Social Distancing & Face Mask Detector with Alarm

Submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering Degree in Computer Science and Engineering

By

SYED RIZWAN (39111008)

TAKKELLAPATI SAI GANESH (39111012)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI – 600119

NOVEMBER- 2022

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with “A” grade by NAAC
Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai - 600 119
www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of Syed Rizwan (39111008),Takkellapati Sai Ganesh (39111012) who carried out the project entitled “**Social Distancing & Face Mask Detector with Alarm”** under my supervision from October 2022 to March 2023.

Internal Guide

Ms. VELVIZHI R

Head of the Department
Dr.L.LAKSHMANAN,M.E., Ph.D.,

Submitted for Viva voce Examination held on

Internal Examiner **External Examiner**

DECLARATION

We, Syed Rizwan (39111008), Takkellapati Sai Ganesh (39111012) hereby declare that the Project Report entitled “**Social Distancing & Face Mask Detector with Alarm** done by us under the guidance of Ms.VELVIZHI R Dept. of Computer Science and Engineering, SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI, is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:	**1.**
	2.
PLACE:	**SIGNATURE OF CANDIDATES**

ACKNOWLEDGEMENT

We are pleased to acknowledge our sincere thanks to the Board of Management of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. We are grateful to them.

We convey our thanks to **Dr. T. Sasikala M.E., Ph.D., Dean, School of Computing and Dr. L. Lakshmanan, M.E., Ph.D., Heads of the Department of Computer Science and Engineering,** for providing us necessary support and details at the right time during the progressive reviews.

We would like to express my sincere and deep sense of gratitude to our Project Guide Ms. VELVIZHI R Associate professor, Department of Computer Science and Engineering, for her valuable guidance, suggestions, and constant encouragement for the successful completion of our project work.

We wish to express my thanks to all Teaching and Non-teaching staff members of the Department of Computer Science and Engineering who were helpful in many ways for the completion of the project.

Abstract

Over the recent past, the raging pandemic due to COVID-19 is making the headlines, bringing about a global crisis with an inevitable spread. The use of a face mask and maintaining physical distancing is a precautionary measure as suggested by the WHO. The individuals infected with COVID-19 suffer respiratory problems accompanied by shortness of breath. The surroundings of the concerned individuals can be contaminated by their droplets carrying the virus. It is mandatory to wear a mask and follow physical distancing, yet many citizens violate the regulations. In such scenarios, frequent checks for face masks in public places and imposing fines are common. As object detection has unfolded to be an approachable biometric process, it has been widely applied in surveillance, security, autonomous driving, etc. With the rapid development of deep learning models, object detectors are highly suitable to develop social distancing and face mask detectors to administer the crowd via CCTV and surveillance cameras. The paper surveys various deep learning networks to develop such detectors. In this survey, the existing object detection models used for surveillance and people detection are analyzed. The one-stage and two-stage detectors along with their applications and performance are outlined in a comprehensive manner. We used technologies such as Opencv ,Machine learning, Deep learning, Python, Computer vision, Convolution neural network(CNN),Tensorflow & Keras and Mobile Ne

TABLE OF CONTENTS

CHAPTER 1

1.1 Introduction

Covid-19 is an epidemic that is spreading across the nations like wildfire. Though the mortality rate of this viral disease is less, the rate of spreading and people getting infected is very high because of its nature. Covid-19 is becoming the talk of everyone be it in villages or cities, people are afraid of and are in a panic state because of this disease. If the disease is not controlled in the initial stages in countries like India, it will lead to a disastrous kind of situation which will lead to heavy loss of life. The earlier it's controlled, it'll be better for the whole of humanity and particularly for countries like India, which is more densely populated. The main goal of our work is to develop a deep learning model to distinguish between individuals who wear masks and to make sure social distance is being followed.

Social Distancing and wearing masks have become the 2 most popular terms in today's life to keep oneself safe from getting infected from the Novel Coronavirus. Researchers have concluded that these 2 practices are the warriors against Covid. Governments across the nation are rigorously making efforts to make these 2 practices in action. We as a team came up to a real time solution of detecting social distance between humans and detecting persons with no mask covering. This solution was needed as there is a lack of awareness among people about the major precautions which is Mask Distance. Neither only mask can save, nor only distancing can save you but the aggregation of these two can.

In our project we have proposed 2 major solutions

(i) Mask Covering

(ii) (ii) Ensuring safe (physical/social) distance between people which is in real

1.2 Problem Statement

The problem statement, here face mask detector, could be constructed by first performing face detection on the frames coming from the video feed and later giving the frames with faces as an input to the classifier, which hence furnishes the desired output, i.e., faces with or without mask. The lack of social distance and people not wearing masks is a major issue we need to solve. Monitoring the crowd manually is very inefficient. We need a better solution until everyone is vaccinated to save lives. Hence, we design a system that uses deep learning to monitor social distancing and classify people without facial masks or coverings.

1.3 Scope and Objectives

The scope of this project is limited to the public areas, which is typical of monitoring by closed-circuit television cameras, e.g., of a bridge or in a shopping mall. It is believed that the scene that is being monitored has to pedestrians, and other objects, such as cars and animals are nonexistent. Here it is, one should be crowds and or in pools of people. Other visual media, such as film, television, fall outside the scope of the directive. We will use the data obtained from fixed surveillance cameras in real-world environments. We will continue to build on the existing research in the area of visual surveillance.

To develop an approach to assist, decrease the spread of coronavirus by monitoring citizens that are:

1. Maintaining Social Distance
2. The wearing of masks at public places.

We used technologies such as Opencv ,Machine learning, Deep learning, Python, Computer vision, Convolution neural network(CNN),Tensorflow & Keras and Mobile Net

OpenCV

OpenCV (Open Source Computer Vision Library) is **an open source computer vision and machine learning software library**. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products.Computer visionComputer vision is **a field of artificial intelligence that trains computers to interpret and understand the visual**

world. Using digital images from cameras and videos and deep learning models, machines can accurately identify and classify objects — and then react to what they “see.”

Convolution neural network(CNN)

Convolution is **a mathematical operation that allows the merging of two sets of information**. In the case of CNN, convolution is applied to the input data to filter the information and produce a feature map. This filter is also called a kernel, or feature detector, and its dimensions

Tensorflow & Keras

TensorFlow is an open-sourced end-to-end platform, a library for multiple machine learning tasks, while Keras is a high-level neural network library that runs on top of TensorFlow. Both provide high-level APIs used for easily building and training models, but Keras is more user-friendly because it's built-in Python.

MobileNet

MobileNet is a CNN architecture that is much faster as well as a smaller model that makes use of a new kind of convolutional layer, known as Depthwise Separable convolution. Because of the small size of the model, these models are considered very useful to be implemented on mobile and embedded devices.

1.4 Motivation

The increase in Covid-19 cases has led to numerous deaths. The Coronavirus has been spreading at a high rate. Hence, we have got an approach by using deep learning techniques combined with data pre-processing tasks to attain effectiveness and potency in the detection of face masks and social distance. It acts as a warning system for people who are not wearing face masks and those who do not maintain social distance. Covid-19 is an epidemic that is spreading across the nations like wildfire.

Though the mortality rate of this viral disease is less, the rate of spreading and people getting infected is very high because of its nature. Covid-19 is becoming the talk of everyone be it in villages or cities, people are afraid of and are in a panic state because of this disease.

This will help to limit the spread of the virus, securing the safety, and security of the people.

