form of two linear units or target nodes [61].

2.1.1.4.6 Naive Bayes Classifier

Schneiderman and Kanade et al [62] made this suggestion. Based on the theory of Bayes, the Naive Bayes algorithm is a classification method in which the predictors are assumed to be independent of each other. It is based on the Naive assumption that features within a class are unrelated to one another that Naive Bayes' classification model is based. It may be an apple when the fruit appears crimson, spherical, and about three inches in diameter. This fruit's Naive Bayes classifier will take all of these qualities into account when determining the likelihood that it is an apple.

When dealing with giant data sets, the Naive Bayes technique is ideal. Naive Bayes outperforms even more advanced classification systems because of their simplicity.

2.1.1.4.7 Hidden Markov Model

It is possible to characterize patterns as a parametric random process, and the parameters can be determined precisely and clearly. Making it as likely as possible that the HMM training data may be viewed is the purpose of the Viterbi segmentation and Baum-Welch methods, which are frequently employed in HMM training [63].

2.1.1.4.8 Information-Theoretical Approach

Markov Random Fields (MRFs) may analyze facial patterns and other associated characteristics. Using Kullback-Leibler divergence, the Markov process maximizes the discrimination across classes. As a result, Face Detection may be performed using this approach [64].

2.1.1.4.9 Inductive Learning

One of the inductive learning algorithms, Quinlan's C4.5 algorithm [65], uses positive and negative instances of face patterns to create a decision tree [66].

2.2 Deep Learning

It is a way of teaching machines to do activities as if they were humans. By stimulating neurons in the human brain, this strategy aims to discover theories and methods that enable devices to learn by themselves and find ways to extract features from large data sets using linear and nonlinear variables [67,68] The basic idea of deep learning is that any object in an image can be described in several ways, such as using the brightness vector for each pixel or the sum of the edges and areas that make up the image, in addition to many additional ways that can be used to describe these images. This is the essence of deep learning. In terms of machine learning, some of these strategies (such as studying a face or noticing expressions) [69] outperform others. Because of this, people who study deep learning want to eliminate the need for human intervention in feature elicitation and replace it with algorithms that generate features automatically or almost automatically [70]**.**

We're seeing a rise in the use of convolutional neural networks (CNNs) and deep neural networks (DNNs) across a wide range of applications like computer vision and voice recognition [3,71,72,73].

2.3 OpenCV

Real-time computer vision is the primary focus of this library, and it is open-source [74]. Intel developed it under the Apache 2 license. For real-time applications, OpenCV is used for GPU acceleration [75]. Its primary interface is written in C++ as well. It works on all desktop operating systems and mobile operating systems like Android, iOS, and Maemo [76,77]**.**

2.3.1 OpenCV-DNN

A highly improved DNN model with deep learning functionality has been added in the OpenCV3.3 version. This module can perform a wide range of computer vision tasks (such as object identification, face detection, image classification, text detection, and recognition) using pre-trained models such as Caffe, TensorFlow, Torch, and PyTorch, which are some of the frameworks for deep learning supported by this module.

2.3.2 Single Shot Detector (SSD)

In contrast to two-stage models, the single-shot MultiBox detector can detect many objects in an image in one step, making it faster and more efficient than two-stage techniques [78]. The SSD consists of two parts: the backbone model and its head. Pre-trained image classification networks, such as VGG16 or ImageNet-trained ResNet, are used as feature extractors and is the backbone. As for its header, one or more convolution layers are added to the spine, and their outputs are interpreted as bounding boxes and object classes in spatial position to animate the final layers [18].

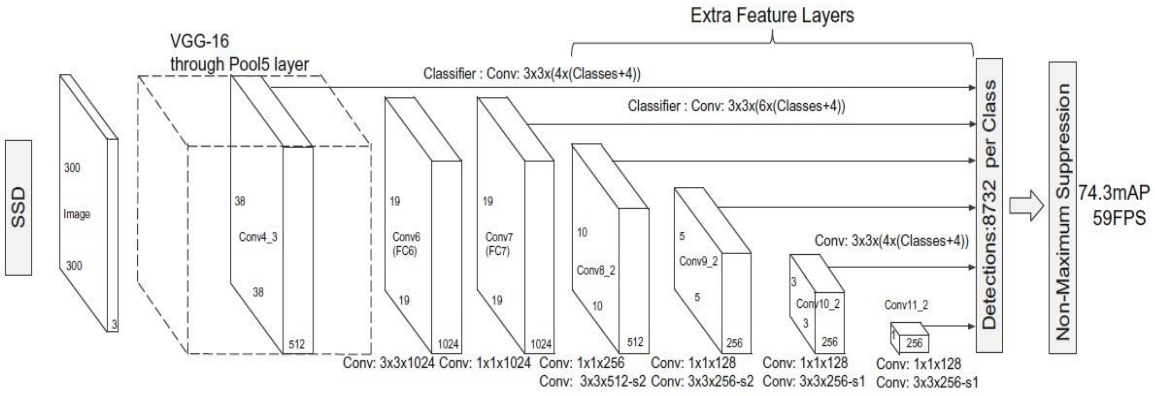

Figure 2-10. Single Shot Detector (SSD) [18].

It uses more than one feature map to define different sizes of objects, so we note in the above figure that there are six other feature maps to reach the final result. Despite his excellent performance, he fails in his ability to recognize small things. Because many grouping layers can drastically reduce the size of elements smaller than 36 x 36 pixels, the detection section of the SSD lacks sufficient spatial information to distinguish small details in the surrounding environment [78].

3 MATERIAL AND METHOD

3.1 Work Description

Real-time mask detection and prediction are the two main objectives of the research. The project will be divided into two phases to achieve this purpose. The training dataset will be described in Section 3.2.

As a first stage, this data will be passed to the model to train it to classify images (faces wearing masks and non-masked faces). The MobileNetV2 [79] neural network was used for this purpose. After completing the model's training, we will move to the second step, where we will use the trained model from the first stage with an accurate face detector to predict in real-time whether a person is wearing a mask or not. The Multi-Box Single Shot Detector (SSD) [18] with ResNet-10 architecture [80] was used for real-time face detection due to its speed and accuracy in detecting faces. After real-time face detection, the regions of interest (ROI) for each face will be cropped and passed through our model, which we trained to identify the faces that wear masks from the faces that do not wear masks.

The OpenCV library was used because it is the best computer vision library which allows us to load pre-trained networks from which to load the face mentioned above detector model.

There are two primary stages of training on the mask detector model, and each phase includes sub-steps, as shown in Figure 3.1:

- Training: Loading data from the hard disk, training the model using Keras and Tensorflow, and then saving the model to the hard disk.
- Deployment: Once the mask detector model has been trained, we will upload it and do face detection in real-time. It will decide for each face whether it is wearing a mask or not.

This work was suggested using an Acer laptop with an Intel Core i7-8565U processor, NVIDIA GeForce MX130 graphics card, Python 3.9 plus PyCharm 2021.3.3, and the latest version of Python.

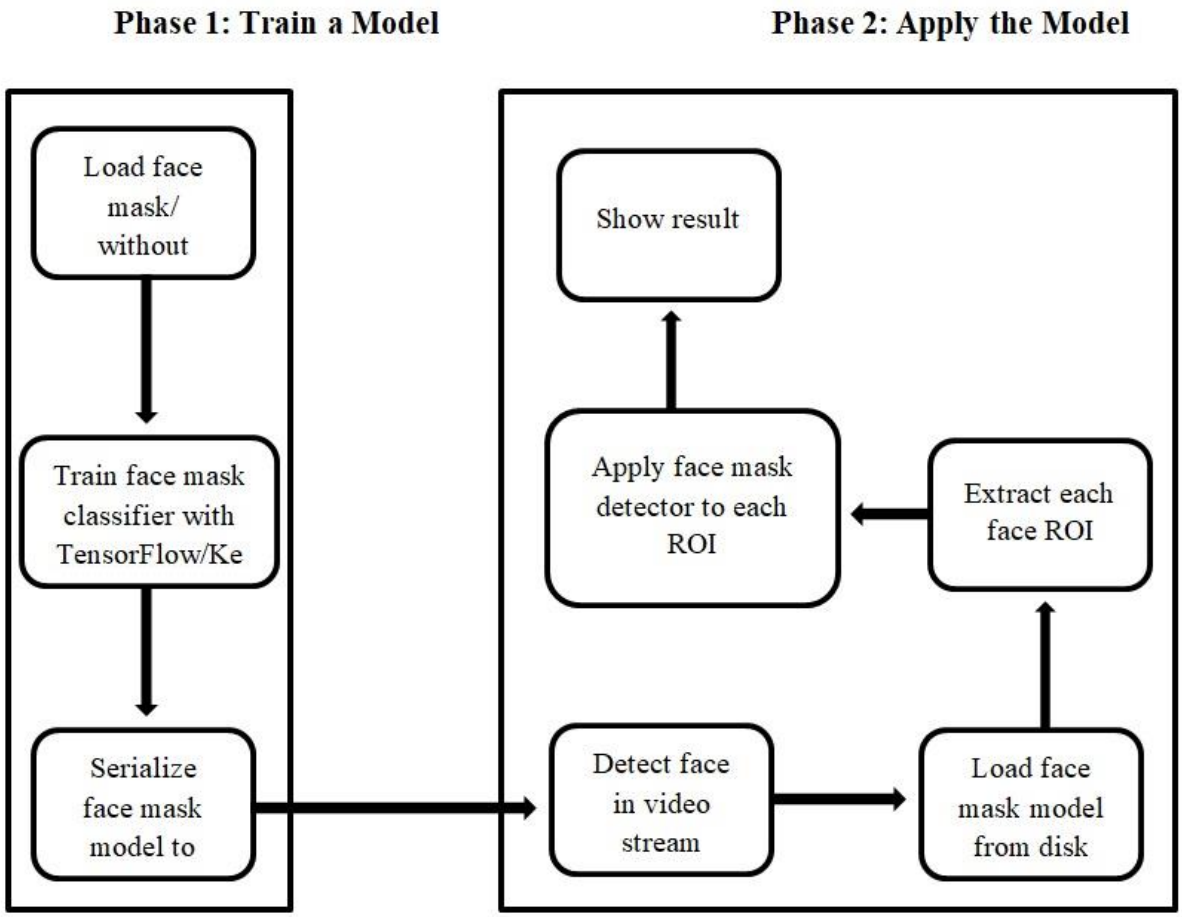

Figure 3-1. The structure of project

3.2 The Dataset

Significant amounts of appropriate training data are required to acquire the most remarkable results from any computer machine learning application. The more training data there is, the more accurate the application.

