Project: Facial Recognition and Mask Detection in the classroom

Course: IoT Application System
Professor: Kakani Vijay
TA: Baydadaev Shokhrukh
Inha Universty, 2022

Team Temirlan

Team leader: Temirlan Ulan 12204575

Team members:

Guliza Aitkulova 12204505

Daniiar Rayimkulov 12204510

Temurbek Akhrorov 12204574

Leyla Khamidova 12204514

Contents

The project:

1.

-Usage: in the classroom

-Face mask detection: for Covid-19 precautions

-Facial recognition:

1. Face detection
2. Face analysis
3. Converting the image to data
4. Finding a match

Face recognition: for attendance check

Libraries

- Keras
- Tensorflow(lite)
- OpenCV
- face_recognition
- NumPy

Keras

- High level, open source Python API developed by Google
- It makes the implementation of neural network less complicated
- Various backend neural network computation is supported
- Interface for TensorFlow

TensorFlow

- Open-source library created for deep learning applications
- Acquires data, trains models, serves predictions, and refines future results
- Used for neural networks and is best suited for dataflow programming across a range of tasks
- It is known for documentation and training support, scalable production and deployment options, multiple abstraction levels, and support for different platforms

- Open-source library for the computer vision, machine learning, and image processing
- Using it, one may analyze pictures and movies to find faces, objects, and even human handwriting
- Vector space is used to Identify image pattern and its various features and perform mathematical operations on these features

NumPy

- Python libary used for working with arrays
- Additionally, it provides functions for working with matrices, the Fourier transform, and the linear algebra domain
- The goal of numpy is to provide an array objects that are 50 times faster than traditional Python lists

face_recognition

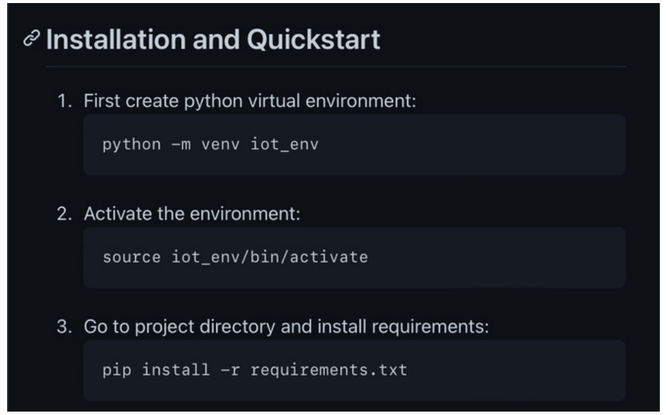
- recognizes and manipulates faces from Python of from the command line
- The model has an accuracy of 99.38% on the Labeled Faces in the Wild benchmark
- It is also possible to use it for applying digital make-up
- finds the face's locations inside a particular image

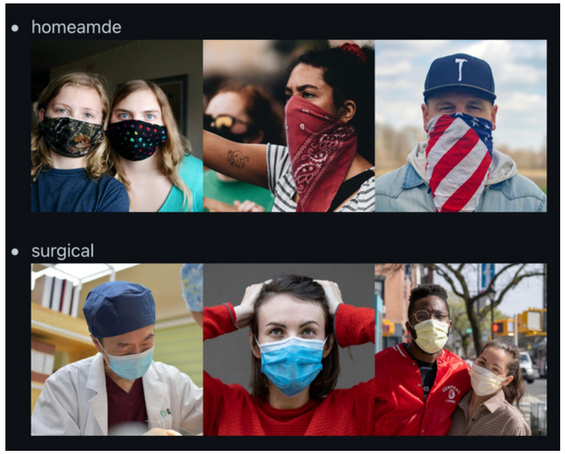
TensorFlow Lite

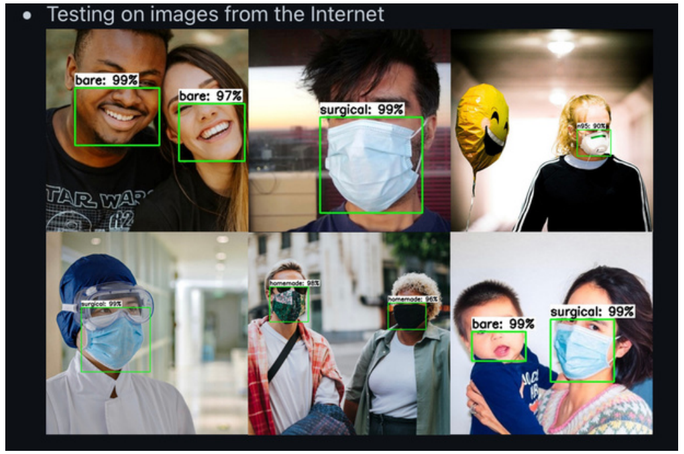
- uses TensorFlow models converted into a smaller, more efficient machine learning (ML) model format
- we can use pre-trained models with TensorFlow Lite, modify existing models, or build our own TensorFlow models and then convert them to TensorFlow Lite format
- TensorFlow Lite models can perform almost any task a regular TensorFlow model can do

