# Preset up installations to run the Face detection in image

**🡪!apt update**

**🡪!apt install -y libgtk-3-dev libgl1-mesa-glx libglu1-mesa libjpeg-dev libpng-dev**

# installation of openvino-dev

**🡪!pip install openvino-dev**

**🡪!pip install nvidia-cublas-cu12==12.1.3.1 nvidia-cuda-cupti-cu12==12.1.105 nvidia-cuda-nvrtc-cu12==12.1.105 nvidia-cuda-runtime-cu12==12.1.105 nvidia-cudnn-cu12==8.9.2.26 nvidia-cufft-cu12==11.0.2.54 nvidia-curand-cu12==10.3.2.106 nvidia-cusolver-cu12==11.4.5.107 nvidia-cusparse-cu12==12.1.0.106 nvidia-nccl-cu12==2.20.5 nvidia-nvtx-cu12==12.1.105**

**🡪import openvino**

**from openvino.runtime import Core**

**# Initialize OpenVINO Runtime**

**core = Core()**

**# Print available devices**

**print("Available devices:", core.available\_devices)**

* **!pip install opencv-python-headless**
* **!pip install openvino-dev[tools]**
* **!omz\_downloader --name face-detection-adas-0001**

Main Program

🡪**import cv2**

**import numpy as np**

**import matplotlib.pyplot as plt**

**from openvino.runtime import Core**

**# Initialize OpenVINO Inference Engine**

**ie = Core()**

**# Load the face detection model**

**model\_xml = 'intel/face-detection-adas-0001/FP32/face-detection-adas-0001.xml'**

**model\_bin = 'intel/face-detection-adas-0001/FP32/face-detection-adas-0001.bin'**

**net = ie.read\_model(model=model\_xml, weights=model\_bin)**

**exec\_net = ie.compile\_model(model=net, device\_name='CPU')**

**# Load and preprocess the input image**

**image\_path = '/content/testimage2.jpg' # Provide the path to your image**

**image = cv2.imread(image\_path)**

**if image is None:**

**raise FileNotFoundError(f"Image file not found: {image\_path}")**

**# Preprocess image for the model**

**input\_blob = next(iter(exec\_net.inputs)) # Get the input node**

**input\_shape = input\_blob.shape # Access shape directly from the input node**

**n, c, h, w = input\_shape**

**input\_image = cv2.resize(image, (w, h))**

**input\_image = input\_image.transpose((2, 0, 1)) # Change data layout from HWC to CHW**

**input\_image = input\_image.reshape((n, c, h, w))**

**# Create InferRequest object**

**infer\_request = exec\_net.create\_infer\_request()**

**# Perform inference using the InferRequest object**

**outputs = infer\_request.infer(inputs={input\_blob.any\_name: input\_image})**

**# Process the output**

**output\_blob = next(iter(outputs))**

**output = outputs[output\_blob]**

**# Draw bounding boxes on detected faces**

**for detection in output[0][0]:**

**confidence = detection[2]**

**if confidence > 0.5: # Filter out weak detections**

**xmin = int(detection[3] \* image.shape[1])**

**ymin = int(detection[4] \* image.shape[0])**

**xmax = int(detection[5] \* image.shape[1])**

**ymax = int(detection[6] \* image.shape[0])**

**cv2.rectangle(image, (xmin, ymin), (xmax, ymax), (0, 255, 0), 2)**

**# Display the result using matplotlib**

**plt.imshow(cv2.cvtColor(image, cv2.COLOR\_BGR2RGB))**

**plt.axis('off')**

**plt.show()**

**Output:**

****

****