```
1 import cv2
2 import numpy as np
3 import pandas as pd
4 import matplotlib.pyplot as plt
5 import seaborn as sns
```

```
1 import  dlib
```

```
1 import cv2
2 import matplotlib.pyplot as plt
```

```
1 image = cv2.imread('/content/people1.jpg')
2
3
```

```
1   display (image.shape)
```

```
(1280, 1920, 3)
```

```
1   plt.imshow(image)
2   plt.show()
3
```


```
1   print(image)
```

```
[[[ 91 103 107]
  [102 114 118]
  [ 92 107 110]
  ...
  [187 192 193]
  [186 191 192]
  [185 190 191]]

 [[ 93 105 109]
  [ 97 109 113]
  [ 95 110 113]
  ...
  [187 192 193]
  [186 191 192]
  [185 190 191]]

 [[ 96 111 114]
  [ 96 111 114]
  [102 117 120]
  ...
  [186 191 192]
  [185 190 191]
  [184 189 190]]

 ...

 [[ 48  43  40]
  [ 46  41  38]
  [ 44  39  36]
  ...
  [251 248 244]
  [252 247 244]
  [251 246 243]]
```

```
       [[ 55  49  44]
        [ 53  47  42]
        [ 49  43  38]
        ...
        [251 248 244]
        [252 247 244]
        [252 247 244]]

       [[ 56  50  45]
        [ 54  48  43]
        [ 50  44  39]
        ...
        [252 249 245]
        [253 248 245]
        [253 248 245]]]
```

```
1 from google.colab.patches import cv2_imshow
```

```
1 from IPython.display import Image
2
```

```
1 import matplotlib.pyplot as plt
2 import matplotlib.image as mpimg
```

```
1
2 cv2.waitKey(10000)
3
```

```
    -1
```

```
1 cv2.destroyAllWindows()
```

```
1
2 image = cv2.resize(image, (800, 600))
3
4 image.shape
5
```

```
    (600, 800, 3)
```

```
1 plt.imshow(image)
2 plt.show()
```


```
1
```

```
1
2 cv2.waitKey(10000)
3
```

```
    -1
```

```
1 cv2.destroyAllWindows()
```

```
image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
plt.imshow(image_gray)
plt.show()
cv2.waitKey(10000)
cv2.destroyAllWindows()

```


```
display (image_gray.shape)
```

```
(600, 800)
```

```
face_detector = cv2.CascadeClassifier('/content/haarcascade_frontalface_default.xml')
display (face_detector)

```

```
detections = face_detector.detectMultiScale(image_gray)
display (detections)

```

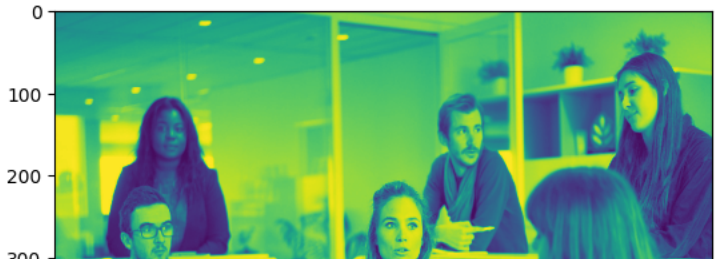
```
array([[677,  72,  68,  68],
       [115, 124,  53,  53],
       [475, 123,  59,  59],
       [387, 233,  73,  73],
       [ 92, 239,  66,  66],
       [390, 323,  56,  56]], dtype=int32)
```

```
display (len(detections))
```

```
6
```

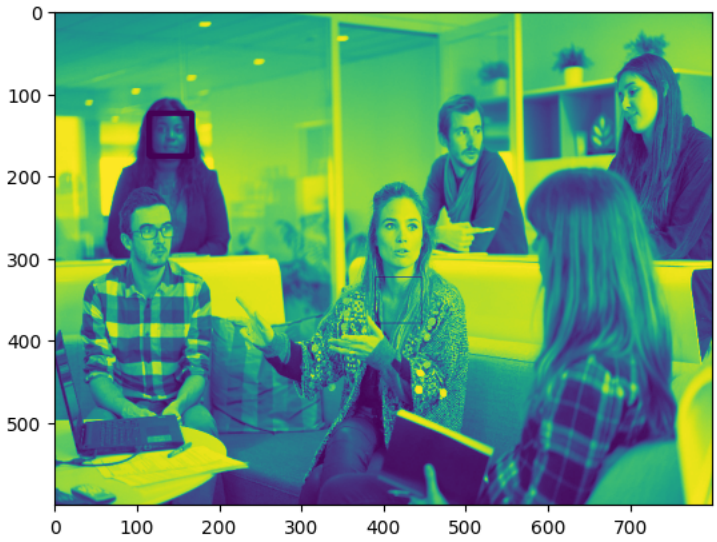
```
x=390  # X - Co ordinates
y=323  # Y- Co ordinates
w=56   # Face Width
h=56   # Face Height