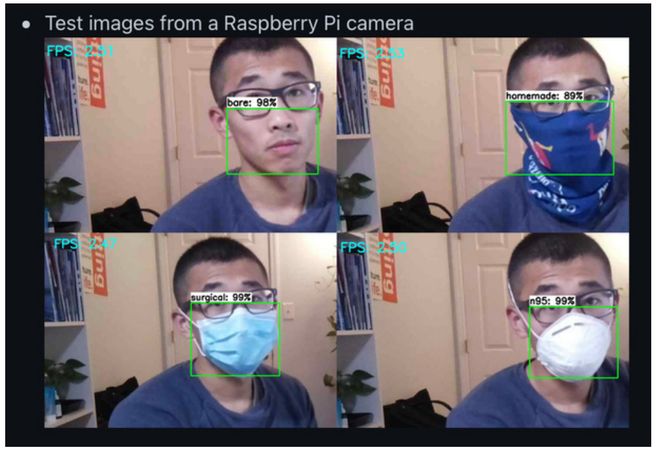
The implementation process:

1. Finding an open-source project and make it work
    -creating a virtual environment
    -installing dependencies
    -TensorFlow? ==> TensorFlow Lite
    - https://github.com/danieldanuega/mask-detector

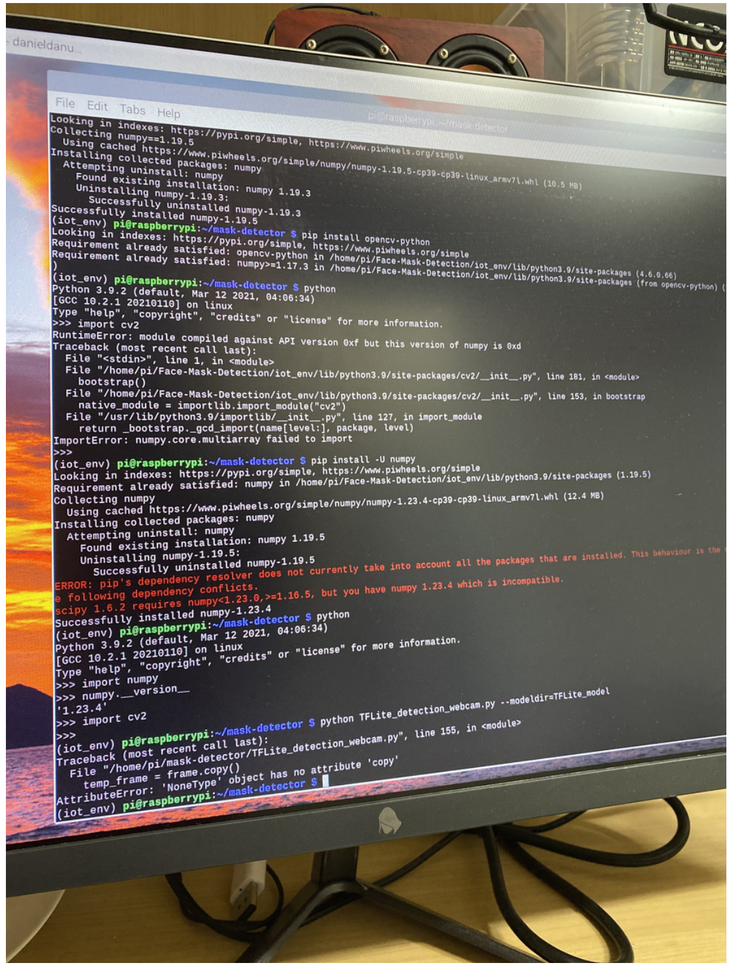








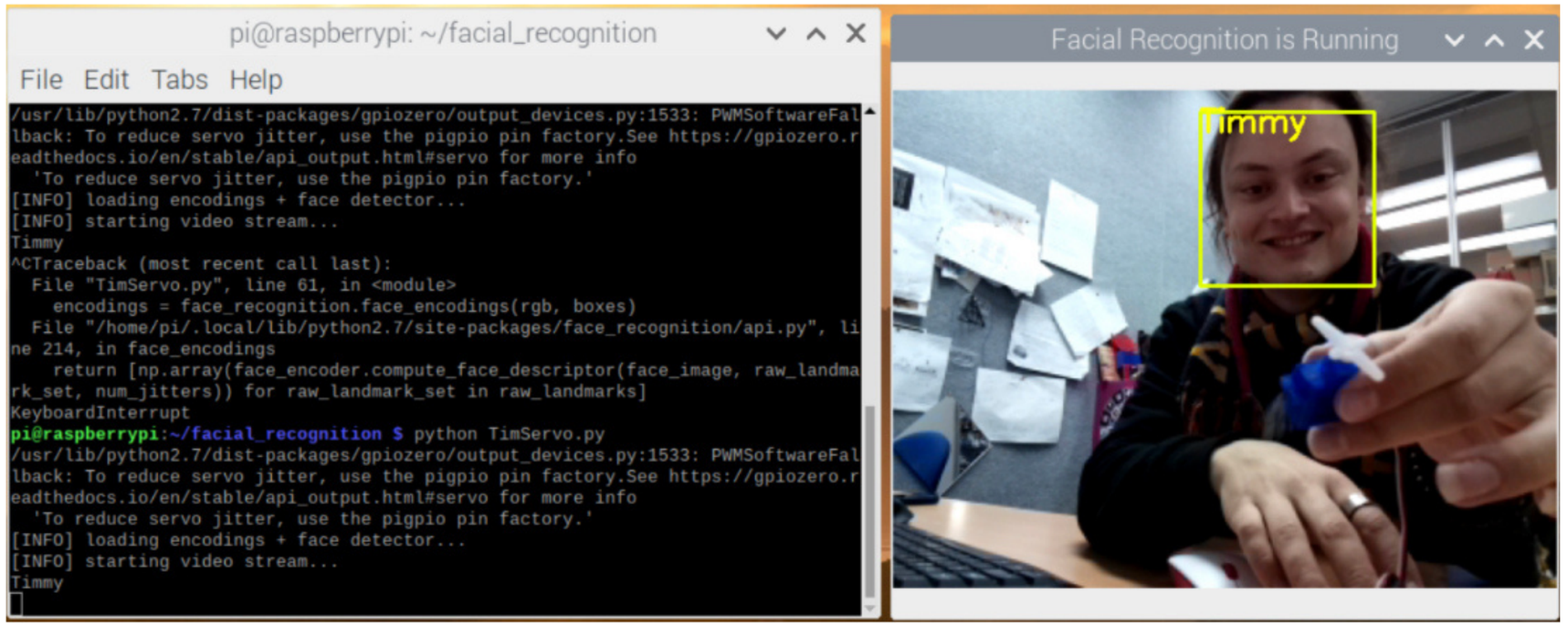
The first project: Face Mask Detection



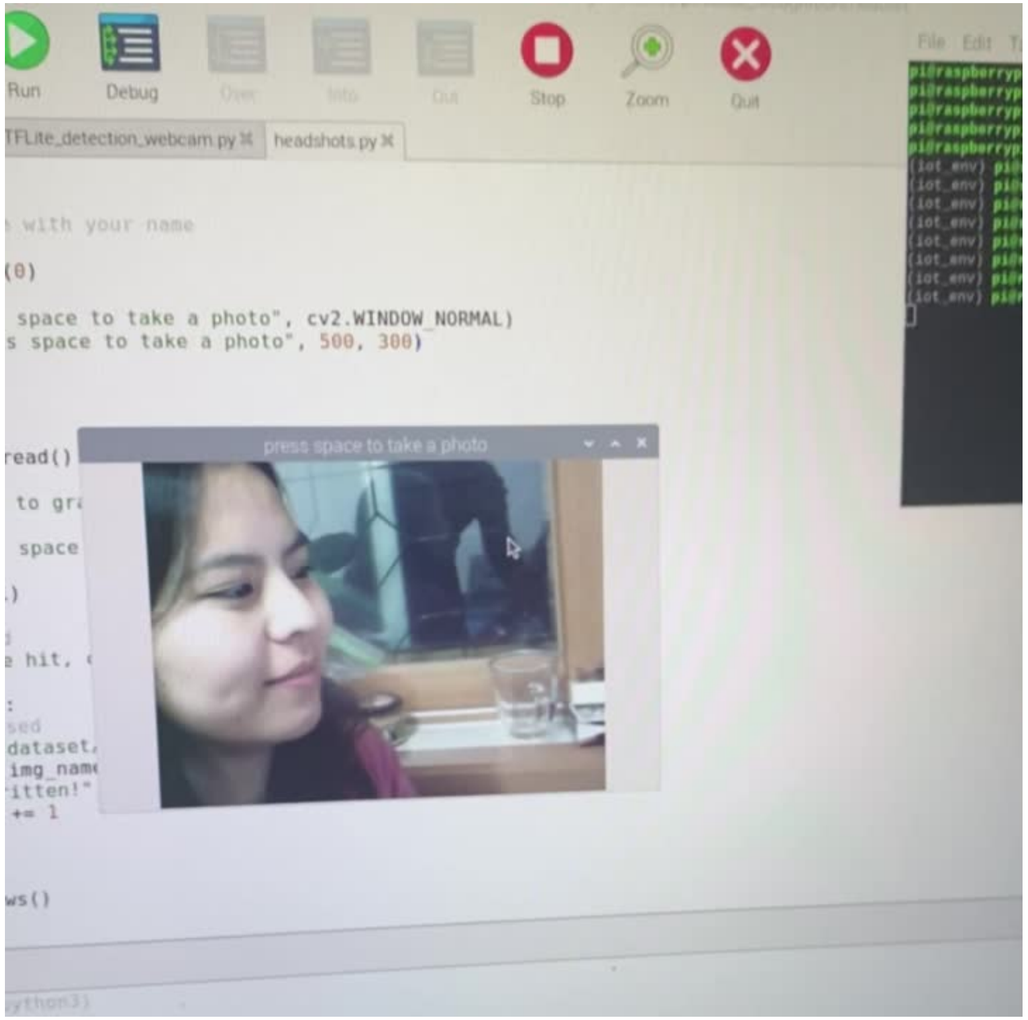
Adding more features: face recognition

-Face mask detection & face recognition in the classroom

-https://core-electronics.com.au/guides/face-identify-raspberry-pi/



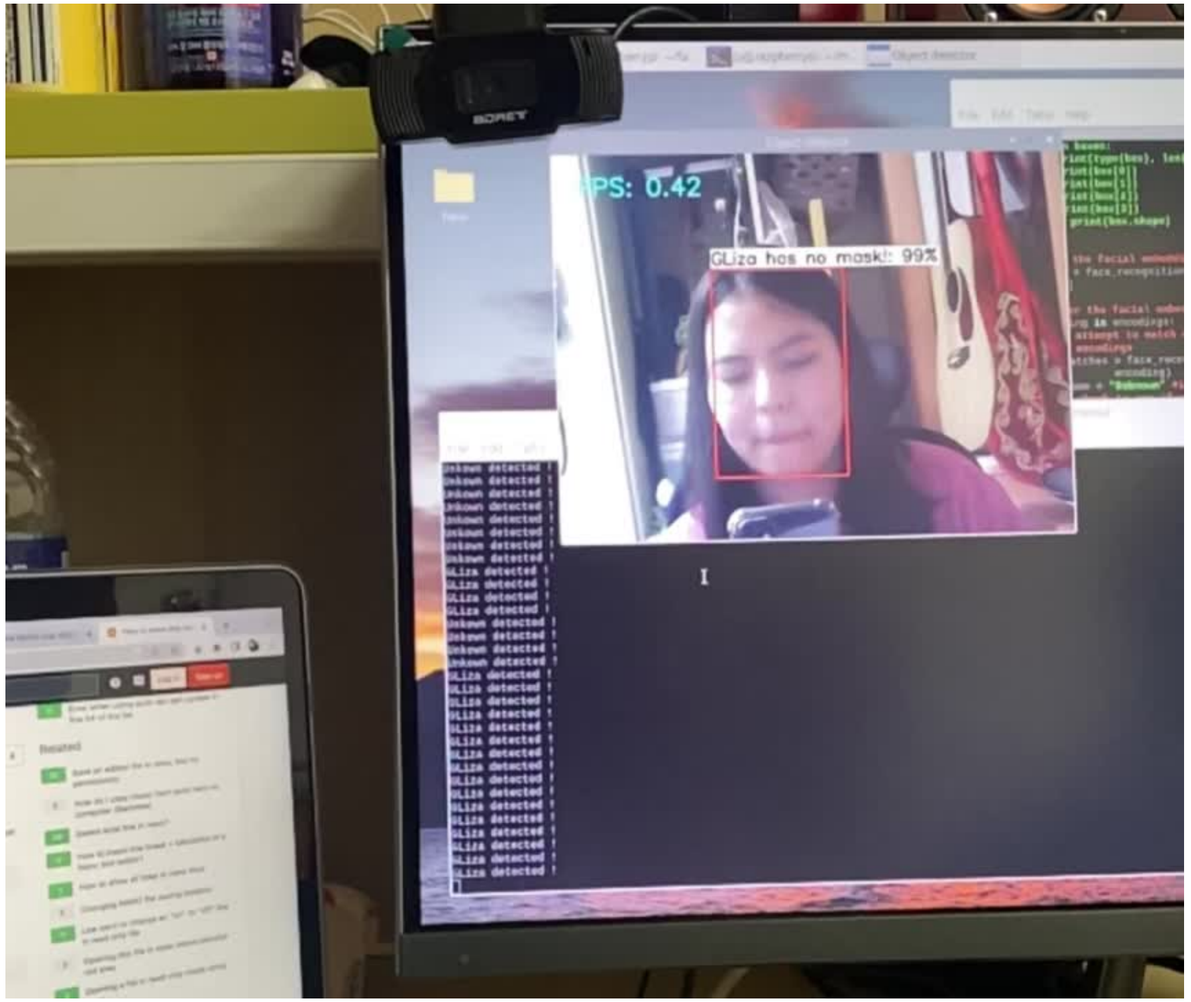
The second project: Face Recognition




```python
frame = vs.read()
frame = imutils.resize(frame, width=500)
# Detect the fce boxes
boxes = face_recognition.face_locations(frame)
# compute the facial embeddings for each face bounding box
encodings = face_recognition.face_encodings(frame, boxes)
names = []

# loop over the facial embeddings
for encoding in encodings:
    # attempt to match each face in the input image to our known
    # encodings
    matches = face_recognition.compare_faces(data["encodings"],
        encoding)
    name = "Unknown" #if face is not recognized, then print Unknown

    # check to see if we have found a match
    if True in matches:
        # find the indexes of all matched faces then initialize a
        # dictionary to count the total number of times each face
        # was matched
        matchedIdxs = [i for (i, b) in enumerate(matches) if b]
        counts = {}

        # loop over the matched indexes and maintain a count for
        # each recognized face face
        for i in matchedIdxs:
            name = data["names"][i]
            counts[name] = counts.get(name, 0) + 1

        # determine the recognized face with the largest number
        # of votes (note: in the event of an unlikely tie Python
        # will select first entry in the dictionary)
        name = max(counts, key=counts.get)

        #If someone in your dataset is identified, print their name on the screen
        if currentname != name:
            currentname = name
            print(currentname)
```