We require two sets of training data for our application: one set of photographs of faces without masks and the other images of faces with masks.

An open-source and varied collection of photos was utilized for training a model, using data from the Prajna Bhandary dataset at PyImageSearch [81] and the Kaggle Medical Mask dataset by Mikolaj Witkowski [82].

Kaggle data consists of 853 images and XML files that describe individuals who wear or do not wear medical masks.

There are 660 images with masks in the "Prajna Bhandary" dataset and 660 photos of faces without masks. Faces and facial features were standardized by Prajna Bhandary (described in section 3.2.1). Unmasked data images compensate for error correction so that the model is not overly affected. The data set is shown in Figure (3.2).

Figure 3.2. The Dataset

3.2.1 How Was Face Mask Dataset Created?

Due to the lack of data available for people wearing masks, Prajna Bhandary devised a method to create this dataset:

- She took a typical face picture.
- She made a computer vision program to add the mask to the face.

The characteristics of the face, which include the eyebrows, eyes, nose, mouth, and jawline, may be determined using this approach.

Figure 3-3. An image containing a face

First, she uses face detection to find the face in the picture. Then, she figures out the square coordinates that surround the face.

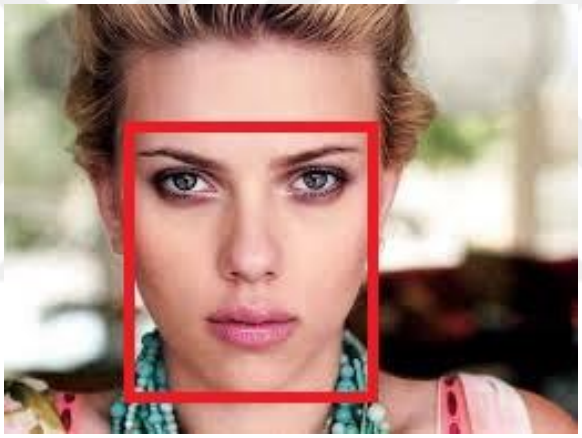

Figure 3-4. Using face detection.

To extract the ROI, we need to know where the face is located in the picture.

Figure 3-5. Extracting a face ROI.

After that, it will work with the face's contours to give the eyes, brows, nose, and lips more definition.

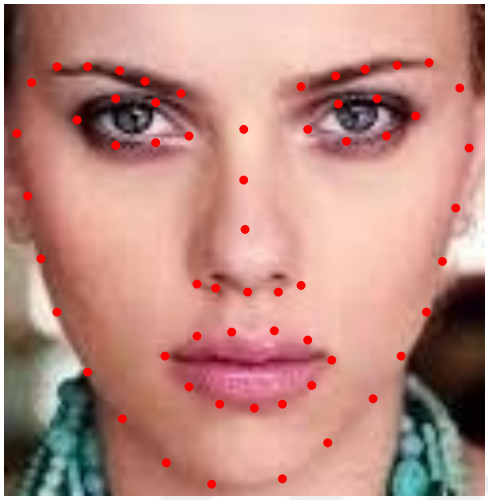

Figure 3-6. The features of the face.

After revealing the features of the face, it became known where the mask would be placed.

Figure 3-7. An example of the mask used to combat the Corona virus.

The mask is automatically applied to the face to determine where to place it (the points along the chin and nose). The mask is then scaled and rotated to fit the upper part of the face and then applied.

Figure 3-8. Putting the mask on the face

The same method is performed for all input images to create an artificial dataset for a face mask.

3.2.2 Pre-Processing

The data in this set contains a large amount of noise and redundancy in large numbers. Because the accuracy of the trained model depends on its training data, the data was manually processed and cleaned to remove any incorrect images detected in the data set and any duplicates. The data were divided into two groups: one with mask images and the other without a mask. Removing defects, distortions, and errors from the training data eliminates harmful effects on any predictive model.

The dataset with masks became 1270 images, and the dataset without masks became 515 images. The total number of images in the training data set is 1785.

3.3 Train a Model

3.3.1 Load Face Mask/ Without Mask Dataset

3.3.1.1 Load Dataset and Pre-Processing

Keras and Tensorflow will be used to train our data. Thus, we'll specify our training interval, starting learning rate, and batch size, utilizing these two libraries as a framework and a backend, respectively. A loop is created to read all of the images and perform preprocessing on them, such as resizing them to 224 x 224, converting them to Matrix format, and altering

the pixel intensity of the input image to a range (1, -1). The preprocessed images and label data are added to the lists (data and labels). For fast computations, these lists are transformed into NumPy arrays.

3.3.1.2 Data Augmentation

There is not enough data to successfully train the model, and therefore a large amount of data is needed. The solution to this problem is to use data augmentation techniques. It is a data augmentation method by adding duplicates of the training data already used to train the model with changes such as permutation, shearing, noise injection, rotation, and random scanning. This procedure is standard in deep learning [83]. An image data generator called ImageDataGenerator can be used by the Keras library to apply accidental changes to the training data and then train CNNs on the new dataset, replacing the original training data set. The practical benefit of this procedure is to increase the model's accuracy, as the more remarkable the amount of training data, the higher the model's accuracy.

3.3.2 Training Face Mask Classifier With Tensorflow/Keras

3.3.2.1 Classification of Images Using Mobilenetv2

To make a model that can distinguish faces that wear a mask from those that don't, the learning must be transferred through deep understanding. Using the Keras library, knowledge is imparted through:

- Take a pre-trained convolutional neural network on a data set.
- Use it to identify the data category that you have not trained.

In general, there are two types of transfer learning:

- Feature extraction: Our features are extracted using a pre-trained grid that allows the input image to advance until it reaches a predefined layer, then stops there, taking the output of this layer as our features.
- Fine-tuning: We update the structure of the model by removing the vertices of the previous fully connected layers, adding new vertices, and then training the new FC layers to predict our input classes.

MobileNetV2 is a deep convolutional neural network. We'll utilize it for this task. More than a thousand different items that this network can classify into photos. More than a million photos from the ImageNet collection were used in training [84].

We load the network with pre-trained ImageNet weights and apply the new FC head to the base instead of the old header. The network's base layers are frozen, and new trainable layer inputs are added, which are then trained on our data to determine the features used to classify the faces wearing masks. Using a pre-trained network saves time because it can use biased weights without losing features it has already learned.

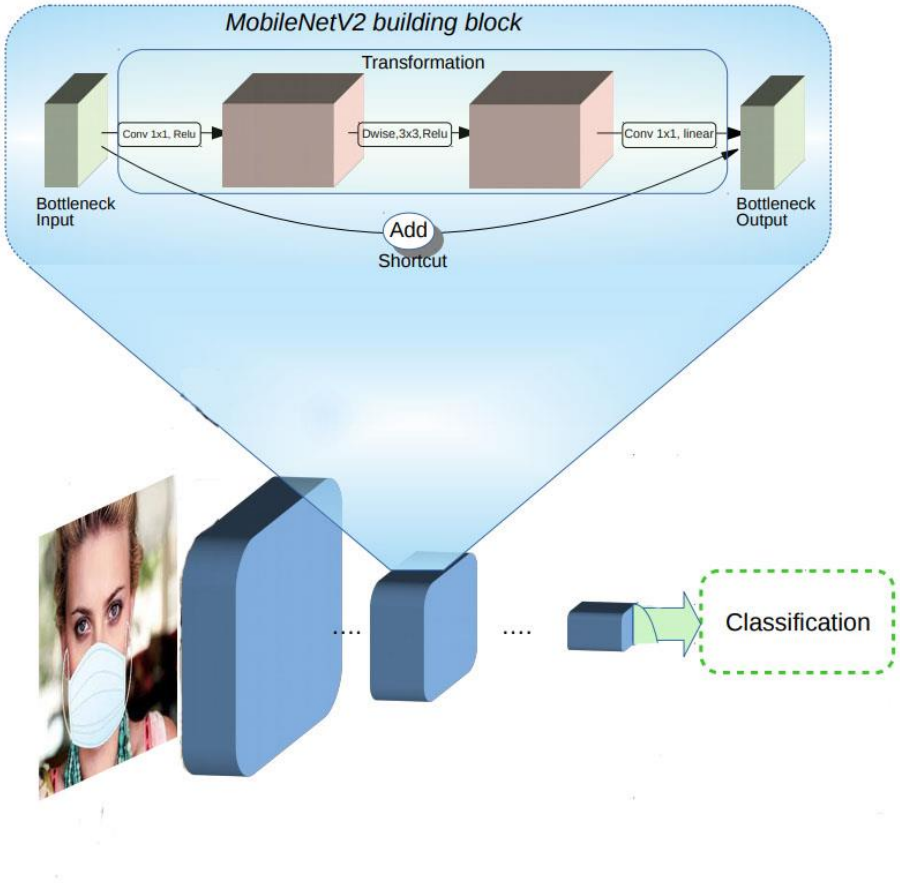

Figure 3-9. MobileNetV2 Architecture [85].