CHAPTER 2

2.1 LITERATURE SURVEY

Nowadays, Convolutional Neural Networks (CNN) are used in many ways to facilitate humans and also prevent their lives from damages such as fire disasters facial feature analysis healthcare, and many others fields in addition, in several studies, a resource-constrained device has been used to interact in real-time for human lives facilitation In this work, our main focus is on face mask detection for COVID19 prevention using a CNN. Generally, in most related publications, the focus is on face recognition and construction when wearing face masks. By contrast, this research focuses on the detection of those individuals who are not wearing face masks in public places to help in reducing the transmission of COVID-19. Scientists and researchers have proved that wearing face masks in public places significantly reduces the spreading rate of COVID-19. Bosheng Qin et al. proposed a new method to detect the face mask
wearing condition. Their trained model can classify the face mask condition into incorrect, correct, and no face mask wearing conditions. Their trained model has achieved 98.70% accuracy. Chong Li et al. proposed a YOLOv3-based framework for face detection. They have verified their method by training on the WIDER FACE CelebA , and FDDB datasets. Their model has achieved 93.9% accuracy. In , Din et al. proposed
a GAN-based novel framework that can detect and segment the face mask from the face image and regenerate the face image using a GAN. Their trained model-generated images look like the actual images. Nieto-Rodríguez et al. proposed a framework to detect
the special face mask in the medical room. They have used real-time image processing for detecting face masks. The objective of their work is to minimize the false-positive rate. They have achieved a 95% recall and a 5% false-positive rate for the detection of the surgical mask. Muhammad et al. proposed a framework called MRGAN to segment the microphone from the face image and used the GAN to reconstruct the segmented holes into the face image. They have trained their model on their synthetic dataset. The trained model works better than the state-of-the-art methods.

Mingijie Jiang et al. proposed a face mask detector called Retina Face Mask. Their method used a novel object-removal algorithm to remove predictions with low confidence. They have achieved 1.5% and 2.3% higher precision and 5.9% and 11.0% higher recall than state-of-the-art results on face mask and face detection, respectively. On the other hand, they also explored the performance of the proposed method on light-weight neural

networks such as MobileNet. Shashi Yadav et al. proposed deep learning mixed with
a geometric-based technique to monitor social distancing and face masks in public areas. They have implemented their model on Raspberry pi4. Their model detects violations of social distancing and face mask wearing by receiving input from cameras. When a violation occurs in a specific area, the developed system sends alerts to the control room and notifies
the public by an alarm. The paper presented in used a transfer learning strategy for the automatic identification of persons who are not wearing a face mask. In this approach, the researchers used pre-trained InceptionV3 with fine-tuning. They used Simulated Masked Face Dataset (SMFD) for training and testing and used image augmentation methods
to increase the number of training samples for better results evaluation. In another work presented in the authors used a hybrid model such as deep
learning and conventional machine learning to identify a person's face mask wearing condition. Their model consists of two main components. The first one is designed to extract prominent features from facial images using Resnet50, and the second component is a different classifier used for classification such as an ensemble algorithm, Support Vector Machine (SVM), and decision trees. For experimental evaluation, three different datasets
were used, namely, Labeled Faces in the Wild (LFW) , SMFD and World Masked Face Dataset (RMFD) . The SVM classifier reached an accuracy of 99.64% during training.

Secondly, we developed a lightweight CNN model that can be easily deployed over resource-constrained (edge devices) to efficiently and accurately detect face masks in a real-time scenario

CHAPTER 3

3.1 Proposed Methodology

Considering all the studies presented in the previous chapter, we have implemented the system architecture. The accuracy of the model depends on the training and the images considering various situations. This chapter presents the approach for the classification problem using a deep learning algorithm and the system architecture with a detailed description of each stage. The main objective of this work was to construct a model which is able to detect whether or not a person is wearing a mask or not, and if people adhere to the social distance, using a picture of people in a public place. A framework was designed using deep learning techniques to train the model on a set of images that include people with and without masks. The images were analyzed, and it worked fine. An analysis has been made to ensure the accuracy of the model by varying the hyperparameters that give the best accuracy.

Based on the literature study done, deep learning models require the data to be available so that the system decides. Hence, the first step of this architecture is data acquisition. This step is required for the collection, processing, and segregation that is the case where the scenarios are based on the roles that are involved in the decision-making cycle of the transfer of the data to the processing unit in order to carry out a further categorization, in the real world, the data consists of noisy, missing values, and, at times, it is in an invalid format, which can then be used directly for the design of the models. Hence, data preprocessing is done to prepare raw data and make it suitable for the models. Classification may be a technique where we categorize data into a given number of classes. The main goal of a classification problem is to spot the category/class to which a replacement data will fall into.

Classification result is the outcome we get after using the classifiers. Evaluation and Analysis is done to check the performance of the algorithm and find the best algorithm that works for our model.