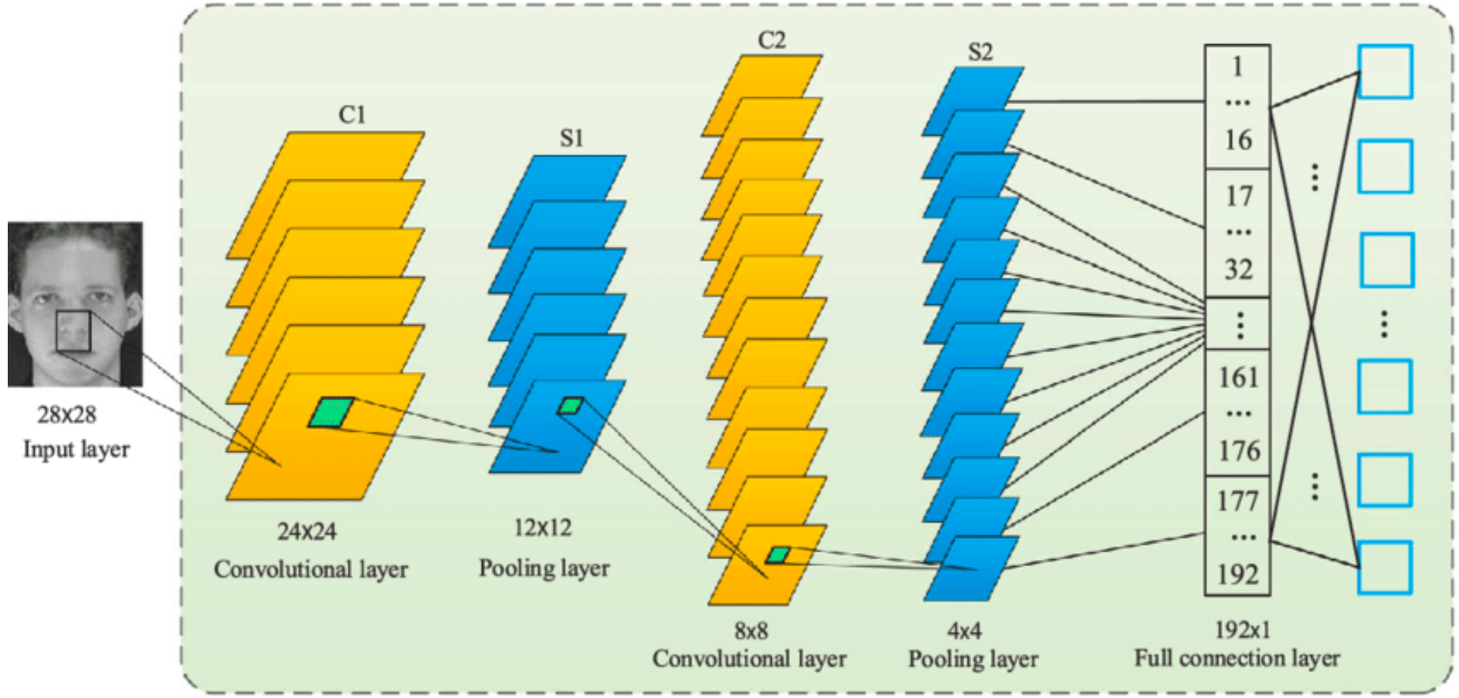
The final project: combination of the two projects

```
# Loop over all detections and draw detection box if confidence is above minimum threshold
for i in range(len(scores)):
    if (scores[i] > min_conf_threshold) and (scores[i] <= 1.0):

        # Get bounding box coordinates and draw box
        # Interpreter can return coordinates that are outside of image dimensions, need to force them to be within image using max() and min()
        ymin = int(max(1, (boxes[i][0] * imH)))
        xmin = int(max(1, (boxes[i][1] * imW)))
        ymax = int(min(imH, (boxes[i][2] * imH)))
        xmax = int(min(imW, (boxes[i][3] * imW)))

        ymin = int(max(ymin - (ymax-ymin), 0))

        face_rec_start = time.time()
        border = 10
        encodings = face_recognition.face_encodings(frame, [(ymin+border, xmax-border, ymax-border, xmin+border)], num_jitters=1, model='small')
        face_rec_encodings += time.time() - face_rec_start
        matches = face_recognition.compare_faces(encoding_data["encodings"], encodings[0])
        name = "Unkown"
        if True in matches:
            matchedIdxs = [i for (i, b) in enumerate(matches) if b]
            counts = {}
            for j in matchedIdxs:
                name = encoding_data["names"][j]
                counts[name] = counts.get(name, 0) + 1
                name = max(counts, key=counts.get)

                if currentname != name:
                    currentname = name
                    print(currentname)

        object_name = labels[int(classes[i])]
        if object_name == "bare":
            description = f"{name} has no mask!"
        else:
            description = f"{name} has mask !"
        print(name, "detected !")
        face_rec_end = time.time()
        face_rec_time += face_rec_end - face_rec_start
```



Types of Algorithms

- Holistic Matching
- Model-Based
- CNN
- Hidden Markov model
- Viola-Jones Algorithm