3.3.2.2 Convolutional Layer

Filters (or kernels) are the building blocks of a convolutional neural network (CNN). Filters smaller than the input image are used to build feature maps that summarize the presence of features in the input picture. There are fewer rows and columns in filters than in input images.

But they are also 3D, with the same number of channels. As a result, a 2D feature map of the activations is generated by applying the filter iteratively to different image areas. The filter's spatial positions are calculated using the input picture's point products, followed by the image's height and breadth and their point products. The filter has weights that must be learned during layer training. There are two ways to measure how well a design or feature has been found: filter weights and activation strength.

Combining two functions yields a new process called a convolution [86].

Feature map = input * filter = $\sum_{y=0}^{columns}(\sum_{x=0}^{rows}) \; input\;(x-p, y-q) filter\;(x,y)$ (1)

The input picture is sent through the network, which has dimensions of (224,224.3), that is, (150528) pixels in a one-dimensional feature vector, and each pixel is coupled to a neuron of the next layer. Then it is trained to detect whether a person is wearing a mask or not. Each neuron in this layer is connected to a neuron in the following layer, and so on until we reach the final call, which has a value of (0,1), indicating whether or not the person is wearing a mask.

Because this procedure takes a lot of time and calculations, mainly if the training data is vast, it may be decreased by stacking the feature maps for each filter to provide a larger output volume of the convolutional layer. Each feature map is regarded as an output of one nerve cell. Because the filter size small than size of the input picture, each neuron is linked to a tiny, local region. Low-level picture features (like lines) are captured by the first convolutional layers, whereas higher-level characteristics are extracted by the following layers [87] (such as selected shapes and objects). Figure (3.10) illustrates the convolution process.

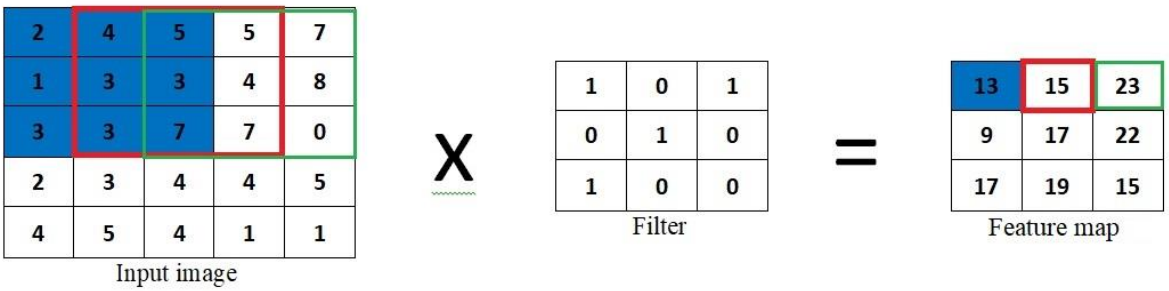

Figure 3-10. The convolution process.

3.3.2.3 Pooling Layer

The pooling layer follows the convolutional layer. It speeds up the calculations. Each feature map is processed separately with the pooling layer to create a new grouped feature map. This reduces the size of the array without losing any features.

Here are some examples of pooling operations:

- Max pooling: Takes the most significant output value present in a given area as the input value of the cell in the new array, based on that cell's output value.
- Average pooling: takes the selected area's arithmetic average and applies it to each cell in the new array (Figure 3.11).

In this project, the AveragePooling2D filter was used. Since it is more potent at summarizing spatial information [88], the aggregation size was 7x7. Each feature map will be shrunk down to a new, clustered 7 x 7 dimension feature map.

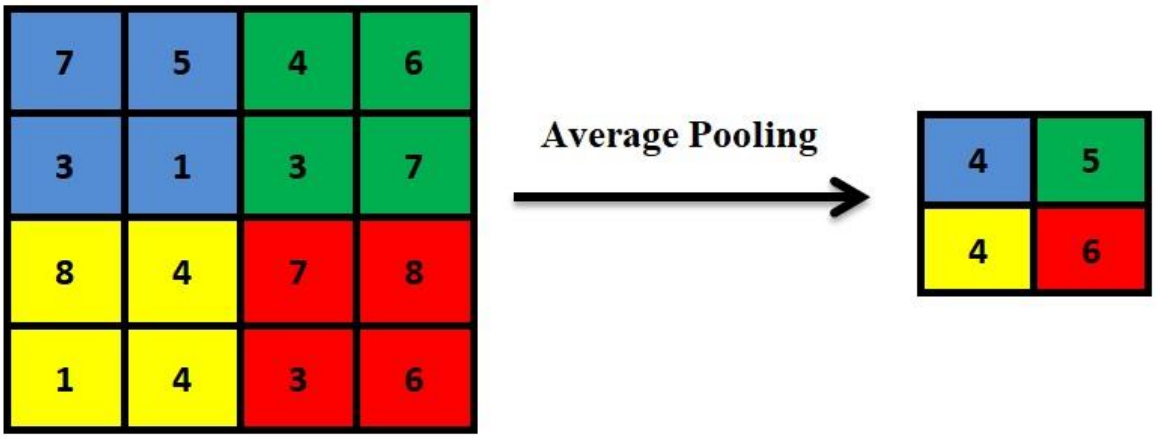

Figure 3-11. The average pooling operation.

3.3.2.4 Flatten Layer

Once the model has learned the features, it is ready to flatten the final feature map and connect it to the neural network (dense layers) for classification and prediction. But the input for the flattened layers is a 1-D vector, while the current is a 3-D tensor. Because of this, flat layers are used to turn the data into a one-dimensional array that the next layer can use. For instance, a 3x3 image array can be turned into a 9x1 vector. As can be seen in Figure (3.12),

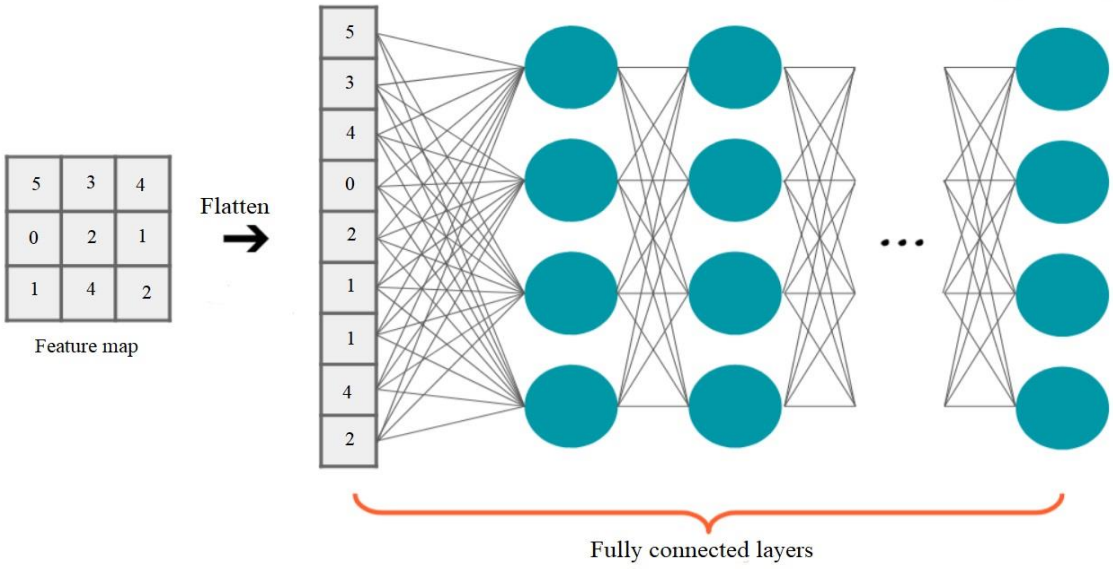

Figure 3-12. Flatten layer

3.3.2.5 Dense Layer (Non-Linear Layer)

The neurons in this layer are tightly linked to the neurons in the previous layer. Upon receiving an output from each neuron in the previous layer, neurons in the dense layer multiply the matrix-vector by each neuron in their layer (a procedure in which the row vector of the output from the previous layers is equal to the column vector of the dense layer). When multiplying matrices with vectors, the row vector must have the same number of columns as the column vector. This layer has the most common nonlinear functions, such as Rectified Linear Unit (ReLU) [89], Leaky ReLU [90], Noisy ReLU [91], Exponential ReLU [92], etc.

3.3.2.6 Dropout Layer

This layer reduces over-allocation during the training process by projecting bias neurons, which can be part of both the visible and hidden layers. The dropout rate can be modified to alter the likelihood of dropping a neuron.

3.3.2.7 Fully-Connected Layer

These layers have complete connections to the activation layers. It helps classify selected images into a multi-category or binary classification. SoftMax was employed in the activation functions used in these layers to obtain two parameters (0, 1).

3.3.3 Serialize Face Mask Model To Disk

The model was assembled using Adam's optimizer, learning rate decrement table, and binary entropy. Then the face mask is trained, and predictions are made on the test set; then, we find an index with the most considerable expected probability corresponding to each image in the test set, then we make a well-formatted classification report, and finally, the model is saved to the hard disk for use in the second phase of our project.

3.4 Apply the Model

3.4.1 Detect Face in Video Stream

As our project goes on, once we're done building and saving the model, we'll use the OpenCV library to find facial features in a webcam video and then run those features through our model to see if the person is wearing a mask.

