"""# **Assignment for Computer vision**: **Face Detection With Deep Learning**

Face detection is a computer vision problem that involves finding faces in photos. Face detection can be performed using the classical feature-based cascade classifier using the OpenCV library. State-of-the-art face detection can be achieved using a Multi-task Cascade CNN via the MTCNN library.

Face detection is a problem in computer vision of locating and localizing one or more faces in a photograph. Locating a face in a photograph refers to finding the coordinate of the face in the image, whereas localization refers to demarcating the extent of the face, often via a bounding box around the face.

A general statement of the problem can be defined as follows: Given a still or video image, detect and localize an unknown number (if any) of faces. Detecting faces in a photograph is easily solved by humans, although has historically been challenging for computers given the dynamic nature of faces. For example, faces must be detected regardless of orientation or angle they are facing, light levels, clothing, accessories, hair color, facial hair, makeup, age, and so on.

The human face is a dynamic object and has a high degree of variability in its appearance, which makes face detection a difficult problem in computer vision. Given a photograph, a face detection system will output zero or more bounding boxes that contain faces. Detected faces can then be provided as input to a subsequent system, such as a face recognition system.

Face detection is a necessary first-step in face recognition systems, with the purpose of localizing and extracting the face region from the background.

A number of deep learning methods have been developed and demonstrated for face detection. One of the more popular approaches is called the “Multi-Task Cascaded Convolutional Neural Network,” or MTCNN for short, described by Kaipeng Zhang, et al. in the 2016 paper titled “Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks.”

https://arxiv.org/ftp/arxiv/papers/1604/1604.02878.pdf

The MTCNN is popular because it achieved then state-of-the-art results on a range of benchmark datasets, and because it is capable of also recognizing other facial features such as eyes and mouth, called landmark detection.

The network uses a cascade structure with three networks; first the image is rescaled to a range of different sizes (called an image pyramid), then the first model (Proposal Network or P-Net) proposes candidate facial regions, the second model (Refine Network or R-Net) filters the bounding boxes, and the third model (Output Network or O-Net) proposes facial landmarks.

The library can be installed via pip

"""

!pip install mtcnn

!pip show mtcnn

confirm mtcnn was installed correctly

import mtcnn

print version

print(mtcnn.__version__)

"""Download the images from the following link and place it in your current working directory with the filename test2.jpg & test1.jpg

https://machinelearningmastery.com/wp-content/uploads/2019/03/test2.jpg

https://machinelearningmastery.com/wp-content/uploads/2019/03/test1.jpg

"""

face detection with mtcnn on a photograph

from matplotlib import pyplot

from mtcnn.mtcnn import MTCNN

```
# load image from file
filename = 'test1.jpg'
pixels = pyplot.imread(filename)
# create the detector, using default weights
detector = MTCNN()
# detect faces in the image
faces = detector.detect_faces(pixels)
for face in faces:
	print(face)

# draw an image with detected objects
def draw_image_with_boxes(filename, result_list):
	# load the image
	data = pyplot.imread(filename)
	# plot the image
	pyplot.imshow(data)
	# get the context for drawing boxes
	ax = pyplot.gca()
	# plot each box
	for result in result_list:
		# get coordinates
		x, y, width, height = result['box']
		# create the shape
		rect = Rectangle((x, y), width, height, fill=False, color='red')
		# draw the box
		ax.add_patch(rect)
	# show the plot
	pyplot.show()

# face detection with mtcnn on a photograph
from matplotlib import pyplot
```

```
from matplotlib.patches import Rectangle
from mtcnn.mtcnn import MTCNN

# draw an image with detected objects
def draw_image_with_boxes(filename, result_list):
	# load the image
	data = pyplot.imread(filename)
	# plot the image
	pyplot.imshow(data)
	# get the context for drawing boxes
	ax = pyplot.gca()
	# plot each box
	for result in result_list:
		# get coordinates
		x, y, width, height = result['box']
		# create the shape
		rect = Rectangle((x, y), width, height, fill=False, color='red')
		# draw the box
		ax.add_patch(rect)
	# show the plot
	pyplot.show()

filename = 'test1.jpg'
# load image from file
pixels = pyplot.imread(filename)
# create the detector, using default weights
detector = MTCNN()
# detect faces in the image
faces = detector.detect_faces(pixels)
# display faces on the original image
draw_image_with_boxes(filename, faces)
```

```
"""You can draw a circle via the Circle class for the eyes, nose, and mouth."""

# face detection with mtcnn on a photograph
from matplotlib import pyplot
from matplotlib.patches import Rectangle
from matplotlib.patches import Circle
from mtcnn.mtcnn import MTCNN

# draw an image with detected objects
def draw_image_with_boxes(filename, result_list):
	# load the image
	data = pyplot.imread(filename)
	# plot the image
	pyplot.imshow(data)
	# get the context for drawing boxes
	ax = pyplot.gca()
	# plot each box
	for result in result_list:
		# get coordinates
		x, y, width, height = result['box']
		# create the shape
		rect = Rectangle((x, y), width, height, fill=False, color='red')
		# draw the box
		ax.add_patch(rect)
		# draw the dots
		for key, value in result['keypoints'].items():
			# create and draw dot
			dot = Circle(value, radius=2, color='red')
			ax.add_patch(dot)
	# show the plot
```

```
        pyplot.show()

filename = 'test1.jpg'
# load image from file
pixels = pyplot.imread(filename)
# create the detector, using default weights
detector = MTCNN()
# detect faces in the image
faces = detector.detect_faces(pixels)
# display faces on the original image
draw_image_with_boxes(filename, faces)

"""You can now try face detection on the test2.jpg photograph.

You may want to extract the detected faces and pass them as input to another system. This can be achieved by extracting the pixel data directly out of the photograph; We can demonstrate this by extracting each face and plotting them as separate subplots. You could just as easily save them to file. The draw_faces() below extracts and plots each detected face in a photograph.
"""

# extract and plot each detected face in a photograph
from matplotlib import pyplot
from matplotlib.patches import Rectangle
from matplotlib.patches import Circle
from mtcnn.mtcnn import MTCNN

# draw each face separately
def draw_faces(filename, result_list):
        # load the image
        data = pyplot.imread(filename)
        # plot each face as a subplot
        for i in range(len(result_list)):
```

```
            # get coordinates
            x1, y1, width, height = result_list[i]['box']
            x2, y2 = x1 + width, y1 + height
            # define subplot
            pyplot.subplot(1, len(result_list), i+1)
            pyplot.axis('off')
            # plot face
            pyplot.imshow(data[y1:y2, x1:x2])
    # show the plot
    pyplot.show()

filename = 'test2.jpg'
# load image from file
pixels = pyplot.imread(filename)
# create the detector, using default weights
detector = MTCNN()
# detect faces in the image
faces = detector.detect_faces(pixels)
# display faces on the original image
draw_faces(filename, faces)

"""Assignment 1: Detect faces from live images taken from webcam.

Assignment 2: Once you're done with images, then take short live video sequence and detect faces.

Assignment 3: Detect faces in mask wearing images.
"""
```