Comparison of different CNN models

Performance on the COCO Dataset

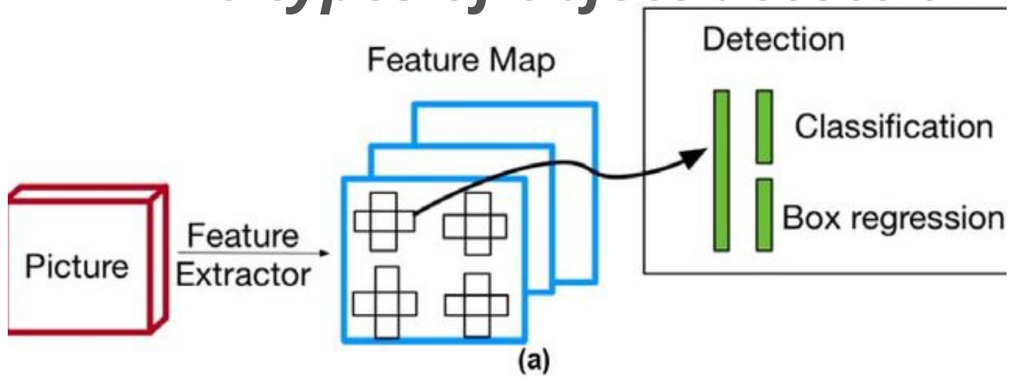
Model	Train	Test	mAP	FLOPS	FPS	Cfg	Weights
SSD300	COCO trainval	test-dev	41.2	-	46		link
SSD500	COCO trainval	test-dev	46.5	-	19		link
YOLOv2 608x608	COCO trainval	test-dev	48.1	62.94 Bn	40	cfg	weights
Tiny YOLO	COCO trainval	test-dev	23.7	5.41 Bn	244	cfg	weights
SSD321	COCO trainval	test-dev	45.4	-	16		link
DSSD321	COCO trainval	test-dev	46.1	-	12		link
R-FCN	COCO trainval	test-dev	51.9	-	12		link
SSD513	COCO trainval	test-dev	50.4	-	8		link
DSSD513	COCO trainval	test-dev	53.3	-	6		link
FPN FRCN	COCO trainval	test-dev	59.1	-	6		link
Retinanet-50-500	COCO trainval	test-dev	50.9	-	14		link
Retinanet-101-500	COCO trainval	test-dev	53.1	-	11		link
Retinanet-101-800	COCO trainval	test-dev	57.5	-	5		link
YOLOv3-320	COCO trainval	test-dev	51.5	38.97 Bn	45	cfg	weights
YOLOv3-416	COCO trainval	test-dev	55.3	65.86 Bn	35	cfg	weights
YOLOv3-608	COCO trainval	test-dev	57.9	140.69 Bn	20	cfg	weights
YOLOv3-tiny	COCO trainval	test-dev	33.1	5.56 Bn	220	cfg	weights
YOLOv3-spp	COCO trainval	test-dev	60.6	141.45 Bn	20	cfg	weights

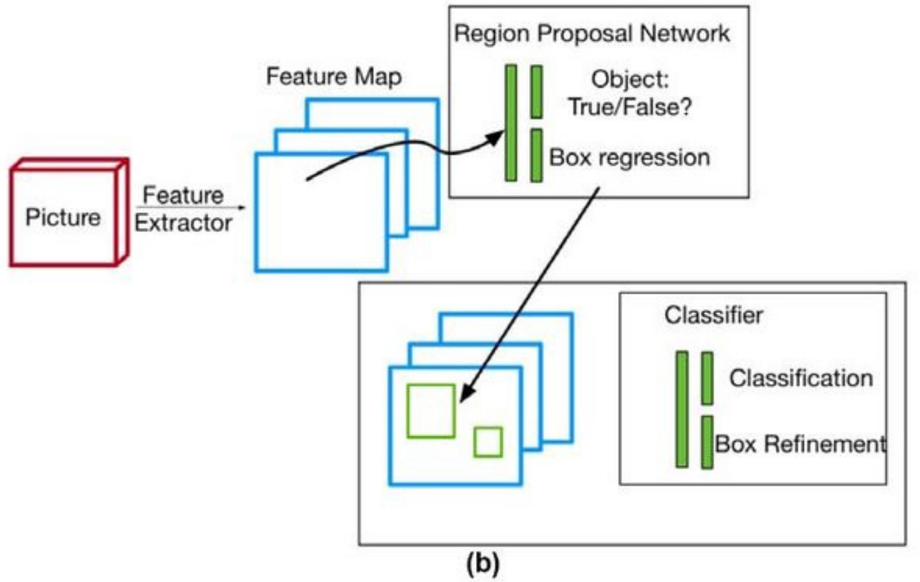
COCO - dataset, meaning “Common Objects In Context”, is a set of challenging, high quality datasets for computer vision, mostly state-of-the-art neural networks. This name is also used to name a format used by those datasets.