We'll use the VideoStream widget from the imutils library, which displays local video streams (like a webcam) and provides access to video stream data from Python. Before putting them through our model, we will first create a function called (detect_and_predict), in which we will configure and process video frames so that we can read them later. This function has three parameters:

- frame: It will contain the video streams from our webcam.
- faceNet: It will include the pre-trained model for face detection (SSD).
- maskNet: It will consist of the model we trained in the previous stage to detect the face mask.

3.4.1.1 OpenCV-DNN-blobFromImage

As mentioned earlier, pre-trained models that use frameworks such as TensorFlow and Caffe are supported by the DNN module in OpenCV. So we will first process the image using OpenCV's blobFromImage() function to perform inference from pre-trained models. Then we use this blog to input the pre-trained models to get the inference output, as will be explained in detail.

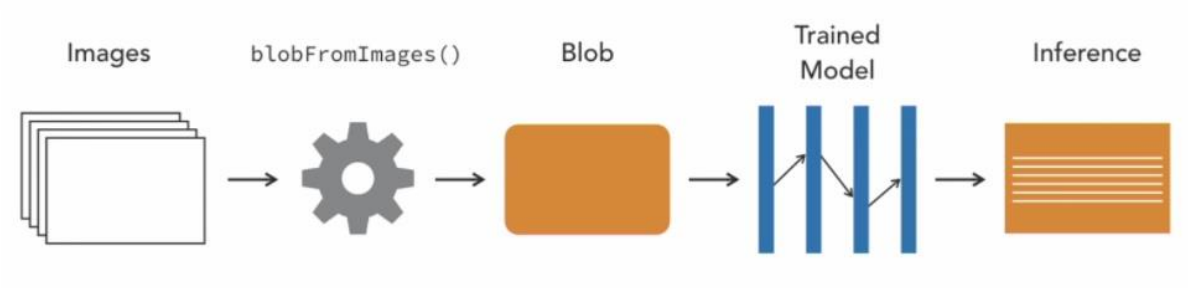

Figure 3-13. blobFromImage [93].

First, we must preprocess the data to get reliable predictions from deep neural networks. Four-dimensional images may be created by using blobFromImage. In addition to resizing and cropping from the center, this tool may also mean subtracting and scaling values by a scale factor and swapping color channels. Pre-trained deep learning models may be used to analyze photos and prepare them for categorization. This function carries out the following tasks:

- Mean subtraction
- Scaling
- Channel swapping

Our convolutional neural network will benefit from average subtraction to overcome the lighting conditions in the input picture. So that our gradients don't grow out of hand, each feature will have a comparable scope throughout this procedure.

In other words, if we don't scale our inputs in a manner that leads to similar-range feature values, the weight will be enormous in one region of the picture and extremely little in another.

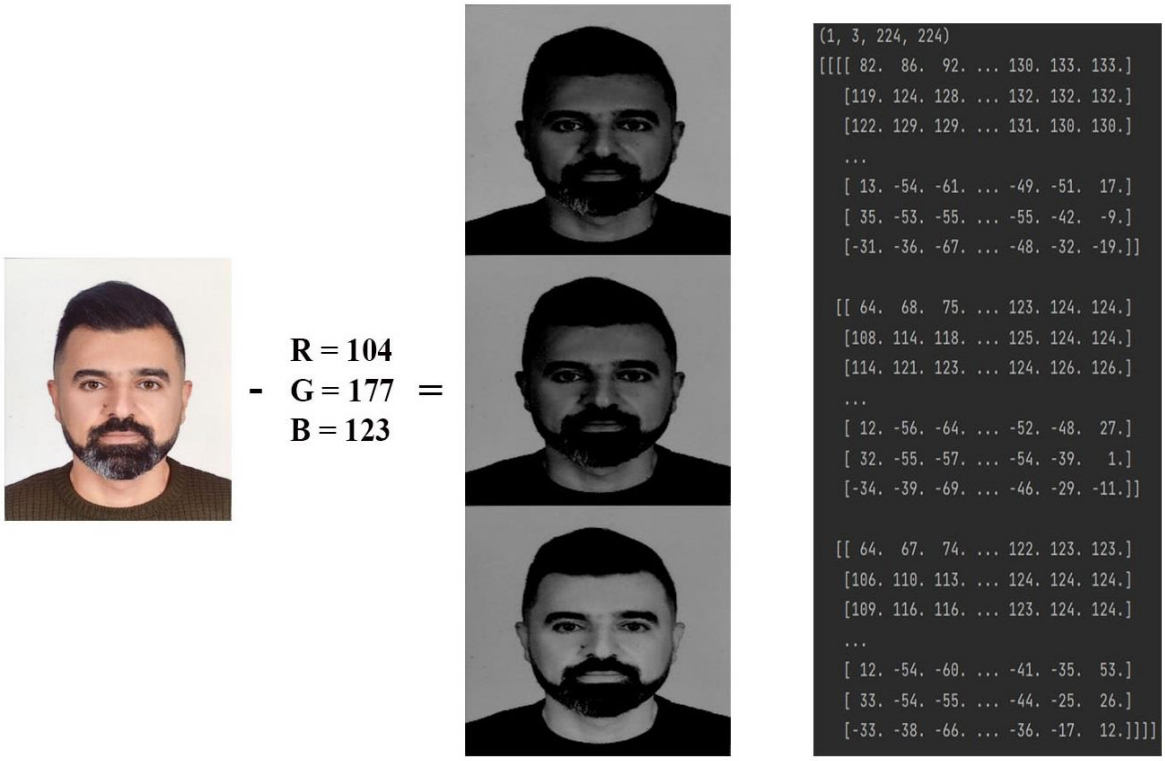

Figure 3-14. Mean subtraction: the original image on the left and the output image on the right.

It is thus necessary for us to determine the average pixel density of each color channel in the streaming video frames collected from the camera at the start of our research. We'll have three sets of numbers, each representing the average of the three different color channels. First, we'll describe the input picture, which is a collection of frames from the video stream, and then we'll provide the default scale for our images, which is "1.0," although we may offer a different number if necessary. To accommodate the convolutional neural network, we will modify the picture size to 224 by 224 pixels. 224 x 224, 227 x 227, or 299 x 299 are the most common dimensions for current neural networks. To determine the average subtraction, we'll utilize the numbers from channels 104, 177, and 123, which are often used. By default, the option swapRB will be true since the OpenCV library expects that images are stored in BGR order and that we wish to show them in RGB.

3.4.1.2 OpenCV-DNN-readNet

Face detection will be achieved by using a pre-trained model based on SSD and the "ResNet-10" architecture. The file type is caffemodel. It can be found on Github.

OpenCV-DNN-readNet, a function of the OpenCV library, allows us to load pre-trained models from specific frameworks. We are also not required to provide a framework argument. So this function was used to load the face detector model:

- Caffemodel [94]
- Prototxt [95]

Convolutional architecture for rapid feature embedding (Caffe) was developed by Berkeley AI Research (BAIR) and community members [96,97], to train deep learning models. Users can create classification and segmentation models for images using it as an alternative to conventional object-finding procedures that are more resilient and efficient. Users first generate and store their models as PROTOTXT text files. When Caffe is used to train and improve a model, the application saves the model as a CAFFEMODEL file [98].

In our project, we used a Caffemodel file. It is an SSD-based face detector and the ResNet-10. We will enforce two variables, "prototxtPath" and "weightsPath", which will contain the path of the face detector model to load from the hard disk. Using OpenCV-DNN-readNet, which allows us to import pre-trained models and frameworks, we put the previous two variables inside the "faceNet" parameter mentioned above, as shown in the command below:

```
prototxtPath = r"face_detector\deploy.prototxt"
weightsPath = r"face_detector\res10_300x300_ssd_iter_140000.caffemodel"
faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)
```

Then we pass the blob we got from the above function (3.4.1.1 section) through the parameter "faceNet" to make predictions of facial detections on our video streams through the command below:

```
faceNet.setInput(blob)
detections = faceNet.forward()
```

3.4.2 Load Face Mask Model

Using the load_model function in the Keras library, the trained face mask detector model is preloaded and put into the "maskNet" parameter, as shown in the command below, to be used later to predict face masks.

```
maskNet = load_model("mask_detection.model")
```

3.4.3 Extract Each Face ROI

After preparing the video frames through the blob function and passing them through the pre-trained model (SSD) to detect the face in the image, we will design three lists: one for the face, the second for the location of the face, and the third for the predictions of face mask detection, and the names of these lists will be (faces, locs, preds) that we will need in our work later. We make a loop to extract the degree of confidence to detect the face in the image, and inside this loop, we filter the weak detections and extract the surrounding boxes for faces. In addition to making sure that the coordinates of these boxes do not fall outside the image's borders, we will also symbolize the degree of confidence in the variable "confidence". As shown below:

```
for i in range(0, detections.shape[2]):
    confidence = detections[0, 0, i, 2]

    if confidence > 0.5:
        box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
        (startX, startY, endX, endY) = box.astype("int")

        (startX, startY) = (max(0, startX), max(0, startY))
        (endX, endY) = (min(w - 1, endX), min(h - 1, endY))
```

The bounding boxes extracted above are the boundaries of the regions of interest (ROI) containing the eyes, eyebrows, nose, and mouth, which are encoded by the variables (startX, startY, endX, and endY). We will crop this region, switch the color channels from BGR to RGB, resize it to dimensions (224,224), convert it to a Numpy matrix, and pass it through our previously mentioned trained grid to make predictions for face mask detection. The variable 'face' is enforced for all previous actions as shown below:

```
face = frame[startY:endY, startX:endX]
face = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)
face = cv2.resize(face, (224, 224))
face = img_to_array(face)
face = preprocess_input(face)
```

After that, we will add the variable "face" and the surrounding boxes (startX, startY, endX, endY) to the lists (faces, locs) that we have created, as shown below:

```
faces.append(face)
locs.append((startX, startY, endX, endY))
```

Figure (3-15) depicts how (ROI) will be calculated if video streams are used.