```

```
cv2.rectangle(image_gray, (x, y), (x + w, y + h), (0,255,255), 1)
plt.imshow(image_gray)
plt.show()
cv2.waitKey(10000)
cv2.destroyAllWindows()
```



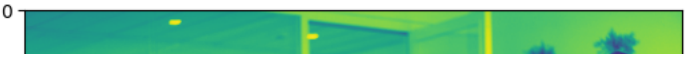

```
x=115  # X - Co ordinates
y=124  # Y- Co ordinates
w=52   # Face Width
h=52   # Face Height

```


```
cv2.rectangle(image_gray, (x, y), (x + w, y + h), (0,255,255), 5)
plt.imshow(image_gray)
plt.show()
cv2.waitKey(10000)
cv2.destroyAllWindows()

```

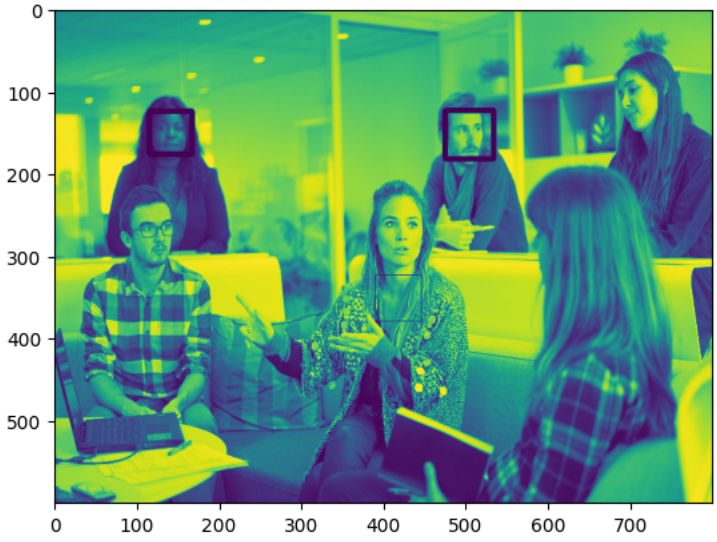



```
x=115  # X - Co ordinates
y=124  # Y- Co ordinates
w=52   # Face Width
h=52   # Face Height

```

```
cv2.rectangle(image_gray, (x, y), (x + w, y + h), (0,255,255), 5)
plt.imshow(image_gray)
plt.show()
cv2.waitKey(10000)
cv2.destroyAllWindows()

```



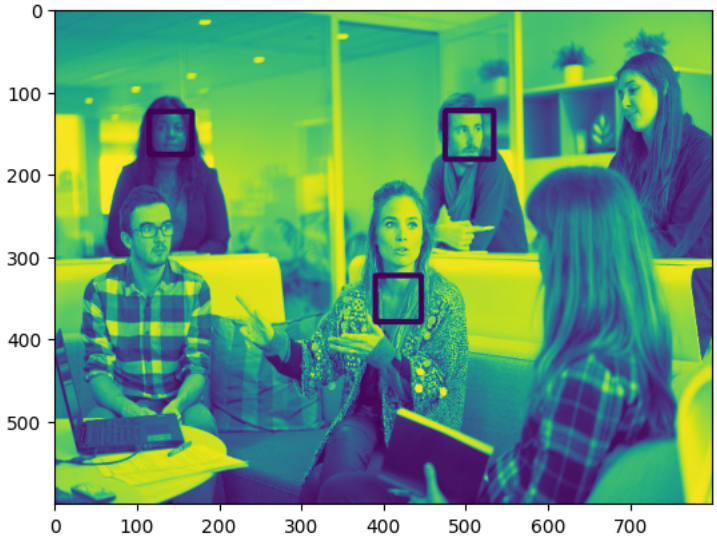
```
x=475  # X - Co ordinates
y=123  # Y- Co ordinates
w=59   # Face Width
h=59  # Face Height
cv2.rectangle(image_gray, (x, y), (x + w, y + h), (0,255,255), 5)
plt.imshow(image_gray)
plt.show()
cv2.waitKey(10000)
cv2.destroyAllWindows()

```




```

for (x, y, w, h) in detections:
  print (x, y, w, h)

```

```
677 72 68 68
115 124 53 53
475 123 59 59
387 233 73 73
92 239 66 66
390 323 56 56
```

```
cv2.rectangle(image_gray, (x, y), (x + w, y + h), (0,255,255), 5)
plt.imshow(image_gray)
plt.show()
cv2.waitKey(10000)
plt.show()
cv2.destroyAllWindows()