Our approach

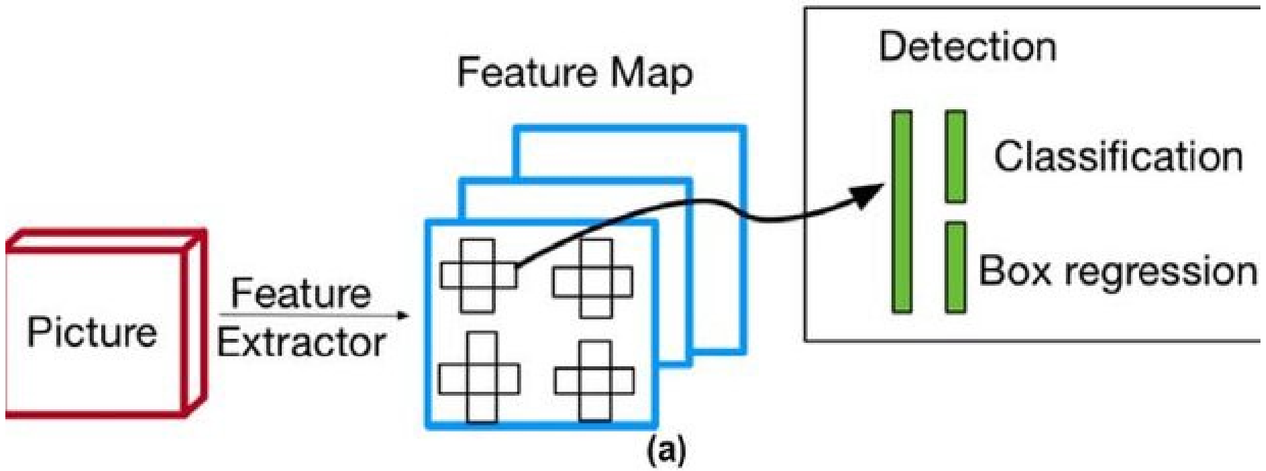
- ***CNN*** - Convolutional Neural Network. In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of artificial neural network (ANN), most commonly applied to analyze visual imagery.[1]. Wikipedia.