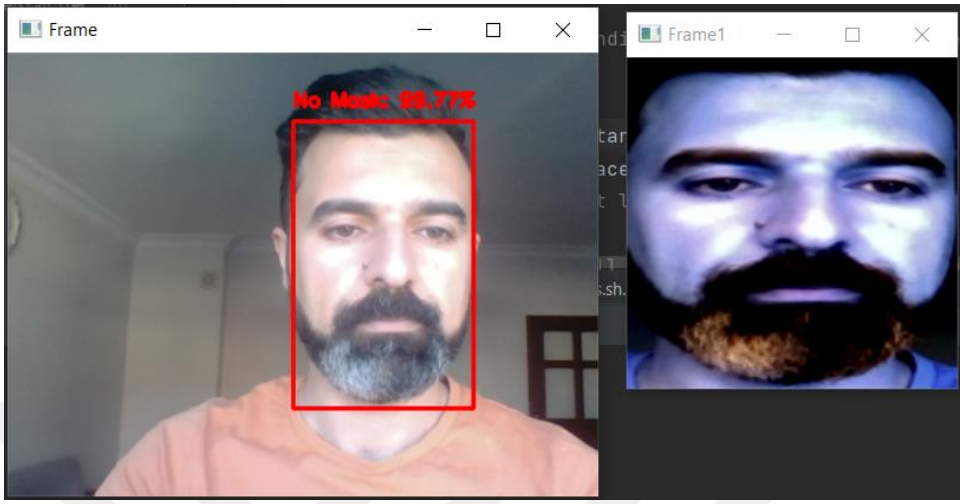

Figure 3-15. ROI for video streams

3.4.4 Apply Face Mask Detector To Each ROI

After we crop the regions of interest (ROI) for each face and add it to the list (faces), we'll convert this list into a NumPy array for faster arithmetic. Then we initialize the third list, which we called "preds", and we pass the NumPy array, which contains (ROI) for each face through parameter "maskNet", which includes our trained model for making face mask predictions as shown below:

```
faces = np.array(faces, dtype="float32")
preds = maskNet.predict(faces, batch_size=32)
```

3.4.4.1 Passing Video-Streams through Our Trained Model

We have preprocessed the video frames, configured lists and parameters, and uploaded the models, the face detector, and the face mask detector. We will start playing the video streams from the webcam of our laptop using the imutils library and put these streams inside the "frame" parameter that we configured earlier. Then we summon the function (detect_and_predict) that we set up earlier to perform all the processing actions mentioned in the previous steps. And finally, we pass it through our model to get predictions of the faces that wear masks or not.

4 RESULT AND DISCUSSION

It was tested on an Acer laptop with an Intel Core i7-8565U 1.8 processor, NVIDIA GeForce MX130 graphics card with 2 GB VRAM, 8 GB DDR4 RAM, PyCharm 2021.3.3, and Python 3.9.

4.1 Accuracy and Loss Calculation For Model Training

When we train deep learning networks, we need to keep track of their development. We can look at several metrics during training to see if the network is getting more accurate and how quickly it is getting better.

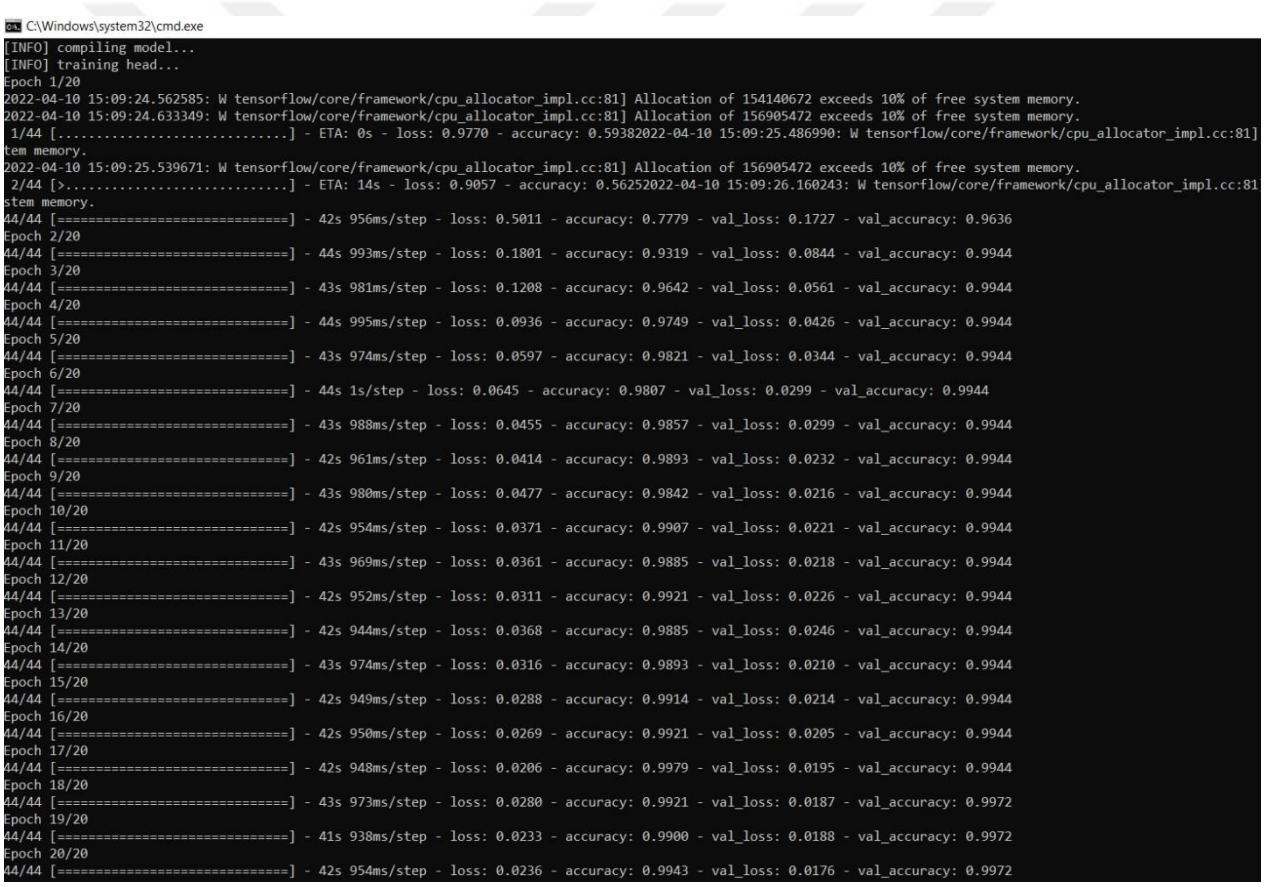

Figure 4-1. History of training

The training loss is the average loss per training dataset and the distance between the base facts and predictions. The neural network adjusts its weights and biases to minimize the loss as the model develops over time. As a result, the loos in the early batches of an epoch are often greater than the loss in the later collections. For a period, the test loss is computed using the model at the end of that epoch, which results in a reduced loss. On the other hand,

val_loss is used to test data, while loss is used on training data alone. So the val_loss is a good indicator of the model's performance in handling the data.

Classification accuracy is merely a measure of how many instances were adequately categorized. Val acc is a better indicator of the model's performance since a well-trained neural network is more likely to match the training data. An accuracy of 99.72% was obtained in our project, as shown in the image above (4-1).

One of the indicators used to evaluate the success of a classification-based machine learning model shows 100% precision. We can get a better idea of our trained model's overall performance by looking at the precision, accuracy, recall, F1 score, and support shown in table (1) by Scikit-Learn [99] is an open-source Python machine learning toolkit that includes a variety of classification, regression, and aggregation methods for the Python programming language.

Table 4.1. sklearn metrics classification_report

	Precision	**Recall**	**F1 – score**	**Support**
Mask	1.00	1.00	1.00	254
No mask	1.00	0.99	1.00	103
Accuracy			1.00	357
Macro avg	1.00	1.00	1.00	357
Weighted	1.00	1.00	1.00	357

The metrics selected for evaluation of network is:

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

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

$$F1 - score = \frac{2}{recall^{-1} + precision^{-1}} = 2 * \frac{(precision * recall)}{(precision + recall)}$$

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

TP = True positive

TN = True negative

FP = False positive

FN = False negative

The 'true positive' values correspond to the photos rated as accurate and delivered a true result after the prediction. The 'true negative' values are the images that are 'true' but after the forecast delivered a false negative, and the 'false positive' values are the images incorrectly rated as false. "False-negative" images have been labeled "false" and, as a result, gave a false negative result.

The harmonic mean of precision and recall are calculated using the f1-score. Using the scores for each class, the classifier's ability to correctly categorize the data points in that class will be shown. When determining a model's support, look at the total number of samples in that category [100].