```

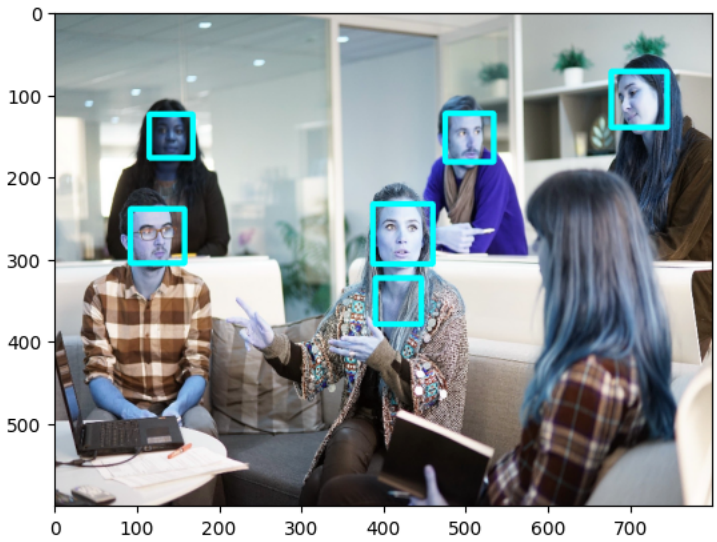


```
image = cv2.imread('/content/people1.jpg')
display (image.shape)
image = cv2.resize(image, (800, 600)) # Resize image
display (image.shape)
detections = face_detector.detectMultiScale(image)
display (detections)

```

```
(1280, 1920, 3)
(600, 800, 3)
array([[677,  72,  68,  68],
       [115, 124,  53,  53],
       [475, 123,  59,  59],
       [387, 233,  73,  73],
       [ 92, 239,  66,  66],
       [390, 323,  56,  56]], dtype=int32)
```

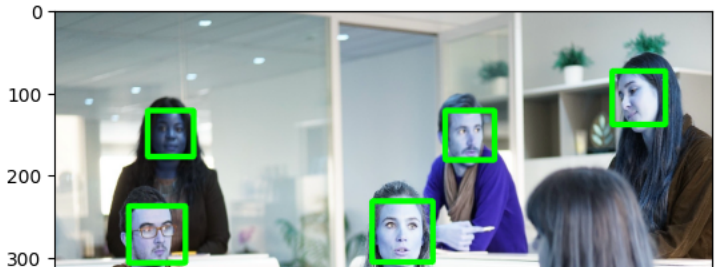
```
for (x, y, w, h) in detections:
  print (x, y, w, h)
  cv2.rectangle(image, (x, y), (x + w, y + h), (0,255,255), 5)
plt.imshow(image)
plt.show()
cv2.waitKey(10000)
plt.show()
cv2.destroyAllWindows()

```

```
677 72 68 68
115 124 53 53
475 123 59 59
387 233 73 73
92 239 66 66
390 323 56 56
```

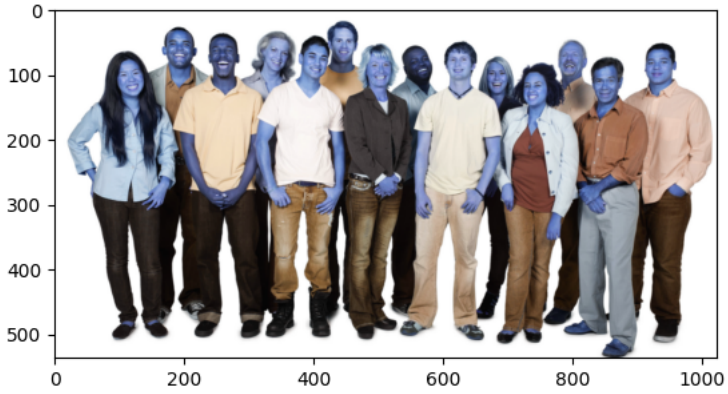



```
image = cv2.imread('/content/people1.jpg')
image = cv2.resize(image, (800, 600)) # Resize image
image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
detections = face_detector.detectMultiScale(image_gray, scaleFactor = 1.09)
display (detections)
for (x, y, w, h) in detections:
    cv2.rectangle(image, (x, y), (x + w, y + h), (0,255,0), 5)
plt.imshow(image)
plt.show()
cv2.waitKey(10000)
plt.show()
cv2.destroyAllWindows()

```

```
array([[678,  74,  65,  65],
       [113, 122,  56,  56],
       [475, 122,  60,  60],
       [386, 232,  74,  74],
       [ 90, 238,  69,  69]], dtype=int32)
```




```
image = cv2.imread('/content/people2.jpg')
plt.imshow(image)
plt.show()
cv2.waitKey(10000)
plt.show()
cv2.destroyAllWindows()

```

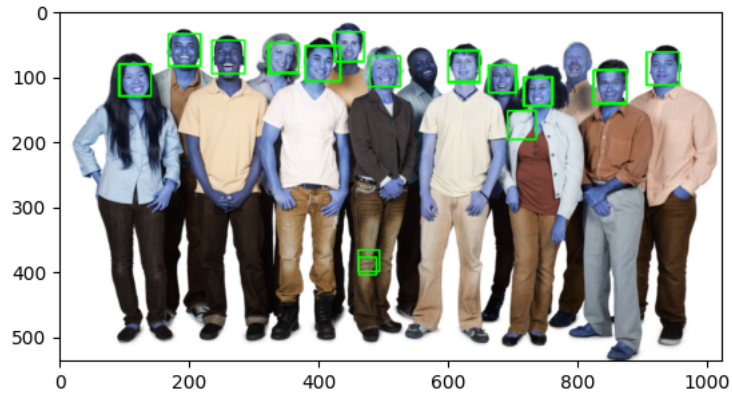



```
image = cv2.imread('/content/people2.jpg')
image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
detections = face_detector.detectMultiScale(image_gray, scaleFactor = 1.09)
display (detections)
for (x, y, w, h) in detections:
    print(w, h)
    cv2.rectangle(image, (x, y), (x + w, y + h), (0,255,0), 2)
plt.imshow(image)
plt.show()
cv2.waitKey(10000)
plt.show()
cv2.destroyAllWindows()