3.2 System Architecture

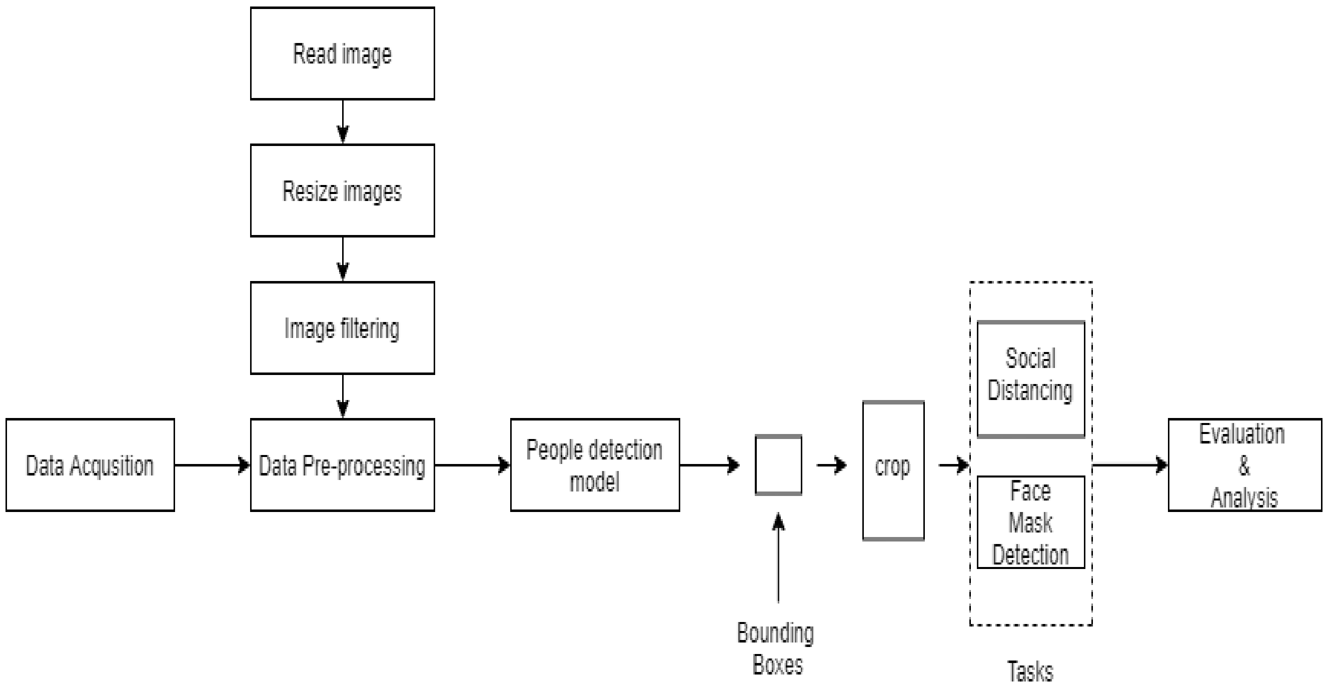

Fig no 3.2 System Architecture

3.3 Use Case Diagram

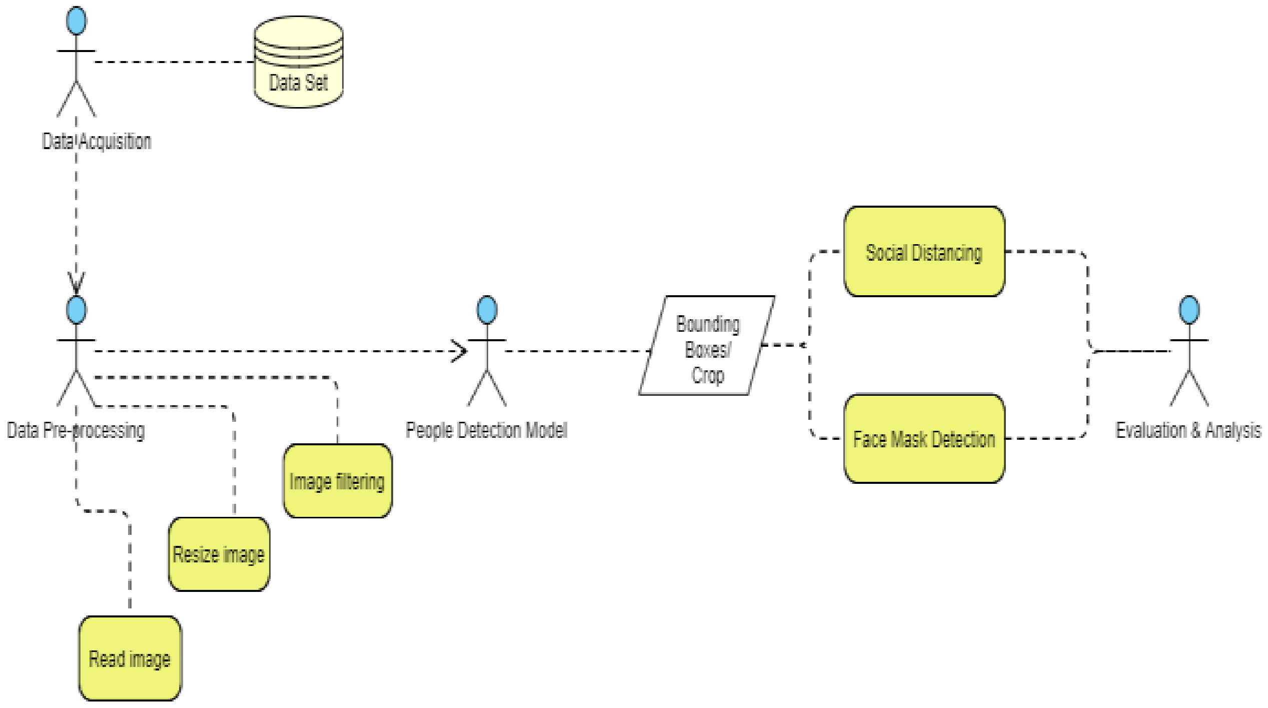

Fig no.3.3 Use Case Diagram

CHAPTER 4

4.1 Sample Code

Social_distance_detection.py:

```
from tensorflow import keras
from keras.applications.mobilenet_v2 import preprocess_input
from keras.utils import img_to_array
from keras.models import load_model
from imutils.video import VideoStream
import numpy as np
import argparse
import imutils
import time
import cv2
import os
import sys
import cv2
from math import pow, sqrt
import playsound
from threading import Thread

def detect_and_predict_mask(frame, faceNet, maskNet):
	# grab the dimensions of the frame and then construct a blob
	# from it
	(h, w) = frame.shape[:2]
	blob = cv2.dnn.blobFromImage(frame, 1.0, (300, 300),
		(104.0, 177.0, 123.0))

	# pass the blob through the network and obtain the face detections
	faceNet.setInput(blob)
	detections = faceNet.forward()

	# initialize our list of faces, their corresponding locations,
	# and the list of predictions from our face mask network
	faces = []
	locs = []
	preds = []

	# loop over the detections
	for i in range(0, detections.shape[2]):
		# extract the confidence (i.e., probability) associated with
		# the detection
		confidence = detections[0, 0, i, 2]

		# filter out weak detections by ensuring the confidence is
		# greater than the minimum confidence
		if confidence > args["confidence"]:
			# compute the (x, y)-coordinates of the bounding box for
```

```
                    # the object
                    box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
                    (startX, startY, endX, endY) = box.astype("int")

                    # ensure the bounding boxes fall within the dimensions of
                    # the frame
                    (startX, startY) = (max(0, startX), max(0, startY))
                    (endX, endY) = (min(w - 1, endX), min(h - 1, endY))

                    # extract the face ROI, convert it from BGR to RGB channel
                    # ordering, resize it to 224x224, and preprocess it
                    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)
                    face = np.expand_dims(face, axis=0)

                    # add the face and bounding boxes to their respective
                    # lists
                    faces.append(face)
                    locs.append((startX, startY, endX, endY))

        # only make a predictions if at least one face was detected
        if len(faces) > 0:
                # for faster inference we'll make batch predictions on all
                # faces at the same time rather than one-by-one predictions
                # in the above `for` loop
                preds = maskNet.predict(faces)

        # return a 2-tuple of the face locations and their corresponding
        # locations
        return (locs, preds)


# Parse the arguments from command line
arg = argparse.ArgumentParser(description='Social distance detection')

arg.add_argument("-a", "--alarm", type=str, default="", help="path alarm .WAV file")

arg.add_argument('-v', '--video', type = str, default = '', help = 'Video file path. If no path is
given, video is captured using device.')

arg.add_argument('-m', '--model', required = True, help = "Path to the pretrained model.")

arg.add_argument('-p', '--prototxt', required = True, help = 'Prototxt of the model.')

arg.add_argument('-l', '--labels', required = True, help = 'Labels of the dataset.')

arg.add_argument('-c', '--confidence', type = float, default = 0.2, help='Set confidence for
detecting objects')
```

```
arg.add_argument("-f", "--face", type=str, default="face_detector", help="path to face
detector model directory")

arg.add_argument("-m1", "--model1", type=str, default="model/mask_detector.model",
help="path to trained face mask detector model")

#arg.add_argument("-c", "--confidence", type=float, default=0.5,       help="minimum
probability to filter weak detections")

args = vars(arg.parse_args())

ALARM_ON = False

def sound_alarm(path):
        # play an alarm sound
        playsound.playsound(path)

labels = [line.strip() for line in open(args['labels'])]

# Generate random bounding box bounding_box_color for each label
bounding_box_color = np.random.uniform(0, 255, size=(len(labels), 3))

print("[INFO] loading face detector model...")

prototxtPath = os.path.sep.join([args["face"], "deploy.prototxt"])

weightsPath = os.path.sep.join([args["face"],
"res10_300x300_ssd_iter_140000.caffemodel"])

faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)

# load the face mask detector model from disk
print("[INFO] loading face mask detector model...")
maskNet = load_model(args["model1"])

# Load model
print("\nLoading model...\n")
network = cv2.dnn.readNetFromCaffe(args["prototxt"], args["model"])

print("\nStreaming video using device...\n")


# Capture video from file or through device
if args['video']:
   cap = cv2.VideoCapture(args['video'])
else:
   cap = cv2.VideoCapture(0)


frame_no = 0
```

```
while True:

    frame_no = frame_no+1

    # Capture one frame after another
    ret, frame = cap.read()

    #frame = imutils.resize(frame, width=1000)

           # detect faces in the frame and determine if they are wearing a

    if not ret:
        break

    (h, w) = frame.shape[:2]

    # Resize the frame to suite the model requirements. Resize the frame to 300X300
pixels
    blob = cv2.dnn.blobFromImage(cv2.resize(frame, (300, 300)), 0.007843, (300, 300),
127.5)

    network.setInput(blob)
    detections = network.forward()

    pos_dict = dict()
    coordinates = dict()

    # Focal length
    F = 615

    for i in range(detections.shape[2]):

        confidence = detections[0, 0, i, 2]

        if confidence > args["confidence"]:

            class_id = int(detections[0, 0, i, 1])

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

            # Filtering only persons detected in the frame. Class Id of 'person' is 15
            if class_id == 15.00:

                (locs, preds) = detect_and_predict_mask(frame, faceNet, maskNet)

                      # loop over the detected face locations and their corresponding
                # locations
                for (box, pred) in zip(locs, preds):
                                # unpack the bounding box and predictions
                (startX_mask, startY_mask, endX_mask, endY_mask) = box
                (mask, withoutMask) = pred

                        # determine the class label and color we'll use to draw
                        # the bounding box and text
                if mask > withoutMask:
                    label_mask = "Mask"
                    color = (0, 255, 0)
                    ALARM_ON = False
                else:
                    label_mask="No Mask"
                    color = (0, 0, 255)
                    if not ALARM_ON:
                        ALARM_ON = True
                        if args["alarm"] != "":
                            t = Thread(target=sound_alarm, args=(args["alarm"],))
                            t.deamon = True
                            t.start()

            # Draw bounding box for the object
            cv2.rectangle(frame, (startX, startY), (endX, endY),
bounding_box_color[class_id], 2)

            label = "{}: {:.2f}%".format(labels[class_id], confidence * 100)
            label_mask = "{}: {:.2f}%".format(label_mask, max(mask, withoutMask) * 100)
            print("{}".format(label))
            cv2.putText(frame, label, (startX_mask, startY_mask - 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
            cv2.rectangle(frame, (startX_mask, startY_mask), (endX_mask, endY_mask),
color, 2)


            coordinates[i] = (startX, startY, endX, endY)

            # Mid point of bounding box
            x_mid = round((startX+endX)/2,4)
            y_mid = round((startY+endY)/2,4)

            height = round(endY-startY,4)

            # Distance from camera based on triangle similarity
            distance = (165 * F)/height
            print("Distance(cm):{dist}\n".format(dist=distance))

            # Mid-point of bounding boxes (in cm) based on triangle similarity technique
            x_mid_cm = (x_mid * distance) / F
            y_mid_cm = (y_mid * distance) / F
            pos_dict[i] = (x_mid_cm,y_mid_cm,distance)

    # Distance between every object detected in a frame
    close_objects = set()
    for i in pos_dict.keys():
        for j in pos_dict.keys():
            if i < j:
            dist = sqrt(pow(pos_dict[i][0]-pos_dict[j][0],2) + pow(pos_dict[i][1]-
pos_dict[j][1],2) + pow(pos_dict[i][2]-pos_dict[j][2],2))

            # Check if distance less than 2 metres or 200 centimetres
            if dist < 200:
                close_objects.add(i)
                close_objects.add(j)

    for i in pos_dict.keys():
        if i in close_objects:
            COLOR = (0,0,255)
            if not ALARM_ON:
            ALARM_ON = True
            if args["alarm"] != "":
                t = Thread(target=sound_alarm, args=(args["alarm"],))
                t.deamon = True
                t.start()

        else:
            COLOR = (0,255,0)
            ALARM_ON = False
        (startX, startY, endX, endY) = coordinates[i]

        cv2.rectangle(frame, (startX, startY), (endX, endY), COLOR, 2)
        y = startY - 15 if startY - 15 > 15 else startY + 15
        # Convert cms to feet
        cv2.putText(frame, 'Depth: {i} ft'.format(i=round(pos_dict[i][2]/30.48,4)), (startX, y),
                cv2.FONT_HERSHEY_SIMPLEX, 0.5, COLOR, 2)

    cv2.namedWindow('Frame',cv2.WINDOW_NORMAL)

    # Show frame
    cv2.imshow('Frame', frame)
    cv2.resizeWindow('Frame',800,600)

    key = cv2.waitKey(1) & 0xFF

    # Press `q` to exit
    if key == ord("q"):
        break

# Clean
cap.release()
cv2.destroyAllWindows()
```