Figure 4-2. Training Loss and Accuracy

4.2 Comparison of models

Our trained model was compared with other models that were used for image classification, InceptionV3[101], NASNetMobile and DenseNet121 [102], Resnet50 [103]. Table (2) compares the accuracy of our model with other models. In Table (3), the F1 degree of our

model was analyzed with the rest of the other models. InceptionV3 is a convolutional neural network used for image classification that was built to reduce the computational effort of neural networks while maintaining network efficiency. In contrast, this network focuses on computational cost, Resnet50 focuses on computational accuracy, and a deep neural network consists of a depth of 50 layers. Despite its high accuracy in classifying the image, it is computationally heavy, so it is difficult to use it in real-time. The same applies to the InceptionV3 network; the deeper the network, the more computational. NASNetMobile and DenseNet121 are deep neural networks trained to classify many images. Despite their classification accuracy, mobilenetv2 remains the fastest and easiest network to use with mobile and embedded devices and is more convenient than other networks in real-time.

Table 4.2. Comparison of accuracy

Architectures Used	**Year**	**Accuracy (%)**
InceptionV3	2020	99.92
NASNetMobile	2021	99.45
Dense Net121	2021	98.73
Resnet50	2021	99.64
Mobilenetv2	**2022**	**99.72**

Table 4.3. Comparison of F1-Score

Architectures Used	**Year**	**F1 score**
InceptionV3	2020	99.9
NASNetMobile	2021	99.13
Dense Net121	2021	99.40
Resnet50	2021	98
Mobilenetv2	**2022**	**100**

4.3 Detecting Model In Real-Time

Tkinter, a free Python GUI package, created the user interface. There are two buttons, one to start the program and the other to leave it, and a label that shows the time and another that displays the date, as shown in Figures (4-3).

Figure 4-3. Application interface

Figure (4-4) depicts the predictions made on video streams from the webcam, which predict the presence of the mask, where the bounding box appears in green, the degree of accuracy, and the 'mask' label.

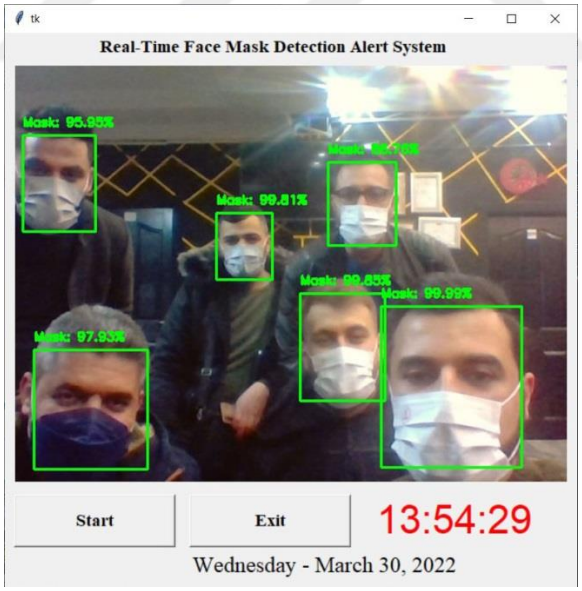

Figure 4-4. wearing a mask

Figure (4-5) depicts the predictions made on video streams from the webcam, which predicts the absence of a catcher, where the bounding box appears in red, the degree of accuracy, and a 'no mask' label, in addition to an alarm (a warning sound).

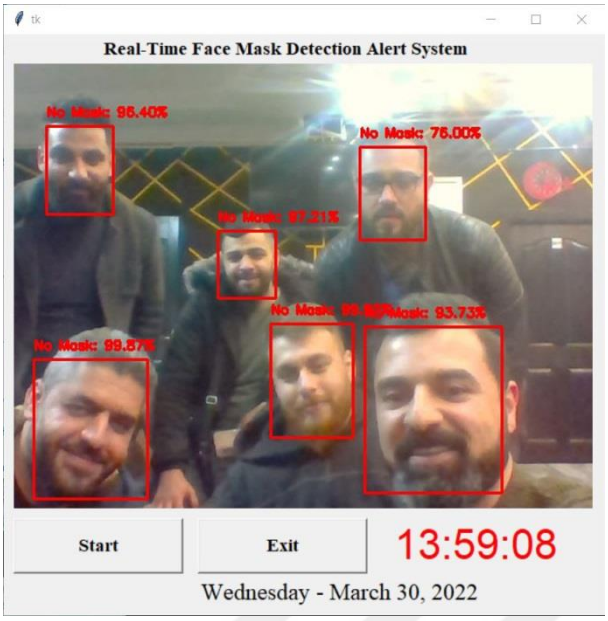

Figure 4-5. not wearing a mask

Figure (4-6) shows the predictions made on the video streams from the webcam, which predict the two cases: wearing a mask and not.

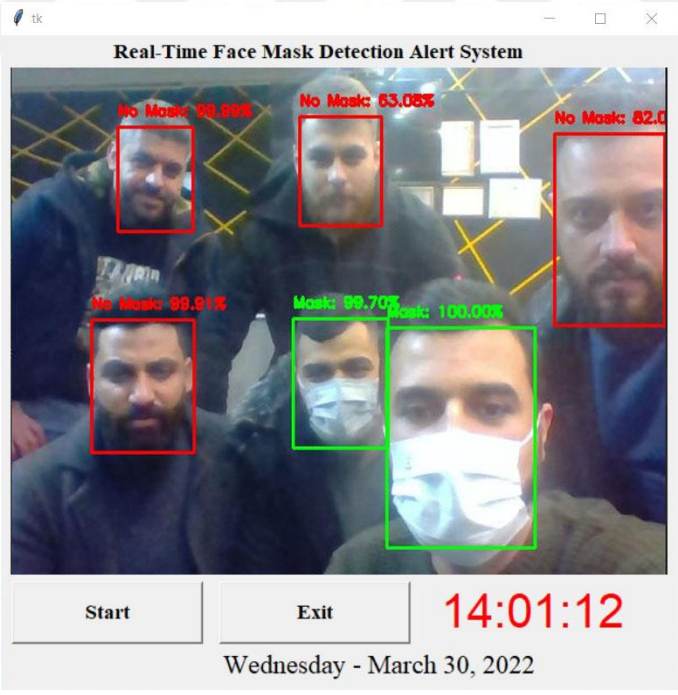

Figure 4-6. wearing a mask and no wearing a mask

5 CONCLUSION

In light of the outbreak of the Coronavirus, which caused the suffering of governments around the world to control this disease and limit its spread, According to the World Health Organization, protection from COVID-19 infection is critical. In light of the preventive steps taken to limit the spread of the Coronavirus, wearing a face mask has become one of the essential preventative measures that prevent the spread of the disease among people, as it is spread through the air through sneezing and coughing. So we presented an application that uses deep machine learning to create a real-time face mask detection warning system to identify those who are not wearing face masks. We used a pre-trained neural network to classify the images. We cleaned the dataset manually by removing duplicate and blurred images to increase the model's accuracy; In addition to preprocessing our data, the data set was loaded from the folder as input to the network. Our data have been resized to 224 x 224. They are separated into two categories (mask and without mask). The images are then converted into a NumPy matrix for quick calculations and lists. Next, we were able to improve the model's accuracy by increasing the data size in the way we discussed previously. Our images were accurately graded using the MobilenetV2 classifier, with an accuracy of 99.72% and an f1-score of 100%. In this project, OpenCV library deep neural networks were used along with TensorFlow and Keras libraries and pre-trained face detection (SSD) model. Using the OpenCV library, we load a face detection (SSD) model to perform face detections on video streams, and after achieving face detection, we pass it through our model to predict a face mask.

We found that our model worked well when applied to video streams. We did this in two steps: first, we found faces in the video streams. Then, we used our model on those faces to figure out if they were wearing masks or not.

Since the mask obscures part of the face, it is impossible to predict the face mask detector if a large part of the face is covered. We suppose use object detection (object detector) with a particular class (mask, without mask). In that case, we think we can get rid of this problem because it is more accurate and also faster because we do not need to identify the face first before applying the face mask detector model.

5.1 Suggestions

- Although there are actual images of people wearing masks in our data set, they are few, as it also includes images of people wearing face masks, as described in Section 3.2.1. This means that the face mask detection model we trained can be improved by collecting more accurate data. It is also possible to include images that baffle the model, such as a person without any mask but with a handkerchief on their face, their shirt on their face, etc.
- Work to solve the face detection problem if the most significant part of the face is covered. This is because our model works on detected faces, and as we mentioned earlier, the video streams pass through a face detector, and if a face is found, it will be passed through our model to predict the face mask. If the more significant part of the face is covered, then no face will be detected in the first place, and as a result, our model cannot predict the face mask.

REFERENCES

[1]. Mitchell, T. M. (1997). Does machine learning really work? AI Magazine, 18(3), 11.

[2]. Koza, J. R., Bennett, F. H., Andre, D., & Keane, M. A. (1996). Automated Design of Both the Topology and Sizing of Analog Electrical Circuits Using Genetic Programming. In Artificial Intelligence in Design '96 (pp. 151–170). Springer Netherlands. https://doi.org/10.1007/978-94-009-0279-4_9

[3]. Hu, J., Niu, H., Carrasco, J., Lennox, B., & Arvin, F. (2020). Voronoi-Based Multi-Robot Autonomous Exploration in Unknown Environments via Deep Reinforcement Learning. IEEE Transactions on Vehicular Technology, 69(12), 14413–14423. https://doi.org/10.1109/TVT.2020.3034800

[4]. Huang, T. S. (1997). Computer Vision: Evolution and Promise. In Report.

[5]. Forsyth, D., & Ponce, J. (2011). Computer vision: A modern approach. Prentice hall.