```

```
array([[424,  32,  46,  46],
       [324,  49,  45,  45],
       [663,  83,  43,  43],
       [600,  61,  49,  49],
       [907,  63,  50,  50],
       [718, 101,  44,  44],
       [235,  45,  51,  51],
       [380,  53,  54,  54],
       [ 91,  81,  50,  50],
       [168,  35,  50,  50],
       [825,  90,  51,  51],
       [479,  69,  47,  47],
       [692, 153,  44,  44],
       [461, 367,  33,  33],
       [462, 378,  27,  27]], dtype=int32)
46 46
45 45
43 43
49 49
50 50
44 44
```

```
image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
detections = face_detector.detectMultiScale(image_gray, scaleFactor = 1.2,minNeighbors=7)
display (detections)
for (x, y, w, h) in detections:
    print(w, h)
    cv2.rectangle(image, (x, y), (x + w, y + h), (0,255,0), 2)
plt.imshow(image)
plt.show()
cv2.waitKey(10000)
plt.show()
cv2.destroyAllWindows()

```

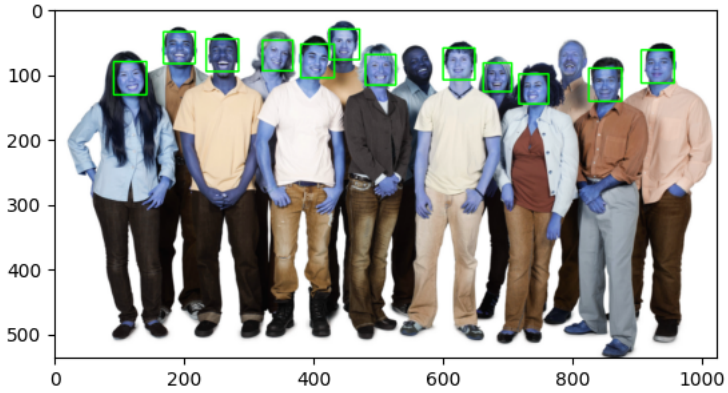
```
array([[425,  32,  45,  45],
       [322,  49,  47,  47],
       [716, 102,  44,  44],
       [600,  60,  48,  48],
       [168,  35,  49,  49],
       [662,  83,  42,  42],
       [ 93,  82,  48,  48],
       [379,  54,  54,  54],
       [906,  63,  51,  51],
       [824,  91,  52,  52]], dtype=int32)
45 45
47 47
44 44
48 48
49 49
42 42
48 48
54 54
51 51
52 52
```




```
image = cv2.imread('/content/people2.jpg')

image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
detections = face_detector.detectMultiScale(image_gray, scaleFactor = 1.2,minNeighbors=7)
display (detections)
for (x, y, w, h) in detections:
    print(w, h)
    cv2.rectangle(image, (x, y), (x + w, y + h), (0,255,0), 2)
plt.imshow(image)
plt.show()
cv2.waitKey(10000)
```

```
12 plt.show()
```

```
array([[424,  31,  47,  47],
       [321,  48,  47,  47],
       [600,  60,  49,  49],
       [168,  35,  49,  49],
       [380,  54,  52,  52],
       [906,  63,  51,  51],
       [717, 100,  46,  46],
       [662,  83,  44,  44],
       [ 91,  81,  51,  51],
       [234,  46,  50,  50],
       [825,  91,  51,  51],
       [480,  70,  47,  47]], dtype=int32)
47 47
47 47
49 49
49 49
52 52
51 51
46 46
44 44
51 51
50 50
51 51
47 47
```




```
image = cv2.imread('/content/people2.jpg')
image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
detections = face_detector.detectMultiScale(image_gray, scaleFactor = 1.2,
                                            minNeighbors=7,minSize=(20,20), maxSize=(100,100))
display (detections)
for (x, y, w, h) in detections:
    print(w, h)
    cv2.rectangle(image, (x, y), (x + w, y + h), (0,255,0), 2)
plt.imshow(image)
plt.show()
cv2.waitKey(10000)
plt.show()
cv2.destroyAllWindows()

```

```
array([[424,  31,  47,  47],
       [321,  48,  47,  47],
       [600,  60,  49,  49],
       [168,  35,  49,  49],
       [380,  54,  52,  52],
       [906,  63,  51,  51],
       [717, 100,  46,  46],
       [662,  83,  44,  44],
       [ 91,  81,  51,  51],
       [234,  46,  50,  50],
       [825,  91,  51,  51],
       [480,  70,  47,  47]], dtype=int32)
47 47
47 47
49 49
49 49
```

```
eye_detector= cv2.CascadeClassifier('/content/haarcascade_eye.xml')
display (eye_detector)