SSD_MobileNet_prototxt

```
name: "MobileNet-SSD"
input: "data"
input_shape {
  dim: 1
  dim: 3
  dim: 300
  dim: 300
}
layer {
  name: "conv0"
  type: "Convolution"
  bottom: "data"
  top: "conv0"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 32
    pad: 1
    kernel_size: 3
    stride: 2
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv0/relu"
  type: "ReLU"
  bottom: "conv0"
  top: "conv0"
}
layer {
  name: "conv1/dw"
  type: "Convolution"
  bottom: "conv0"
  top: "conv1/dw"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 32
    pad: 1
    kernel_size: 3
    group: 32
    engine: CAFFE
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv1/dw/relu"
  type: "ReLU"
  bottom: "conv1/dw"
  top: "conv1/dw"
}
layer {
  name: "conv1"
  type: "Convolution"
  bottom: "conv1/dw"
  top: "conv1"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 64
    kernel_size: 1
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
```

```
layer {
  name: "conv1/relu"
  type: "ReLU"
  bottom: "conv1"
  top: "conv1"
}
layer {
  name: "conv2/dw"
  type: "Convolution"
  bottom: "conv1"
  top: "conv2/dw"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 64
    pad: 1
    kernel_size: 3
    stride: 2
    group: 64
    engine: CAFFE
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv2/dw/relu"
  type: "ReLU"
  bottom: "conv2/dw"
  top: "conv2/dw"
}
layer {
  name: "conv2"
  type: "Convolution"
  bottom: "conv2/dw"
  top: "conv2"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 128
    kernel_size: 1
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv2/relu"
  type: "ReLU"
  bottom: "conv2"
  top: "conv2"
}
layer {
  name: "conv3/dw"
  type: "Convolution"
  bottom: "conv2"
  top: "conv3/dw"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 128
    pad: 1
    kernel_size: 3
    group: 128
    engine: CAFFE
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv3/dw/relu"
  type: "ReLU"
  bottom: "conv3/dw"
  top: "conv3/dw"
}
layer {
  name: "conv3"
  type: "Convolution"
  bottom: "conv3/dw"
  top: "conv3"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 128
    kernel_size: 1
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv3/relu"
  type: "ReLU"
  bottom: "conv3"
  top: "conv3"
}
layer {
  name: "conv4/dw"
  type: "Convolution"
  bottom: "conv3"
  top: "conv4/dw"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 128
    pad: 1
    kernel_size: 3
    stride: 2
    group: 128
    engine: CAFFE
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv4/dw/relu"
  type: "ReLU"
  bottom: "conv4/dw"
  top: "conv4/dw"
}
layer {
  name: "conv4"
  type: "Convolution"
  bottom: "conv4/dw"
  top: "conv4"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 256
    kernel_size: 1
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv4/relu"
  type: "ReLU"
  bottom: "conv4"
  top: "conv4"
}
layer {
  name: "conv5/dw"
  type: "Convolution"
  bottom: "conv4"
  top: "conv5/dw"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 256
    pad: 1
    kernel_size: 3
    group: 256
    engine: CAFFE
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv5/dw/relu"
  type: "ReLU"
  bottom: "conv5/dw"
  top: "conv5/dw"
}
layer {
  name: "conv5"
  type: "Convolution"
  bottom: "conv5/dw"
  top: "conv5"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 256
    kernel_size: 1
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv5/relu"
  type: "ReLU"
  bottom: "conv5"
  top: "conv5"
}
layer {
  name: "conv6/dw"
  type: "Convolution"
  bottom: "conv5"
  top: "conv6/dw"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 256
    pad: 1
    kernel_size: 3
    stride: 2
    group: 256
    engine: CAFFE
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv6/dw/relu"
  type: "ReLU"
  bottom: "conv6/dw"
  top: "conv6/dw"
}
layer {
  name: "conv6"
  type: "Convolution"
  bottom: "conv6/dw"
  top: "conv6"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 512
    kernel_size: 1
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv6/relu"
  type: "ReLU"
  bottom: "conv6"
  top: "conv6"
}
layer {
  name: "conv7/dw"
  type: "Convolution"
  bottom: "conv6"
  top: "conv7/dw"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 512
    pad: 1
    kernel_size: 3
    group: 512
    engine: CAFFE
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv7/dw/relu"
  type: "ReLU"
  bottom: "conv7/dw"
  top: "conv7/dw"
}
layer {
  name: "conv7"
  type: "Convolution"
  bottom: "conv7/dw"
  top: "conv7"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 512
    kernel_size: 1
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv7/relu"
  type: "ReLU"
  bottom: "conv7"
  top: "conv7"
}
layer {
  name: "conv8/dw"
  type: "Convolution"
  bottom: "conv7"
  top: "conv8/dw"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 512
    pad: 1
    kernel_size: 3
    group: 512
    engine: CAFFE
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv8/dw/relu"
  type: "ReLU"
  bottom: "conv8/dw"
  top: "conv8/dw"
}
layer {
  name: "conv8"
  type: "Convolution"
  bottom: "conv8/dw"
  top: "conv8"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 512
    kernel_size: 1
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv8/relu"
  type: "ReLU"
  bottom: "conv8"
  top: "conv8"
}
layer {
  name: "conv9/dw"
  type: "Convolution"
  bottom: "conv8"
  top: "conv9/dw"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 512
    pad: 1
    kernel_size: 3
    group: 512
    engine: CAFFE
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv9/dw/relu"
  type: "ReLU"
  bottom: "conv9/dw"
  top: "conv9/dw"
}
layer {
  name: "conv9"
  type: "Convolution"
  bottom: "conv9/dw"
  top: "conv9"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 512
    kernel_size: 1
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv9/relu"
  type: "ReLU"
  bottom: "conv9"
  top: "conv9"
}
layer {
  name: "conv10/dw"
  type: "Convolution"
  bottom: "conv9"
  top: "conv10/dw"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 512
    pad: 1
    kernel_size: 3
    group: 512
    engine: CAFFE
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv10/dw/relu"
  type: "ReLU"
  bottom: "conv10/dw"
  top: "conv10/dw"
}
layer {
  name: "conv10"
  type: "Convolution"
  bottom: "conv10/dw"
  top: "conv10"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 512
    kernel_size: 1
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv10/relu"
  type: "ReLU"
  bottom: "conv10"
  top: "conv10"
}
layer {
  name: "conv11/dw"
  type: "Convolution"
  bottom: "conv10"
  top: "conv11/dw"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 512
    pad: 1
    kernel_size: 3
    group: 512
    engine: CAFFE
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv11/dw/relu"
  type: "ReLU"
  bottom: "conv11/dw"
  top: "conv11/dw"
}
layer {
  name: "conv11"
  type: "Convolution"
  bottom: "conv11/dw"
  top: "conv11"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 512
    kernel_size: 1
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv11/relu"
  type: "ReLU"
  bottom: "conv11"
  top: "conv11"
}
layer {
  name: "conv12/dw"
  type: "Convolution"
  bottom: "conv11"
  top: "conv12/dw"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 512
    pad: 1