[6]. Hjelmås, E., & Low, B. K. (2001). Face Detection: A Survey. Computer Vision and Image Understanding, 83(3), 236–274. https://doi.org/10.1006/cviu.2001.0921

[7]. Malviya, R., Kumar, D., & Sharma, P. K. (2020). Corona Virus A Review of COVID19-51418.pdf (p. 18). https://doi.org/10.14744/ejmo.2020.51418

[8]. Xing-Yi Gea, Ying Pub, c, Ce-Heng Liaoa, Wen-Fen Huangd, Qi Zenge, Hui Zhouf, Bin Yif, Ai-Min Wangg, Qing-Ya Doub, Peng-Cheng Zhoub, Hui-Ling Chenc, Hui-Xia Liuc, Dao-Miao Xuh, Xiang Cheni, X. H. (2020). Evaluation of the exposure risk of SARS-CoV-2 in different hospital.pdf (p. 7). https://doi.org/10.1016/j.scs.2020.102413

[9]. CDC. (2020). Guidance for Wearing Masks. Centers for Disease Control and Prevention (CDC). https://www.cdc.gov/coronavirus/2019-ncov/prevent-getting-sick/cloth-face-cover-guidance.html

[10]. Nagrath, P., Jain, R., Madan, A., Arora, R., Kataria, P., & Hemanth, J. (2021). SSDMNV2: A real time DNN-based face mask detection system using single shot multibox detector and MobileNetV2. Sustainable Cities and Society, 66, 11. https://doi.org/10.1016/j.scs.2020.102692

[11]. Li, Y., Sun, B., Wu, T., & Wang, Y. (2016). Face detection with end-to-end integration of a convNet and a 3D model. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 9907 LNCS, 420–436. https://doi.org/10.1007/978-3-319-46487-9_26

[12]. Viola, P., & Jones, M. (2001). Managing work role performance: Challenges for twenty-first century organizations and their employees. In Rapid Object Detection using a Boosted Cascade of Simple Features (pp. 511–518).

[13]. Li, H., Hua, G., Lin, Z., Brandt, J., & Yang, J. (2013). Probabilistic Elastic Part Model for Unsupervised Face Detector Adaptation.

[14]. Xiangxin Zhu, & Ramanan, D. (2012). Face detection, pose estimation, and landmark localization in the wild. 2012 IEEE Conference on Computer Vision and Pattern Recognition, 2879–2886. https://doi.org/10.1109/CVPR.2012.6248014

[15]. Lee, D.-H., Chen, K.-L., Liou, K.-H., Liu, C.-L., & Liu, J.-L. (2021). Deep learning and control algorithms of direct perception for autonomous driving. Applied Intelligence, 51(1), 237–247. https://doi.org/10.1007/s10489-020-01827-9

[16]. Zhao, Z. Q., Zheng, P., Xu, S. T., & Wu, X. (2019). Object Detection with Deep Learning: A Review. In IEEE Transactions on Neural Networks and Learning Systems (Vol. 30, Issue 11, pp. 3212–3232). https://doi.org/10.1109/TNNLS.2018.2876865

[17]. Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X., & Pietikäinen, M. (2020). Deep Learning for Generic Object Detection: A Survey. In International Journal of Computer Vision (Vol. 128, Issue 2, pp. 261–318). https://doi.org/10.1007/s11263-019-01247-4

[18]. Liu, W. (1999). SSD: Single Shot MultiBox Detector. BMJ (Clinical Research Ed.), 319(7209), A. http://www.ncbi.nlm.nih.gov/pubmed/10463930

[19]. Wang, Y., & Zheng, J. (2018). Real-time face detection based on YOLO. 1st IEEE International Conference on Knowledge Innovation and Invention, ICKII 2018, 221–224. https://doi.org/10.1109/ICKII.2018.8569109

[20]. Girshick, R., Donahue, J., Darrell, T., Malik, J., Berkeley, U. C., & Malik, J. (2014). 1043.0690. In Proceedings of the IEEE Computer Society Conference on Computer

Vision and Pattern Recognition (Vol. 1, p. 5000). https://doi.org/10.1109/CVPR.2014.81

[21]. Jiang, M., Fan, X., & Yan, H. (2020). RetinaMask: A Face Mask detector. http://arxiv.org/abs/2005.03950

[22]. Norman, J. (2019). Woodrow Bledsoe Originates of Automated Facial Recognition : History of Information. History of Information. https://www.historyofinformation.com/detail.php?id=2126

[23]. Nilsson, N. J. (2009). The quest for artificial intelligence. Cambridge University Press.

[24]. de Leeuw, K. M. M., & Bergstra, J. (2007). The history of information security: a comprehensive handbook. Elsevier.

[25]. Gates, K. (2011). Our Biometric Future: Facial Recognition Technology and the Culture of Surveillance, New York u. London.

[26]. Kundu, M. K., Mitra, S., Mazumdar, D., & Pal, S. K. (2012). Perception and Machine Intelligence: First Indo-Japan Conference, PerMIn 2012, Kolkata, India, January 12-13, 2011, Proceedings (Vol. 7143). Springer Science & Business Media.

[27]. Fallis, A. . (2013). Eigenface - Wikipedia. Journal of Chemical Information and Modeling. https://en.wikipedia.org/wiki/Eigenface

[28]. Wechsler, H. (2009). Reliable face recognition methods: system design, implementation and evaluation (Vol. 7). Springer Science & Business Media.

[29]. Wang, J., Chan, L., & Wang, D. (2006). Neural Information Processing: 13th International Conference, ICONIP 2006, Hong Kong, China, October 3-6, 2006, Proceedings, Part III (Vol. 4234). Springer.

[30]. Stat, T. (2003). Face Recognition Using Laplacianfaces (Vol. 4, Issue 2, pp. 2–5). https://doi.org/10.1109/TPAMI.2005.55 ·

[31]. University Of Southern California. (1997). “Mugspot” Can Find A Face In The Crowd -- Face-Recognition Software Prepares To Go To Work In The Streets. ScienceDaily. https://www.sciencedaily.com/releases/1997/11/971112070100.htm

[32]. Li, S. Z. (2005). Anil K. Jain-Handbook of Face Recognition-Springer Science & Business Media, 15.

[33]. Datta, A. K., Datta, M., & Banerjee, P. K. (2015). Face detection and recognition: theory and practice. CRC Press.

[34]. Egorov, A. D., Shtanko, A. N., & Minin, P. E. (2015). Selection of Viola–Jones algorithm parameters for specific conditions. In Bulletin of the Lebedev Physics Institute (Vol. 42, Issue 8, pp. 244–248). https://doi.org/10.3103/S1068335615080060

[35]. Brownlee, J. (2019). A Gentle Introduction to the ImageNet Challenge (ILSVRC). Deep Learning for Computer Vision. https://machinelearningmastery.com/introduction-to-the-imagenet-large-scale-visual-recognition-challenge-ilsvrc/

[36]. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., & Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision, 115(3), 211–252. https://doi.org/10.1007/s11263-015-0816-y

[37]. Yang, M.-H., Kriegman, D. J., & Ahuja, N. (2002). Detecting faces in images: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(1), 34–58.

[38]. Rizvi, D. Q. M. (n.d.). A Review on Face Detection Methods.pdf.

[39]. Doğrukartal, E. (2010). Neural network based face detection.

[40]. Fissore, L., Kaltenmeier, A., Laface, P., Micca, G., & Pieraccini, R. (1990). The Recognition Algorithms. In Advanced Algorithms and Architectures for Speech Understanding (pp. 7–78). https://doi.org/10.1007/978-3-642-84341-9_2

[41]. Divyansh, D. (2018). Face Detection For Beginners. In the past few years, face recognition… | by Divyansh Dwivedi | Towards Data Science. https://towardsdatascience.com/face-detection-for-beginners-e58e8f21aad9

[42]. Kohonen, T. (2012). Self-organization and associative memory (Vol. 8). Springer Science & Business Media.

[43]. Kirby, M., & Sirovich, L. (1990). Application of the Karhunen-Loeve procedure for the characterization of human faces. IEEE Transactions on Pattern Analysis and Machine Intelligence, 12(1), 103–108.

[44]. Turk, M., & Pentland, A. (1991). Eigenfaces for Recognition: Journal of Cognitive

Neurosicence.

[45]. Sung, K.-K. (1996). Learning and example selection for object and pattern detection.

[46]. Sung, K.-K., & Poggio, T. (1998). Example-based learning for view-based human face detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, 20(1), 39–51.

[47]. Propp, M., & Samal, A. (1992). Artificial neural network architectures for human face detection. Proceedings of the 1992 Artificial Neural Networks in Engineering, ANNIE'92, 535–540.

[48]. FOGELMAN SOULIE, F., Viennet, E., & Lamy, B. (1993). Multi-modular neural network architectures: applications in optical character and human face recognition. International Journal of Pattern Recognition and Artificial Intelligence, 7(4), 721–755.

[49]. Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., & Lang, K. J. (1989). Phoneme recognition using time-delay neural networks. IEEE Transactions on Acoustics, Speech, and Signal Processing, 37(3), 328–339.

[50]. Vaillant, R., Monrocq, C., & Le Cun, Y. (1994). Original approach for the localisation of objects in images. IEE Proceedings-Vision, Image and Signal Processing, 141(4), 245–250.

[51]. Burel, G., & Carel, D. (1994). Detection and localization of faces on digital images. Pattern Recognition Letters, 15(10), 963–967.

[52]. Feraud, R. (1998). PCA, neural networks and estimation for face detection. In Face Recognition (pp. 424–432). Springer.

[53]. Feraud, R., & Bernier, O. (1997). Ensemble and Modular Approaches for Face Detection: A Comparison. NIPS, 472–478.

[54]. Feraund, R., Bernier, O. J., Viallet, J.-E., & Collobert, M. (2001). A fast and accurate face detector based on neural networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 23(1), 42–53.