```

```
< cv2.CascadeClassifier 0x7e69554627d0>
```

```
image = cv2.imread('/content/people1.jpg')
display (image.shape)

image = cv2.resize(image, (1600,1000)) # Resize image
print(image.shape)
face_detections = face_detector.detectMultiScale(image, scaleFactor = 1.3, minSize = (30,30))
for (x, y, w, h) in face_detections:
  cv2.rectangle(image, (x, y), (x + w, y + h), (0,255,0), 2)

eye_detections = eye_detector.detectMultiScale(image, scaleFactor = 1.1, minNeighbors=10, maxSize=(60,60))

for (x, y, w, h) in eye_detections:
  print(w, h)
  cv2.rectangle(image, (x, y), (x + w, y + h), (0,0,255), 2)
plt.imshow(image)
plt.show()
cv2.waitKey(10000)
plt.show()
cv2.destroyAllWindows()

```

```
(1280, 1920, 3)
(1000, 1600, 3)
33 33
30 30
23 23
22 22
29 29
33 33
31 31
25 25
22 22
```

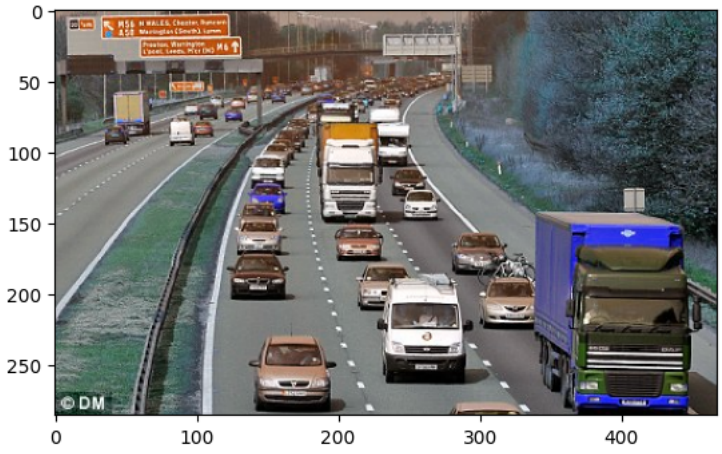

```
car_detector = cv2.CascadeClassifier('/content/cars.xml')

```

```
display (car_detector)

```

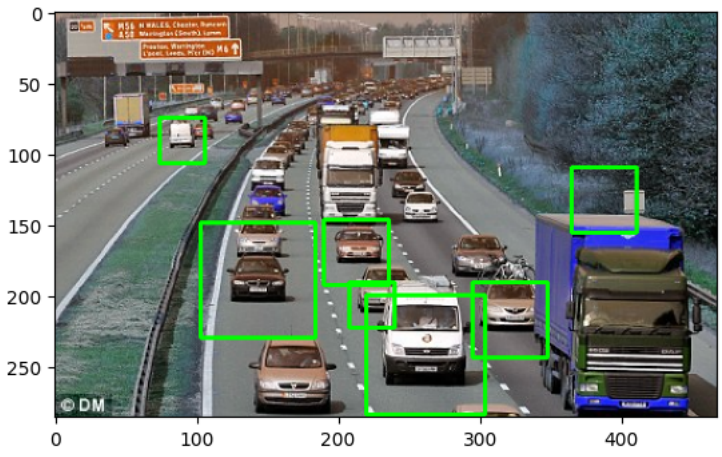
```
< cv2.CascadeClassifier 0x7e6955304fb0>
```

```
image = cv2.imread('/content/car.jpg')
display (image.shape)
plt.imshow(image)
plt.show()
cv2.waitKey(10000)
plt.show()
cv2.destroyAllWindows()

```

```
(286, 468, 3)
```




```
image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
detections = car_detector.detectMultiScale(image_gray, scaleFactor = 1.03, minNeighbors=8)
for (x, y, w, h) in detections:
    cv2.rectangle(image, (x, y), (x + w, y + h), (0,255,0), 2)
plt.imshow(image)
plt.show()
cv2.waitKey(10000)
plt.show()
cv2.destroyAllWindows()

```



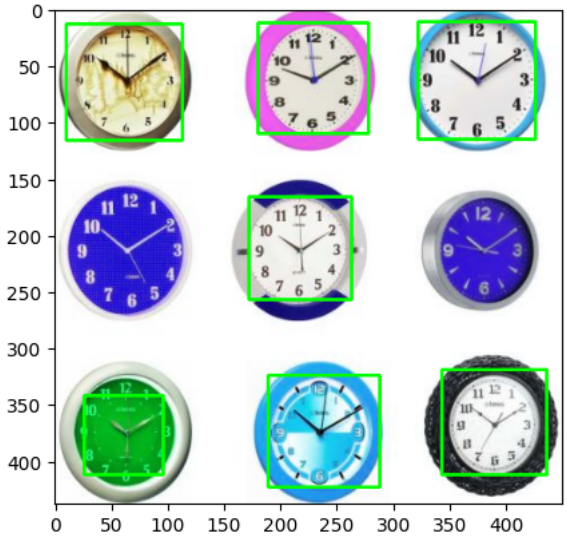

```
clock_detector = cv2.CascadeClassifier('/content/clocks.xml')
display (clock_detector)