    kernel_size: 3
    stride: 2
    group: 512
    engine: CAFFE
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv12/dw/relu"
  type: "ReLU"
  bottom: "conv12/dw"
  top: "conv12/dw"
}
layer {
  name: "conv12"
  type: "Convolution"
  bottom: "conv12/dw"
  top: "conv12"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 1024
    kernel_size: 1
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv12/relu"
  type: "ReLU"
  bottom: "conv12"
  top: "conv12"
}
layer {
  name: "conv13/dw"
  type: "Convolution"
  bottom: "conv12"
  top: "conv13/dw"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 1024
    pad: 1
    kernel_size: 3
    group: 1024
    engine: CAFFE
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv13/dw/relu"
  type: "ReLU"
  bottom: "conv13/dw"
  top: "conv13/dw"
}
layer {
  name: "conv13"
  type: "Convolution"
  bottom: "conv13/dw"
  top: "conv13"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 1024
    kernel_size: 1
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv13/relu"
  type: "ReLU"
  bottom: "conv13"
  top: "conv13"
}
layer {
  name: "conv14_1"
  type: "Convolution"
  bottom: "conv13"
  top: "conv14_1"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 256
    kernel_size: 1
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv14_1/relu"
  type: "ReLU"
  bottom: "conv14_1"
  top: "conv14_1"
}
layer {
  name: "conv14_2"
  type: "Convolution"
  bottom: "conv14_1"
  top: "conv14_2"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 512
    pad: 1
    kernel_size: 3
    stride: 2
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv14_2/relu"
  type: "ReLU"
  bottom: "conv14_2"
  top: "conv14_2"
}
layer {
  name: "conv15_1"
  type: "Convolution"
  bottom: "conv14_2"
  top: "conv15_1"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 128
    kernel_size: 1
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv15_1/relu"
  type: "ReLU"
  bottom: "conv15_1"
  top: "conv15_1"
}
layer {
  name: "conv15_2"
  type: "Convolution"
  bottom: "conv15_1"
  top: "conv15_2"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 256
    pad: 1
    kernel_size: 3
    stride: 2
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv15_2/relu"
  type: "ReLU"
  bottom: "conv15_2"
  top: "conv15_2"
}
```

```
layer {
  name: "conv16_1"
  type: "Convolution"
  bottom: "conv15_2"
  top: "conv16_1"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 128
    kernel_size: 1
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv16_1/relu"
  type: "ReLU"
  bottom: "conv16_1"
  top: "conv16_1"
}
layer {
  name: "conv16_2"
  type: "Convolution"
  bottom: "conv16_1"
  top: "conv16_2"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 256
    pad: 1
    kernel_size: 3
    stride: 2
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv16_2/relu"
  type: "ReLU"
  bottom: "conv16_2"
  top: "conv16_2"
}
layer {
  name: "conv17_1"
  type: "Convolution"
  bottom: "conv16_2"
  top: "conv17_1"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 64
    kernel_size: 1
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv17_1/relu"
  type: "ReLU"
  bottom: "conv17_1"
  top: "conv17_1"
}
layer {
  name: "conv17_2"
  type: "Convolution"
  bottom: "conv17_1"
  top: "conv17_2"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 128
    pad: 1
    kernel_size: 3
    stride: 2
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv17_2/relu"
  type: "ReLU"
  bottom: "conv17_2"
  top: "conv17_2"
}
layer {
  name: "conv11_mbox_loc"
  type: "Convolution"
  bottom: "conv11"
  top: "conv11_mbox_loc"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 12
    kernel_size: 1
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
```

```
layer {
  name: "conv11_mbox_loc_perm"
  type: "Permute"
  bottom: "conv11_mbox_loc"
  top: "conv11_mbox_loc_perm"
  permute_param {
    order: 0
    order: 2
    order: 3
    order: 1
  }
}
layer {
  name: "conv11_mbox_loc_flat"
  type: "Flatten"
  bottom: "conv11_mbox_loc_perm"
  top: "conv11_mbox_loc_flat"
  flatten_param {
    axis: 1
  }
}
layer {
  name: "conv11_mbox_conf"
  type: "Convolution"
  bottom: "conv11"
  top: "conv11_mbox_conf"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 63
    kernel_size: 1
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv11_mbox_conf_perm"
  type: "Permute"
  bottom: "conv11_mbox_conf"
  top: "conv11_mbox_conf_perm"
  permute_param {
    order: 0
    order: 2
    order: 3
    order: 1
  }
}
layer {
  name: "conv11_mbox_conf_flat"
  type: "Flatten"
  bottom: "conv11_mbox_conf_perm"
  top: "conv11_mbox_conf_flat"
  flatten_param {
    axis: 1
  }
}
layer {
  name: "conv11_mbox_priorbox"
  type: "PriorBox"
  bottom: "conv11"
  bottom: "data"
  top: "conv11_mbox_priorbox"
  prior_box_param {
    min_size: 60.0
    aspect_ratio: 2.0
    flip: true
    clip: false
    variance: 0.1
    variance: 0.1
    variance: 0.2
    variance: 0.2
    offset: 0.5
  }
}
layer {
  name: "conv13_mbox_loc"
  type: "Convolution"
  bottom: "conv13"
  top: "conv13_mbox_loc"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 24
    kernel_size: 1
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv13_mbox_loc_perm"
  type: "Permute"
  bottom: "conv13_mbox_loc"
  top: "conv13_mbox_loc_perm"
  permute_param {
    order: 0
    order: 2
    order: 3
    order: 1
  }
}
layer {
  name: "conv13_mbox_loc_flat"
  type: "Flatten"
  bottom: "conv13_mbox_loc_perm"
  top: "conv13_mbox_loc_flat"
  flatten_param {
    axis: 1
  }
}
layer {
  name: "conv13_mbox_conf"
  type: "Convolution"
  bottom: "conv13"
  top: "conv13_mbox_conf"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 126
    kernel_size: 1
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv13_mbox_conf_perm"
  type: "Permute"
  bottom: "conv13_mbox_conf"
  top: "conv13_mbox_conf_perm"
  permute_param {
    order: 0
    order: 2
    order: 3
    order: 1
  }
}
layer {
  name: "conv13_mbox_conf_flat"
  type: "Flatten"
  bottom: "conv13_mbox_conf_perm"
  top: "conv13_mbox_conf_flat"
  flatten_param {
    axis: 1
  }
}
layer {
  name: "conv13_mbox_priorbox"
  type: "PriorBox"
  bottom: "conv13"
  bottom: "data"
  top: "conv13_mbox_priorbox"
  prior_box_param {
    min_size: 105.0
    max_size: 150.0
    aspect_ratio: 2.0
    aspect_ratio: 3.0
    flip: true
    clip: false
    variance: 0.1
    variance: 0.1
    variance: 0.2
    variance: 0.2
    offset: 0.5
  }
}
layer {
  name: "conv14_2_mbox_loc"
  type: "Convolution"
  bottom: "conv14_2"
  top: "conv14_2_mbox_loc"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 24
    kernel_size: 1
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv14_2_mbox_loc_perm"
  type: "Permute"
  bottom: "conv14_2_mbox_loc"
  top: "conv14_2_mbox_loc_perm"
  permute_param {
    order: 0
    order: 2
    order: 3
    order: 1
  }
}
layer {
  name: "conv14_2_mbox_loc_flat"
  type: "Flatten"
  bottom: "conv14_2_mbox_loc_perm"
  top: "conv14_2_mbox_loc_flat"
  flatten_param {
    axis: 1
  }
}
layer {
  name: "conv14_2_mbox_conf"
  type: "Convolution"
  bottom: "conv14_2"
  top: "conv14_2_mbox_conf"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 126
    kernel_size: 1
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv14_2_mbox_conf_perm"