[55]. Lin, S.-H., Kung, S.-Y., & Lin, L.-J. (1997). Face recognition/detection by probabilistic decision-based neural network. IEEE Transactions on Neural Networks, 8(1), 114–132.

[56]. Rowley, H. A., Baluja, S., & Kanade, T. (1995). Human face detection in visual scenes. Citeseer.

[57]. ROWLEY, G. A. (1996). Neural network-based afce detection. Proc. CVOR, 203–208.

[58]. Rowley, H. A., Baluja, S., & Kanade, T. (1998). Neural Network-Based Face Detection.pdf. IEEE Transactions on Pattern Analysis and Machine Intelligence. https://doi.org/10.1109/34.655647

[59]. Osuna, E., Freund, R., & Girosit, F. (1997). Training support vector machines: an application to face detection. Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 130–136.

[60]. Roth, D., Cumby, C., Carlson, A., & Rosen, J. (1999). The SNoW learning architecture. Technical Report UIUCDCS–R–99–2102, UIUC Computer Science Department., 8–10.

[61]. Littlestone, N. (1988). Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. Machine Learning, 2(4), 285–318.

[62]. Schneiderman, H., & Kanade, T. (1998). Probabilistic modeling of local appearance and spatial relationships for object recognition. Proceedings. 1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No. 98CB36231), 45–51.

[63]. Rabiner, L. (1993). Fundamentals of speech recognition. Fundamentals of Speech Recognition.

[64]. Qian, R. J., & Huang, T. S. (1997). Object detection using hierarchical MRF and MAP estimation. Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 186–192.

[65]. Quinlan, J. R. (2014). C4. 5: programs for machine learning. Elsevier.

[66]. Huang, J., Gutta, S., & Wechsler, H. (1996). Detection of human faces using decision trees. Proceedings of the Second International Conference on Automatic Face and Gesture Recognition, 248–252.

[67]. Deng, L., & Yu, D. (2014). Deep learning: methods and applications. Foundations and Trends in Signal Processing, 7(3–4), 197–387.

[68]. Schmidhuber, J. (2015). Deep learning in neural networks: An overview. Neural

Networks, 61, 85–117.

[69]. Glauner, P. O. (2015). Deep convolutional neural networks for smile recognition. ArXiv Preprint ArXiv:1508.06535.

[70]. Song, H. A., & Lee, S.-Y. (2013). Hierarchical representation using NMF. International Conference on Neural Information Processing, 466–473.

[71]. Ciregan, D., Meier, U., & Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. 2012 IEEE Conference on Computer Vision and Pattern Recognition, 3642–3649.

[72]. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems, 25.

[73]. Russell, J. (2017). Google's AlphaGo AI wins three-match series against the world's best Go player. TechCrunch. https://techcrunch.com/2017/05/24/alphago-beats-planets-best-human-go-player-ke-jie/amp/

[74]. Pulli, K., Baksheev, A., Kornyakov, K., & Eruhimov, V. (2012). Real-time computer vision with OpenCV. Communications of the ACM, 55(6), 61–69.

[75]. CUDA | OpenCV. (2016). Web. https://opencv.org/platforms/cuda/

[76]. garage: OpenCV for Maemo: Project Info. (n.d.). Retrieved February 24, 2022, from https://garage.maemo.org/projects/opencv

[77]. GitHub - blackberry/OpenCV: Open Computing Vision. (n.d.). Retrieved February 24, 2022, from https://github.com/blackberry/OpenCV

[78]. Fu, C.-Y., Liu, W., Ranga, A., Tyagi, A., & Berg, A. C. (2017). DSSD : Deconvolutional Single Shot Detector. http://arxiv.org/abs/1701.06659

[79]. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 4510–4520.

[80]. Anisimov, D., & Khanova, T. (2017). Towards lightweight convolutional neural networks for object detection. 2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), 1–8.

[81]. GitHub - prajnasb/datas. (n.d.). Retrieved December 10, 2021, from https://github.com/prajnasb/datas

[82]. Larxel. (2020). Face Mask Detection | Kaggle. In Www.Kaggle.Com. https://www.kaggle.com/andrewmvd/face-mask-detection

[83]. Shorten, C., & Khoshgoftaar, T. M. (n.d.). A survey on image data augmentation for deep learning. J. Big Data 6 (1), 1–48 (2019).

[84]. ImageNet. (n.d.). Retrieved January 3, 2022, from https://image-net.org/update-mar-11-2021.php

[85]. Why Google's MobileNet-V2 is a revolutionary next gen on-device computer vision network. (n.d.). Retrieved April 10, 2022, from https://analyticsindiamag.com/why-googles-mobilenetv2-is-a-revolutionary-next-gen-on-device-computer-vision-network/

[86]. Singh, S. A., Meitei, T. G., & Majumder, S. (2020). 6 - Short PCG classification based on deep learning (B. Agarwal, V. E. Balas, L. C. Jain, R. C. Poonia, & B. T.-D. L. T. for B. and H. I. Manisha (Eds.); pp. 141–164). Academic Press. https://doi.org/https://doi.org/10.1016/B978-0-12-819061-6.00006-9

[87]. Quan, C., Hua, L., Sun, X., & Bai, W. (2016). Multichannel convolutional neural network for biological relation extraction. BioMed Research International, 2016.

[88]. Springenberg, J. T., Dosovitskiy, A., Brox, T., & Riedmiller, M. (2015). Striving for simplicity: The all convolutional net. In 3rd International Conference on Learning Representations, ICLR 2015 - Workshop Track Proceedings.

[89]. Hahnloser, R. H. R., Sarpeshkar, R., Mahowald, M. A., Douglas, R. J., & Seung, H. S. (2000). Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. Nature, 405(6789), 947–951.

[90]. Zhang, X., Zou, Y., & Shi, W. (2017). Dilated convolution neural network with LeakyReLU for environmental sound classification. 2017 22nd International Conference on Digital Signal Processing (DSP), 1–5.

[91]. Gulcehre, C., Moczulski, M., Denil, M., & Bengio, Y. (2016). Noisy activation functions. International Conference on Machine Learning, 3059–3068.

[92]. Opschoor, J. A., Schwab, C., & Zech, J. (2019). Exponential ReLU DNN expression 703 of holomorphic maps in high dimension. Technical report, Zurich. 704.

[93]. [Deep Learning] Using OpenCV as deep learning inference engine with application to image classification - Tech(B)log of Dr. Shrishail Gajbhar. (n.d.). Retrieved March 23, 2022, from https://shrishailsgajbhar.github.io/post/Deep-Learning-Image-Classification-Opencv-DNN

[94]. computer_vision/res10_300x300_ssd_iter_140000.caffemodel at master · gopinath-balu/computer_vision · GitHub. (n.d.). Retrieved March 24, 2022, from https://github.com/gopinath-balu/computer_vision/blob/master/CAFFE_DNN/res10_300x300_ssd_iter_140000.caffemodel

[95]. opencv/samples/dnn/face_detector at 4.x · opencv/opencv · GitHub. (n.d.). Retrieved March 24, 2022, from https://github.com/opencv/opencv/tree/4.x/samples/dnn/face_detector

[96]. Keras deep learning framework. (2020). http://caffe.berkeleyvision.org/

[97]. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., & Darrell, T. (2014). Caffe: Convolutional Architecture for Fast Feature Embedding. http://arxiv.org/abs/1408.5093

[98]. CAFFEMODEL File Extension - What is a .caffemodel file and how do I open it? (2021). Fileinfo. https://fileinfo.com/extension/caffemodel

[99]. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., & Dubourg, V. (2011). Scikit-learn: Machine learning in Python. The Journal of Machine Learning Research, 12, 2825–2830.

[100]. sklearn.metrics.f1_score — scikit-learn 1.0.2 documentation. (n.d.). Retrieved March 27, 2022, from https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html

[101]. Jignesh Chowdary, G., Punn, N. S., Sonbhadra, S. K., & Agarwal, S. (2020). Face Mask Detection Using Transfer Learning of InceptionV3. In Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes

in Bioinformatics): Vol. 12581 LNCS (pp. 81–90). https://doi.org/10.1007/978-3-030-66665-1_6

[102]. Chavda, A., Dsouza, J., Badgujar, S., & Damani, A. (2021). Multi-Stage CNN Architecture for Face Mask Detection. 2021 6th International Conference for Convergence in Technology (I2CT), 1–8. https://doi.org/10.1109/I2CT51068.2021.9418207

[103]. Loey, M., Manogaran, G., Taha, M. H. N., & Khalifa, N. E. M. (2021). A hybrid deep transfer learning model with machine learning methods for face mask detection in the era of the COVID-19 pandemic | Elsevier Enhanced Reader. https://reader.elsevier.com/reader/sd/pii/S0263224120308289?token=AED6DEA43C78E36CBF13DD5231834B6D1041B09E1F19390DA70832A1C24C7C3DEE24E5A86611E07AD9D8B4CF9285C510&originRegion=eu-west-1&originCreation=20210812162720

CURRICULUM VITAE

Kişisel Bilgiler	
Adı Soyadı	Ali Abbas Jasim
Doğum Yeri	
Doğum Tarihi	
Uyruğu	❒ T.C. ☑ Diğer:

Eğitim Bilgileri	
Lisans	
Üniversite	AL- Imam al sadiq
Fakülte	Fen Fakültesi
Bölümü	Computer Science
Mezuniyet Yılı	2009

Yüksek Lisans	
Üniversite	Kırşehir Ahi Evran Üniversitesi
Enstitü Adı	Fen Bilimleri Enstitüsü
Anabilim Dalı	İleri Teknolojiler
Programı	İleri Teknolojiler (Tezli Yüksek Lisans)
Mezuniyet Tarihi	2022