```

```
< cv2.CascadeClassifier 0x7e6955376b90>
```

```
image = cv2.imread('/content/clock.jpg')
display (image.shape)
plt.imshow(image)
plt.show()
cv2.waitKey(10000)
plt.show()
cv2.destroyAllWindows()

```

```
(438, 450, 3)
```

```
image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
detections = clock_detector.detectMultiScale(image_gray, scaleFactor = 1.03, minNeighbors=1)
for (x, y, w, h) in detections:
    cv2.rectangle(image, (x, y), (x + w, y + h), (0,255,0), 2)
plt.imshow(image)
plt.show()
cv2.waitKey(10000)
plt.show()
cv2.destroyAllWindows()

```

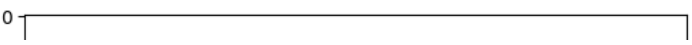


```
full_detector  = cv2.CascadeClassifier('/content/fullbody.xml')
display (full_detector)

```

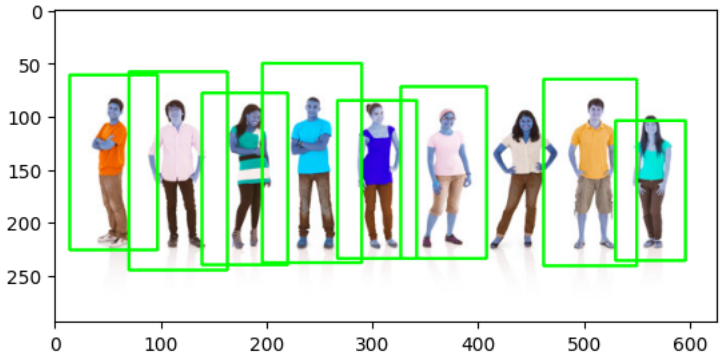
```
< cv2.CascadeClassifier 0x7e6955374f70>
```

```
image = cv2.imread('/content/people3.jpg')
display (image.shape)
plt.imshow(image)
plt.show()
cv2.waitKey(10000)
plt.show()
cv2.destroyAllWindows()

```

```
(294, 626, 3)
```



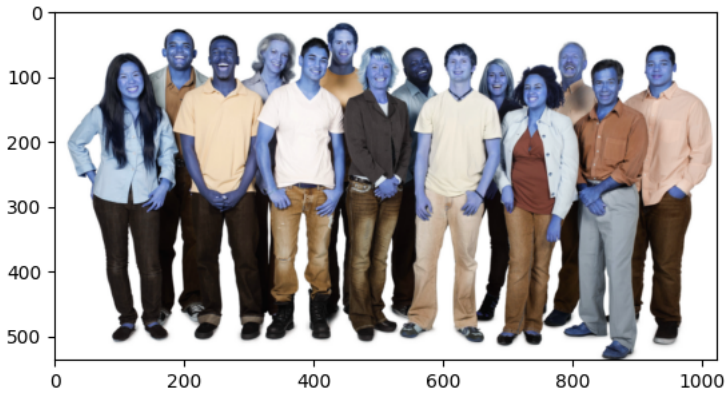
```
image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
detections = full_detector.detectMultiScale(image_gray, scaleFactor = 1.05, minNeighbors=5,    minSize = (50,50))
for (x, y, w, h) in detections:
    cv2.rectangle(image, (x, y), (x + w, y + h), (0,255,0), 2)
plt.imshow(image)
plt.show()
cv2.waitKey(10000)
plt.show()
cv2.destroyAllWindows()

```



Face detection with HOG and Dlib

```
import dlib
```

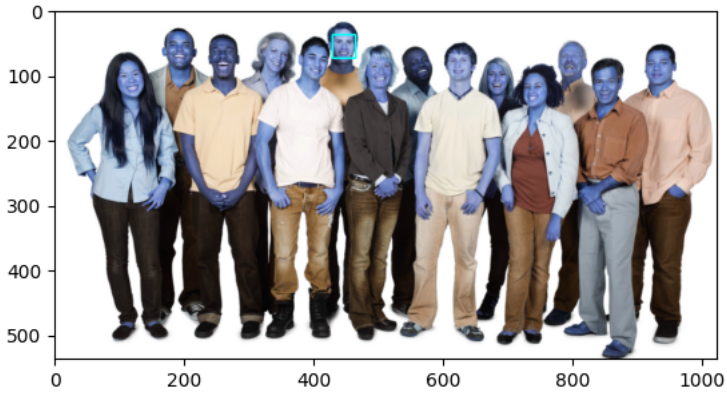
```
image = cv2.imread('/content/people2.jpg')
plt.imshow(image)
plt.show()
cv2.waitKey(10000)
plt.show()
cv2.destroyAllWindows()