  type: "Permute"
  bottom: "conv14_2_mbox_conf"
  top: "conv14_2_mbox_conf_perm"
  permute_param {
    order: 0
    order: 2
    order: 3
    order: 1
  }
}
layer {
  name: "conv14_2_mbox_conf_flat"
  type: "Flatten"
  bottom: "conv14_2_mbox_conf_perm"
  top: "conv14_2_mbox_conf_flat"
  flatten_param {
    axis: 1
  }
}
layer {
  name: "conv14_2_mbox_priorbox"
  type: "PriorBox"
  bottom: "conv14_2"
  bottom: "data"
  top: "conv14_2_mbox_priorbox"
  prior_box_param {
    min_size: 150.0
    max_size: 195.0
    aspect_ratio: 2.0
    aspect_ratio: 3.0
    flip: true
    clip: false
    variance: 0.1
    variance: 0.1
    variance: 0.2
    variance: 0.2
    offset: 0.5
  }
}
layer {
  name: "conv15_2_mbox_loc"
  type: "Convolution"
  bottom: "conv15_2"
  top: "conv15_2_mbox_loc"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 24
    kernel_size: 1
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv15_2_mbox_loc_perm"
  type: "Permute"
  bottom: "conv15_2_mbox_loc"
  top: "conv15_2_mbox_loc_perm"
  permute_param {
    order: 0
    order: 2
    order: 3
    order: 1
  }
}
layer {
  name: "conv15_2_mbox_loc_flat"
  type: "Flatten"
  bottom: "conv15_2_mbox_loc_perm"
  top: "conv15_2_mbox_loc_flat"
  flatten_param {
    axis: 1
  }
}
layer {
  name: "conv15_2_mbox_conf"
  type: "Convolution"
  bottom: "conv15_2"
  top: "conv15_2_mbox_conf"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 126
    kernel_size: 1
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv15_2_mbox_conf_perm"
  type: "Permute"
  bottom: "conv15_2_mbox_conf"
  top: "conv15_2_mbox_conf_perm"
  permute_param {
    order: 0
    order: 2
    order: 3
    order: 1
  }
}
layer {
  name: "conv15_2_mbox_conf_flat"
  type: "Flatten"
  bottom: "conv15_2_mbox_conf_perm"
  top: "conv15_2_mbox_conf_flat"
  flatten_param {
    axis: 1
  }
}
layer {
  name: "conv15_2_mbox_priorbox"
  type: "PriorBox"
  bottom: "conv15_2"
  bottom: "data"
  top: "conv15_2_mbox_priorbox"
  prior_box_param {
    min_size: 195.0
    max_size: 240.0
    aspect_ratio: 2.0
    aspect_ratio: 3.0
    flip: true
    clip: false
    variance: 0.1
    variance: 0.1
    variance: 0.2
    variance: 0.2
    offset: 0.5
  }
}
layer {
  name: "conv16_2_mbox_loc"
  type: "Convolution"
  bottom: "conv16_2"
  top: "conv16_2_mbox_loc"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 24
    kernel_size: 1
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv16_2_mbox_loc_perm"
  type: "Permute"
  bottom: "conv16_2_mbox_loc"
  top: "conv16_2_mbox_loc_perm"
  permute_param {
    order: 0
    order: 2
    order: 3
    order: 1
  }
}
layer {
  name: "conv16_2_mbox_loc_flat"
  type: "Flatten"
  bottom: "conv16_2_mbox_loc_perm"
  top: "conv16_2_mbox_loc_flat"
  flatten_param {
    axis: 1
  }
}
layer {
  name: "conv16_2_mbox_conf"
  type: "Convolution"
  bottom: "conv16_2"
  top: "conv16_2_mbox_conf"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 126
    kernel_size: 1
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv16_2_mbox_conf_perm"
  type: "Permute"
  bottom: "conv16_2_mbox_conf"
  top: "conv16_2_mbox_conf_perm"
  permute_param {
    order: 0
    order: 2
    order: 3
    order: 1
  }
}
layer {
  name: "conv16_2_mbox_conf_flat"
  type: "Flatten"
  bottom: "conv16_2_mbox_conf_perm"
  top: "conv16_2_mbox_conf_flat"
  flatten_param {
    axis: 1
  }
}
layer {
  name: "conv16_2_mbox_priorbox"
  type: "PriorBox"
  bottom: "conv16_2"
  bottom: "data"
  top: "conv16_2_mbox_priorbox"
  prior_box_param {
    min_size: 240.0
    max_size: 285.0
    aspect_ratio: 2.0
    aspect_ratio: 3.0
    flip: true
    clip: false
    variance: 0.1
    variance: 0.1
    variance: 0.2
    variance: 0.2
    offset: 0.5
  }
}
layer {
  name: "conv17_2_mbox_loc"
  type: "Convolution"
  bottom: "conv17_2"
  top: "conv17_2_mbox_loc"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 24
    kernel_size: 1
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv17_2_mbox_loc_perm"
  type: "Permute"
  bottom: "conv17_2_mbox_loc"
  top: "conv17_2_mbox_loc_perm"
  permute_param {
    order: 0
    order: 2
    order: 3
    order: 1
  }
}
layer {
  name: "conv17_2_mbox_loc_flat"
  type: "Flatten"
  bottom: "conv17_2_mbox_loc_perm"
  top: "conv17_2_mbox_loc_flat"
  flatten_param {
    axis: 1
  }
}
layer {
  name: "conv17_2_mbox_conf"
  type: "Convolution"
  bottom: "conv17_2"
  top: "conv17_2_mbox_conf"
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  convolution_param {
    num_output: 126
    kernel_size: 1
    weight_filler {
      type: "msra"
    }
    bias_filler {
      type: "constant"
      value: 0.0
    }
  }
}
layer {
  name: "conv17_2_mbox_conf_perm"
  type: "Permute"
  bottom: "conv17_2_mbox_conf"
  top: "conv17_2_mbox_conf_perm"
  permute_param {
    order: 0
    order: 2
    order: 3
    order: 1
  }
}
layer {
  name: "conv17_2_mbox_conf_flat"
  type: "Flatten"
  bottom: "conv17_2_mbox_conf_perm"
  top: "conv17_2_mbox_conf_flat"
  flatten_param {
    axis: 1
  }
}
layer {
  name: "conv17_2_mbox_priorbox"
  type: "PriorBox"
  bottom: "conv17_2"
  bottom: "data"
  top: "conv17_2_mbox_priorbox"
  prior_box_param {
    min_size: 285.0
    max_size: 300.0
    aspect_ratio: 2.0
    aspect_ratio: 3.0
    flip: true
    clip: false
    variance: 0.1
    variance: 0.1
    variance: 0.2
    variance: 0.2
    offset: 0.5
  }
}
layer {
  name: "mbox_loc"
  type: "Concat"
  bottom: "conv11_mbox_loc_flat"
  bottom: "conv13_mbox_loc_flat"
  bottom: "conv14_2_mbox_loc_flat"
  bottom: "conv15_2_mbox_loc_flat"
  bottom: "conv16_2_mbox_loc_flat"
  bottom: "conv17_2_mbox_loc_flat"
  top: "mbox_loc"
  concat_param {
    axis: 1
  }
}
layer {
  name: "mbox_conf"
  type: "Concat"
  bottom: "conv11_mbox_conf_flat"
  bottom: "conv13_mbox_conf_flat"
  bottom: "conv14_2_mbox_conf_flat"
  bottom: "conv15_2_mbox_conf_flat"
  bottom: "conv16_2_mbox_conf_flat"
  bottom: "conv17_2_mbox_conf_flat"
  top: "mbox_conf"
  concat_param {
    axis: 1
  }
}
layer {
  name: "mbox_priorbox"
  type: "Concat"
  bottom: "conv11_mbox_priorbox"
  bottom: "conv13_mbox_priorbox"
  bottom: "conv14_2_mbox_priorbox"
  bottom: "conv15_2_mbox_priorbox"
  bottom: "conv16_2_mbox_priorbox"
  bottom: "conv17_2_mbox_priorbox"
  top: "mbox_priorbox"
  concat_param {
    axis: 2
  }
}
layer {
  name: "mbox_conf_reshape"
  type: "Reshape"
  bottom: "mbox_conf"
  top: "mbox_conf_reshape"
  reshape_param {
    shape {
      dim: 0
      dim: -1
      dim: 21
    }
  }
}
layer {
  name: "mbox_conf_softmax"
  type: "Softmax"
  bottom: "mbox_conf_reshape"
  top: "mbox_conf_softmax"
  softmax_param {
    axis: 2
  }
}
layer {
  name: "mbox_conf_flatten"
  type: "Flatten"
  bottom: "mbox_conf_softmax"
  top: "mbox_conf_flatten"
  flatten_param {
    axis: 1
  }
}
layer {
  name: "detection_out"
  type: "DetectionOutput"
  bottom: "mbox_loc"
  bottom: "mbox_conf_flatten"
  bottom: "mbox_priorbox"
  top: "detection_out"
  include {
    phase: TEST
  }
  detection_output_param {
    num_classes: 21
    share_location: true
    background_label_id: 0
    nms_param {
      nms_threshold: 0.45
      top_k: 100
    }
    code_type: CENTER_SIZE
    keep_top_k: 100
    confidence_threshold: 0.25
  }
}
```