- ***Two types of object detection***



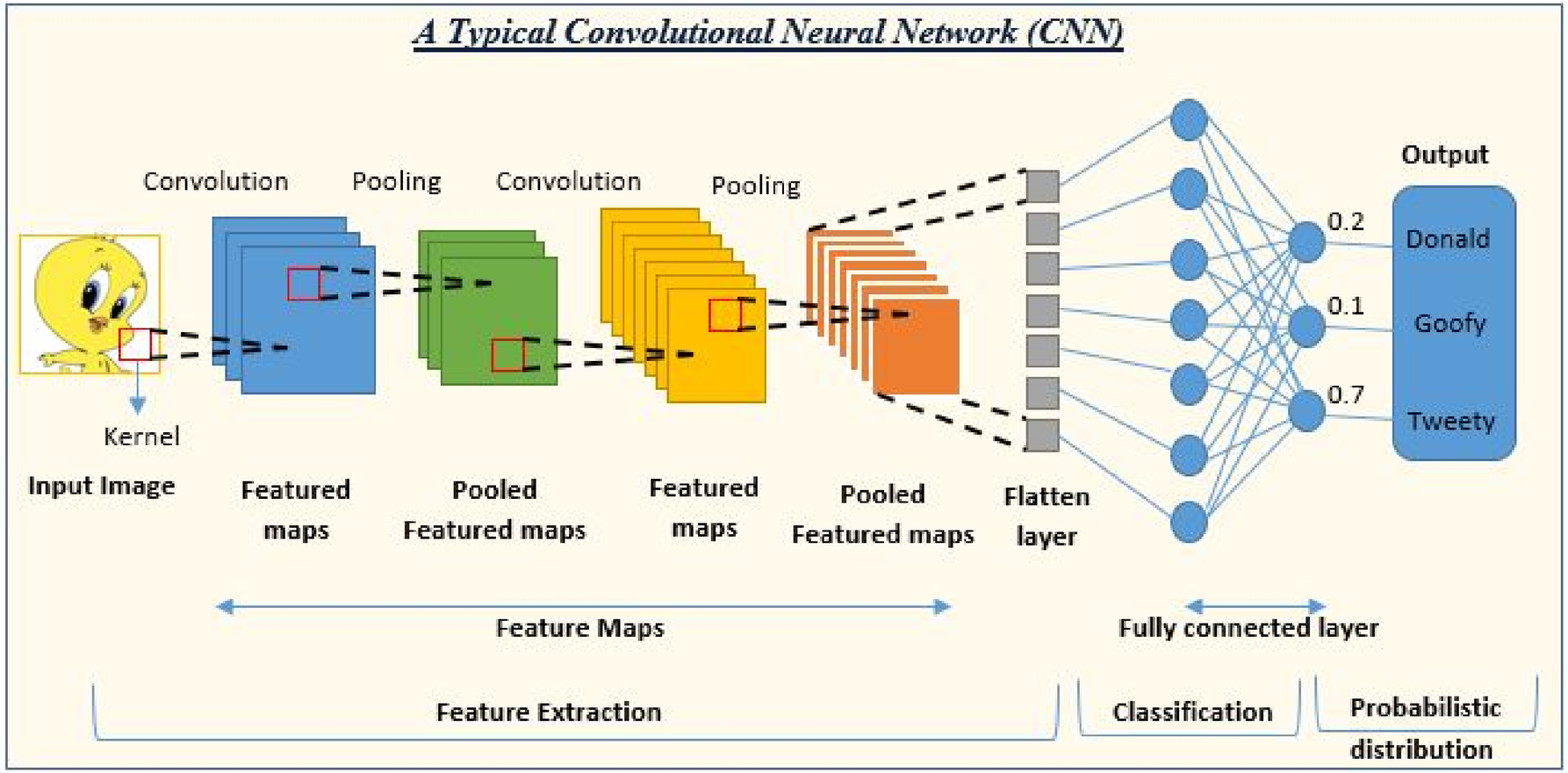


ssd_mobilenet_v2

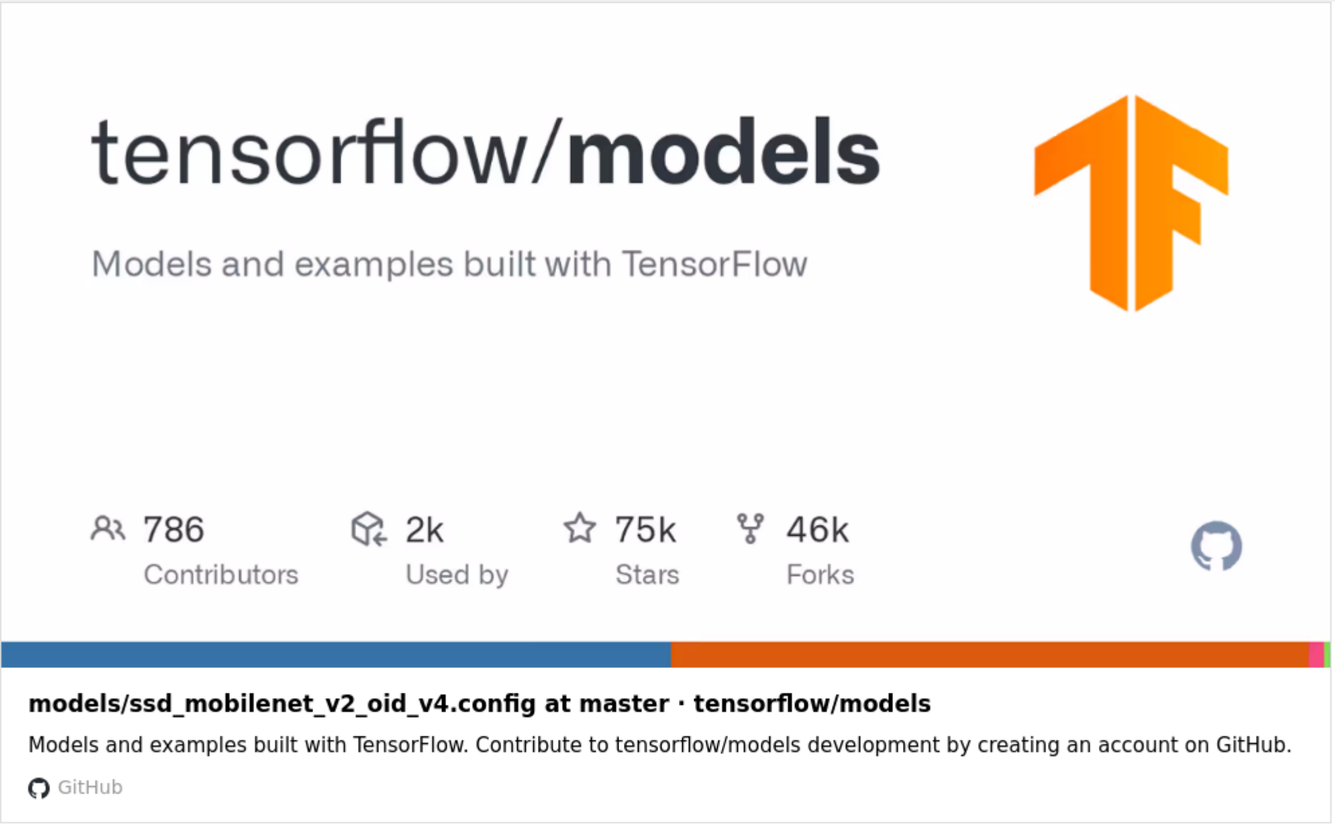


SSD Mobilenet V2 is a one-stage object detection model which has gained popularity for its lean network and novel depthwise separable convolutions. It is a model commonly deployed on low compute devices such as mobile (hence the name Mobilenet) with high accuracy performance.

ssd_mobilenet_v2



ssd_mobilenet_v2



Batch size = 24
Image resizing = 300x300
Kernel size = 1

Implementation of the model

```
127     interpreter = Interpreter(model_path=PATH_TO_CKPT)
128
129 interpreter.allocate_tensors()
130
131 # Get model details
132 input_details = interpreter.get_input_details()
133 output_details = interpreter.get_output_details()
134 height = input_details[0]["shape"][1]
135 width = input_details[0]["shape"][2]
136
137 floating_model = input_details[0]["dtype"] == np.float32
```

```
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
[{'name': 'normalized_input_image_tensor', 'index': 260, 'shape': array([  1, 300, 300,   3]
, dtype=int32), 'shape_signature': array([  1, 300, 300,   3], dtype=int32), 'dtype': <class
 'numpy.float32'>, 'quantization': (0.0, 0), 'quantization_parameters': {'scales': array([],
 dtype=float32), 'zero_points': array([], dtype=int32), 'quantized_dimension': 0}, 'sparsity
_parameters': {}}]
```

Conclusion


FPS: 0.90
bare: 83%
>>> import numpy
>>> numpy.__version__
'1.23.4'
>>> import cv2
>>>
(iot_env) pi@raspberrypi:~/mask-de