```



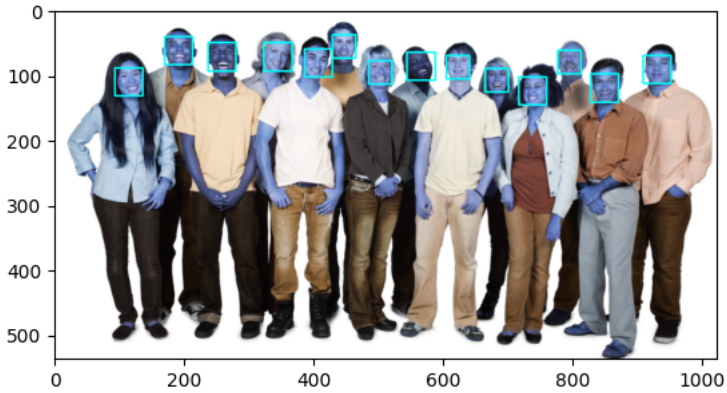
```
face_detector_hog = dlib.get_frontal_face_detector()
detections = face_detector_hog(image, 1) # 1 - is the scale factor

```

```
print(detections[0])
print(detections[0].left())
print(detections[0].top())
print(detections[0].right())
print(detections[0].bottom())
cv2.rectangle(image, (detections[0].left(), detections[0].top()), (detections[0].right(), detections[0].bottom()), (0, 255, 255), 2)
plt.imshow(image)
plt.show()
cv2.waitKey(10000)
plt.show()
cv2.destroyAllWindows()

```

```
[(429, 38) (465, 74)]
429
38
465
74
```

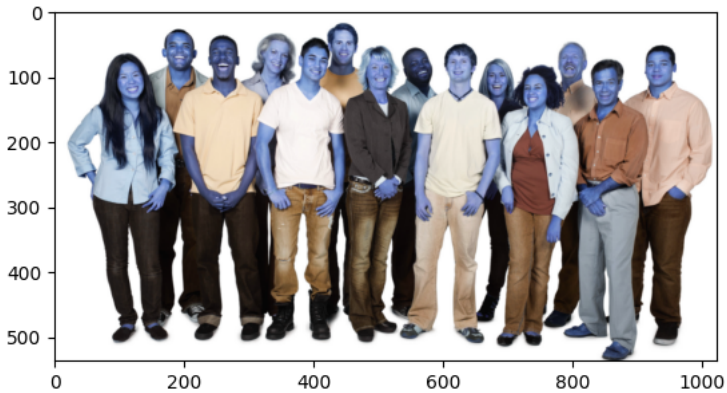



```
for face in detections:
  l, t, r, b = face.left(), face.top(), face.right(), face.bottom()
  cv2.rectangle(image, (l, t), (r, b), (0, 255, 255), 2)
plt.imshow(image)
plt.show()
cv2.waitKey(10000)
plt.show()
cv2.destroyAllWindows()

```



Face detection with CNN and Dlib

```
image = cv2.imread('/content/people2.jpg')
plt.imshow(image)
plt.show()
cv2.waitKey(10000)
plt.show()
cv2.destroyAllWindows()

```




```
pip install tensorflow

```

```
Requirement already satisfied: tensorflow in /usr/local/lib/python3.10/dist-packages (2.13.0)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.4.0)
```

```
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=23.1.21 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (23.5.26)
Requirement already satisfied: gast<=0.4.0,>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.4.0)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.59.0)
Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.9.0)
Requirement already satisfied: keras<2.14,>=2.13.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.13.1)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (16.0.6)
Requirement already satisfied: numpy<=1.24.3,>=1.22 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.23.5)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.3.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from tensorflow) (23.2)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in /usr/local/lib/py
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from tensorflow) (67.7.2)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.16.0)
Requirement already satisfied: tensorboard<2.14,>=2.13 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.13.0)
Requirement already satisfied: tensorflow-estimator<2.14,>=2.13.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.13
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.3.0)
Requirement already satisfied: typing-extensions<4.6.0,>=3.6.6 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (4.5.0)
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.15.0)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.10/dist-packages (from astunparse>=1.6.0->tensorflow) (0
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.14,>=2.13->tenso
Requirement already satisfied: google-auth-oauthlib<1.1,>=0.5 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.14,>=2
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.14,>=2.13->tensorflow
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.14,>=2.13->tensorf
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.14,>=2.13->tensorflow
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensor
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorb
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboard<2.1
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from google-auth-oauthlib<1.1,>=
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensor
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<2.14,>
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.10/dist-packages (from werkzeug>=1.0.1->tensorboard<2.14,
Requirement already satisfied: pyasn1<0.6.0,>=0.4.6 in /usr/local/lib/python3.10/dist-packages (from pyasn1-modules>=0.2.1->google-a
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from requests-oauthlib>=0.7.0->google-aut
```

```
pip install tensorflow-gpu