Chapter 5

5.1 CONCLUSION

Corporate giants from various verticals are turning to AI and ML, leveraging technology at the service of humanity amid the pandemic. Digital product development companies are launching mask detection API services that enable developers to build a face mask detection system quickly to serve the people amid the crisis.

We proposed an approach that uses computer vision and MobileNet V2, SSD architecture to help maintain a secure environment and ensure individuals protection by automatically monitoring public places to avoid the spread of the COVID-19virus and assist the police by minimizing their physical surveillance work in containment zones and public areas where surveillance is required by means of camera feeds. The technology assures reliable and real-time face detection of users wearing masks. Besides, the system is easy to deploy into any existing system of a business while keeping the safety of the people involved. This proposed system will operate in an efficient manner in the current situation when the lockdown is eased and helps to track public places easily in an automated manner. We have addressed in depth the tracking of social distancing and the identification off face masks that help to ensure human health. So, the Social Distancing and face mask detection system is going to be the leading digital solution for most industries, especially retail, healthcare, temples, shopping complex, metro stations. airports and corporate sectors.

REFERENCES

1. M. Loey, G. Manogaran, M. H. N. Taha, and N. E. M. Khalifa, “A hybrid deep transfer learning model with machine learning methods for face mask detection in the era of the COVID-19 pandemic,” *Measurement*, vol. 167, Article ID 108288, 2021.
   View at: Publisher Site | Google Scholar
2. B. Qin and D. Li, “Identifying facemask-wearing condition using image super-resolution with classification network to prevent COVID-19,” *Sensors*, vol. 20, no. 18, p. 5236, 2020.
   View at: Publisher Site | Google Scholar
3. Z. Wang, P. Wang, P. C. Louis, L. E. Wheless, and Y. Huo, “Wearmask: fast In-browser face mask detection with serverless edge computing for COVID-19,” 2021, https://arxiv.org/abs/2101.00784.
   View at: Google Scholar
4. X. Zhang, H. Saleh, E. M. Younis, R. Sahal, and A. A. Ali, “Predicting coronavirus pandemic in real-time using machine learning and big data streaming system,” *Complexity*, vol. 2020, Article ID 6688912, 10 pages, 2020.
   View at: Google Scholar
5. M. Razavi, H. Alikhani, V. Janfaza, B. Sadeghi, and E. Alikhani, “An automatic system to monitor the physical distance and face mask wearing of construction workers in COVID-19 pandemic,” 2021, https://arxiv.org/abs/2101.01373.
   View at: Google Scholar
6. S. K. Dey, A. Howlader, and C. Deb, “MobileNet mask: a multi-phase face mask detection model to prevent person-to-person transmission of SARS-CoV-2,” in *Proceedings of International Conference on Trends in Computational and Cognitive Engineering*, pp. 603–613, Springer, Dhaka, Bangladesh, December 2021.
   View at: Publisher Site | Google Scholar
7. K. M. A. Kabir and J. Tanimoto, “Evolutionary game theory modelling to represent the behavioural dynamics of economic shutdowns and shield immunity in the COVID-19 pandemic,” *Royal Society Open Science*, vol. 7, no. 9, Article ID 201095, 2020.
   View at: Publisher Site | Google Scholar
8. R. Blundell, M. Costa Dias, R. Joyce, and X. Xu, “COVID-19 and inequalities*,” *Fiscal Studies*, vol. 41, no. 2, pp. 291–319, 2020.
   View at: Publisher Site | Google Scholar
9. J. S. Weitz, S. J. Beckett, A. R. Coenen et al., “Modeling shield immunity to reduce COVID-19 epidemic spread,” *Nature Medicine*, vol. 26, no. 6, pp. 849–854, 2020.
   View at: Publisher Site | Google Scholar

10. D. Sridhar and D. Gurdasani, “Herd immunity by infection is not an option,” *Science*, vol. 371, no. 6526, pp. 230-231, 2021.
View at: Publisher Site | Google Scholar
11. J. Wong and N. Wong, “The economics and accounting for COVID-19 wage subsidy and other government grants,” *Pacific Accounting Review*, vol. 33, no. 2, pp. 199–211, 2021.
View at: Google Scholar
12. M. Piraveenan, S. Sawleshwarkar, M. Walsh et al., “Optimal governance and implementation of vaccination programmes to contain the COVID-19 pandemic,” *Royal Society Open Science*, vol. 8, no. 6, Article ID 210429, 2021.
View at: Publisher Site | Google Scholar
13. A. Echtioui, W. Zouch, M. Ghorbel, C. Mhiri, and H. Hamam, “Detection methods of COVID-19,” *SLAS TECHNOLOGY: Translating Life Sciences Innovation*, vol. 25, no. 6, pp. 566–572, 2020.
View at: Publisher Site | Google Scholar
14. N. Abbassi, R. Helaly, M. A. Hajjaji, and A. Mtibaa, “A deep learning facial emotion classification system: a VGGNet-19 based approach,” in *Proceedings of the 2020 20th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*, pp. 271–276, IEEE, Monastir, Tunisia, December 2020.
View at: Publisher Site | Google Scholar
15. R. Helaly, M. A. Hajjaji, F. M’Sahli, and A. Mtibaa, “Deep convolution neural network implementation for emotion recognition system,” in *Proceedings of the 2020 20th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*, pp. 261–265, IEEE, Monastir, Tunisia, December 2020.
View at: Publisher Site | Google Scholar
16. S. Bouaafia, S. Messaoud, R. Khemiri, and F. E. Sayadi, “COVID-19 recognition based on deep transfer learning,” in *Proceedings of the 2021 IEEE International Conference on Design & Test of Integrated Micro & Nano-Systems (DTS)*, pp. 1–4, IEEE, Sfax, Tunisia, July 2021.
View at: Google Scholar
17. I. Khriji, A. Ammari, S. Messaoud, S. Bouaafia, A. Maraoui, and M. Machhout, “COVID-19 recognition based on patient’s coughing and breathing patterns analysis: deep learning approach,” in *Proceedings of the 2021 29th Conference of Open Innovations Association (FRUCT)*, pp. 185–191, IEEE, Tampere, Finland, May 2021.
View at: Google Scholar

