```python
import cv2
import sys
import os
from zipfile import ZipFile
from urllib.request import urlretrieve


# ========================-Downloading Assets-========================
def download_and_unzip(url, save_path):
    print(f"Downloading and extracting assests....", end="")

    # Downloading zip file using urllib package.
    urlretrieve(url, save_path)

    try:
        # Extracting zip file using the zipfile package.
        with ZipFile(save_path) as z:
            # Extract ZIP file contents in the same directory.
            z.extractall(os.path.split(save_path)[0])

        print("Done")

    except Exception as e:
        print("\nInvalid file.", e)


URL = r"https://www.dropbox.com/s/efitgt363ada95a/opencv_bootcamp_assets_12.zip?dl=1"

asset_zip_path = os.path.join(os.getcwd(), f"opencv_bootcamp_assets_12.zip")

# Download if assest ZIP does not exists.
if not os.path.exists(asset_zip_path):
    download_and_unzip(URL, asset_zip_path)
# ====================================================================


s = 0
if len(sys.argv) > 1:
    s = sys.argv[1]

source = cv2.VideoCapture(s)

windowName = "Camera Preview"
cv2.namedWindow(windowName, cv2.WINDOW_NORMAL)

# This Function Specifically Design to Read Caffe Model
# This Function Takes Two Arguments --> First is THe Network Architecture Information and Second is The Model File
net = cv2.dnn.readNetFromCaffe('../Face-Detection/deploy.prototxt',
                               '../Face-Detection/res10_300x300_ssd_iter_140000_fp16.caffemodel')

# Model Parameters
width = 300
height = 300
mean = [104, 117, 123]
conf_threshold = 0.7

while cv2.waitKey(1) != 27:
    has_frame, frame = source.read()
    if not has_frame:
        break

    frame = cv2.flip(frame, 1)
    frameHeight = frame.shape[0]
    frameWidth = frame.shape[1]

    # Crete a 4D blob from a Frame to Preprocessing Input Image in Proper Format
    blob = cv2.dnn.blobFromImage(frame, 1.0, (width, height), mean, swapRB=False, crop=False)

    # Run a Model
    net.setInput(blob)
    detections = net.forward()

    for i in range(detections.shape[2]):
        confidence = detections[0, 0, i, 2]
        if confidence > conf_threshold:
            x_left_bottom = int(detections[0, 0, i, 3] * frameWidth)
            y_left_bottom = int(detections[0, 0, i, 4] * frameHeight)
            x_right_top = int(detections[0, 0, i, 5] * frameWidth)
            y_right_top = int(detections[0, 0, i, 6] * frameHeight)

            cv2.rectangle(frame, (x_left_bottom, y_left_bottom), (x_right_top, y_right_top), (0, 255, 0))
            label = "Confidence: %.4f" % confidence
            label_size, baseline = cv2.getTextSize(label, cv2.FONT_HERSHEY_SIMPLEX, 0.5, 1)

            cv2.rectangle(frame(x_left_bottom, y_left_bottom - label_size[1]),
                          (x_left_bottom + label_size[0], y_left_bottom + baseline),
                          (255, 255, 255), cv2.FILLED)
            cv2.putText(frame, label, (x_left_bottom, y_left_bottom),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0))

    t, _ = net.getPerfProfile()
    label = "Inference time: %.2f ms" % (t * 1000.0 / cv2.getTickFrequency())
    cv2.putText(frame, label, (0, 15), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0))
    cv2.imshow(windowName, frame)
```