```

```
import tensorflow as tf
print(tf.__version__)

```

```
2.13.0
```

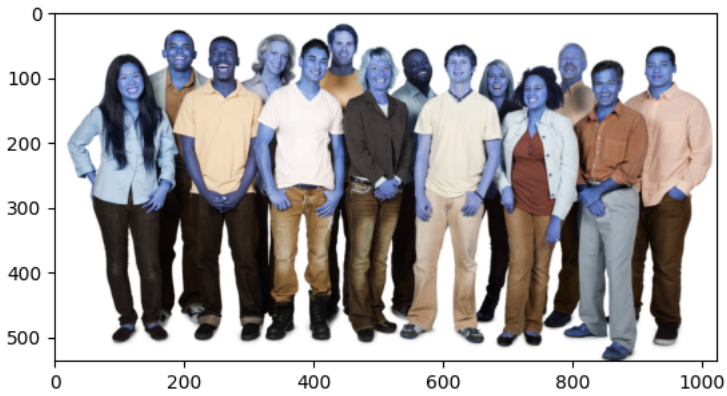
```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers

# Create a sequential model
model = keras.Sequential()

# Add convolutional layers
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))

# Flatten the output
model.add(layers.Flatten())

# Add dense layers
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))

# Compile the model
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Summary of the model architecture
model.summary()

```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 32)        320
```

```
 max_pooling2d (MaxPooling2  (None, 13, 13, 32)        0
 D)

 conv2d_1 (Conv2D)           (None, 11, 11, 64)        18496

 max_pooling2d_1 (MaxPoolin  (None, 5, 5, 64)          0
 g2D)

 flatten (Flatten)           (None, 1600)              0

 dense (Dense)               (None, 64)                102464

 dense_1 (Dense)             (None, 10)                650

=================================================================
Total params: 121930 (476.29 KB)
Trainable params: 121930 (476.29 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
import os
os.environ['CUDA_HOME'] = '/usr/local/cuda-X.Y'  # Update with your CUDA version

```

```

  image = cv2.imread('/content/people2.jpg')
  plt.imshow(image)
  plt.show()
  cv2.waitKey(10000)
  plt.show()
  cv2.destroyAllWindows()

```



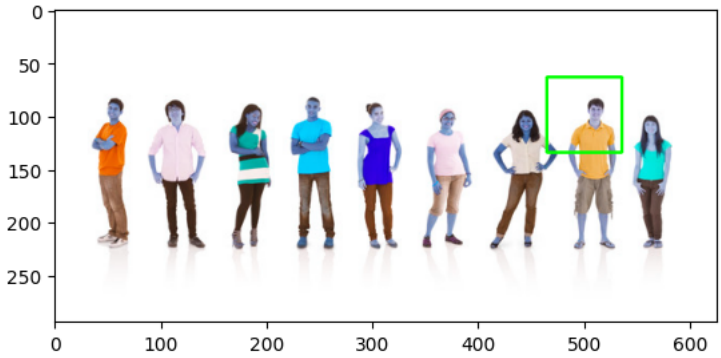

```
cnn_detector = dlib.cnn_face_detection_model_v1('/content/mmod_human_face_detector.dat')
display (cnn_detector)

```

```
<_dlib_pybind11.cnn_face_detection_model_v1 at 0x7e68d06ea1b0>
```

```
  detections = cnn_detector(image, 1)
  for face in detections:
    l, t, r, b, c = face.rect.left(), face.rect.top(), face.rect.right(), face.rect.bottom(), face.confidence
    print(c)
    cv2.rectangle(image, (l, t), (r, b), (255, 255, 0), 2)
  plt.imshow(image)
  plt.show()
  cv2.waitKey(10000)
  plt.show()
  cv2.destroyAllWindows()

```

```
1.1118313074111938
1.1012212038040161
1.0860981941223145
1.0829256772994995
1.072746992111206
1.067101001739502
1.062952995300293
1.0371497869491577
1.0303109884262085
1.0288389921188354
1.0205779075622559
0.8558180332183838
0.8077936172485352
```

```
image = cv2.imread('/content/people3.jpg')
image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
haarcascade_detector = cv2.CascadeClassifier('/content/haarcascade_frontalface_default.xml')
detections = haarcascade_detector.detectMultiScale(image_gray, scaleFactor = 1.001, minNeighbors=5, minSize = (5,5))
for (x, y, w, h) in detections:
  cv2.rectangle(image, (x, y), (x + w, y + h), (0,255,0), 2)
plt.imshow(image)
plt.show()
cv2.waitKey(10000)
plt.show()
cv2.destroyAllWindows()

```



```
image = cv2.imread('/content/people3.jpg')
face_detector_hog = dlib.get_frontal_face_detector()
detections = face_detector_hog(image, 4)
for face in detections:
    l, t, r, b = (face.left(), face.top(), face.right(), face.bottom())
    cv2.rectangle(image, (l, t), (r, b), (0, 255, 255), 2)
plt.imshow(image)
plt.show()
cv2.waitKey(10000)
plt.show()
cv2.destroyAllWindows()

```