18. T. Ozturk, M. Talo, E. A. Yildirim, U. B. Baloglu, O. Yildirim, and U. Rajendra Acharya, “Automated detection of COVID-19 cases using deep neural networks with X-ray images,” *Computers in Biology and Medicine*, vol. 121, Article ID 103792, 2020.
View at: Publisher Site | Google Scholar
19. L. Wang, Z. Q. Lin, and A. Wong, “Covid-net: a tailored deep convolutional neural network design for detection of covid-19 cases from chest x-ray images,” *Scientific Reports*, vol. 10, no. 1, pp. 1–12, 2020.
View at: Publisher Site | Google Scholar
20. V. Shah, R. Keniya, A. Shridharani, M. Punjabi, J. Shah, and N. Mehendale, “Diagnosis of COVID-19 using CT scan images and deep learning techniques,” *Emergency Radiology*, vol. 49, pp. 1–9, 2021.
View at: Publisher Site | Google Scholar
21. A. Sedik, M. Hammad, F. E. Abd El-Samie, B. B. Gupta, and A. A. Abd El-Latif, “Efficient deep learning approach for augmented detection of Coronavirus disease,” *Neural Computing & Applications*, vol. 18, pp. 1–18, 2021.
View at: Publisher Site | Google Scholar
22. M. Loey, G. Manogaran, M. H. N. Taha, and N. E. M. Khalifa, “Fighting against COVID-19: a novel deep learning model based on YOLO-v2 with ResNet-50 for medical face mask detection,” *Sustainable Cities and Society*, vol. 65, Article ID 102600, 2021.
View at: Publisher Site | Google Scholar
23. A. Nieto-Rodríguez, M. Mucientes, and V. M. Brea, “System for medical mask detection in the operating room through facial attributes,” in *Proceedings of the Iberian Conference on Pattern Recognition and Image Analysis*, pp. 138–145, Springer, Santiago de Compostela, Spain, June 2015.
View at: Publisher Site | Google Scholar
24. M. Perc, M. Ozer, and J. Hojnik, “Social and juristic challenges of artificial intelligence,” *Palgrave Communications*, vol. 5, no. 1, pp. 1–7, 2019.
View at: Publisher Site | Google Scholar
25. S. Saponara, A. Elhanashi, and A. Gagliardi, “Implementing a real-time, AI-based, people detection and social distancing measuring system for Covid-19,” *Journal of Real-Time Image Processing*, vol. 11, pp. 1–11, 2021.
View at: Publisher Site | Google Scholar
26. M. Coşkun, A. Uçar, O. Yildirim, and Y. Demir, “November). Face recognition based on convolutional neural network,” in *Proceedings of the 2017 International Conference on Modern Electrical and Energy Systems (MEES)*, pp. 376–379, IEEE, Kremenchuk, Ukraine, November 2017.
View at: Google Scholar

27. F. Sultana, A. Sufian, and P. Dutta, “Advancements in image classification using convolutional neural network,” in *Proceedings of the 2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*, pp. 122–129, IEEE, Kolkata, India, November 2018.
View at: Google Scholar
28. S. M. Jameel, M. A. Hashmani, M. Rehman, and A. Budiman, “Adaptive CNN ensemble for complex multispectral image analysis,” *Complexity*, vol. 2020, Article ID 83561989, 21 pages, 2020.
View at: Publisher Site | Google Scholar
29. Q. Zhao, S. Lyu, B. Zhang, and W. Feng, “Multiactivation pooling method in convolutional neural networks for image recognition,” *Wireless Communications and Mobile Computing*, vol. 2018, Article ID 8196906, 15 pages, 2018.
View at: Publisher Site | Google Scholar
30. S. Albawi, T. A. Mohammed, and S. Al-Zawi, “Understanding of a convolutional neural network,” in *Proceedings of the 2017 International Conference on Engineering and Technology (ICET)*, pp. 1–6, Ieee, Antalya, Turkey, August 2017.
View at: Publisher Site | Google Scholar
31. F. Demir, D. A. Abdullah, and A. Sengur, “A new deep CNN model for environmental sound classification,” *IEEE Access*, vol. 8, pp. 66529–66537, 2020.
View at: Publisher Site | Google Scholar
32. https://drive.google.com/drive/folders/1IPwsC30wNAc74_GTXuEWX_F8m2n-ZBCH.
33. M. Grandini, E. Bagli, and G. Visani, “Metrics for multi-class classification: an overview,” 2020, https://arxiv.org/abs/2008.05756.
View at: Google Scholar
34. N. Hasan, Y. Bao, A. Shawon, and Y. Huang, “Densenet convolutional neural networks application for predicting COVID-19 using CT image,” *SN Computer Science*, vol. 2, no. 5, 2020.
View at: Publisher Site | Google Scholar
35. L. Sarker, M. M. Islam, T. Hannan, and Z. Ahmed, “Covid-densenet: a deep learning architecture to detect Covid-19 from chest radiology images,” vol. 21, 2020.
View at: Google Scholar
36. K. El Asnaoui and Y. Chawki, “Using X-ray images and deep learning for automated detection of coronavirus disease,” *Journal of Biomolecular Structure and Dynamics*, pp. 1–12, 2020.
View at: Publisher Site | Google Scholar
37. R. Jain, M. Gupta, S. Taneja, and D. J. Hemanth, “Deep learning based detection and analysis of COVID-19 on chest X-ray images,” *Applied Intelligence*, vol. 51, pp. 1–11, 2020.

View at: Publisher Site | Google Scholar

38. M. A. R. Ratul, M. T. Elahi, K. Yuan, and W. Lee, “RAM-Net: a residual attention MobileNet to detect COVID-19 cases from chest X-ray images,” in *Proceedings of the 2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 195–200, IEEE, Miami, FL, USA, December 2020.
    View at: Google Scholar
39. M. Toğaçar, B. Ergen, and Z. Cömert, “COVID-19 detection using deep learning models to exploit Social Mimic Optimization and structured chest X-ray images using fuzzy color and stacking approaches,” *Computers in Biology and Medicine*, vol. 121, Article ID 103805, 2020.
    View at: Publisher Site | Google Scholar
40. Z. Karhan and F. Akal, “Covid-19 classification using deep learning in chest X-ray images,” in *Proceedings of the 2020 Medical Technologies Congress (TIPTEKNO)*, pp. 1–4, IEEE, Antalya, Turkey, November 2020.
    View at: Google Scholar
41. M. M. Ahsan, M. T. Ahad, F. A. Soma et al., “Detecting SARS-CoV-2 from chest X-ray using artificial intelligence,” *IEEE Access*, vol. 9, pp. 35501–35513, 2021.
    View at: Publisher Site | Google Scholar
42. S. S. Paima, N. Hasanzadeh, A. Jodeiri, and &H. Soltanian-Zadeh, “Detection of COVID-19 from chest radiographs: comparison of four end-to-end trained deep learning models,” in *Proceedings of the 2020 27th National and 5th International Iranian Conference on Biomedical Engineering (ICBME)*, pp. 217–221, IEEE, Tehran, Iran, November 2020.
    View at: Google Scholar
43. A. P. Bradley, “The use of the area under the ROC curve in the evaluation of machine learning algorithms,” *Pattern Recognition*, vol. 30, no. 7, pp. 1145–1159, 1997.
    View at: Publisher Site | Google Scholar