|  |  |
| --- | --- |
| 교육 제목 | **OpenCV DNN practice** |
| 교육 일시 | 2021년 12월 27 일 월요일 |
| 교육 장소 | 영우글로벌러닝 2층 |
| **교육 내용** | |
| 오전 | 1. OpenCV DNN(Deep Neural Network) 모듈    1. 미리 학습된 딥러닝 파일을 OpenCV DNN 모듈로 실행할 수 있다.    2. 순전파(foward), 추론(inference)만 가능하며 학습은 지원하지 않음 2. 네트워크 불러오기 - cv2.dnn.readNet    1. OpenCV로 딥러닝을 실행하기 위해서는 우선 cv2.dnn\_Net 클래스 객체를 생성해야 합니다.    2. 객체 생성에는 훈련된 가중치와 네트워크 구성을 저장하고 있는 파일이 필요합니다. 3. cv2.dnn.readNet(model, config=None, framework=None) -> retval    1. • model: 훈련된 가중치를 저장하고 있는 이진 파일 이름    2. • config: 네트워크 구성을 저장하고 있는 텍스트 파일 이름, config가 없는 경우도 많습니다.    3. • framework: 명시적인 딥러닝 프레임워크 이름    4. • retval: cv2.dnn\_Net 클래스 객체 4. 딥러닝 프레임워크    1. model 파일 확장자 config 파일 확장자 framework 문자열    2. 카페 \*.caffemodel \*.prototxt "caffe"    3. 텐서플로우 \*.pb \*.pbtxt "tensorflow"    4. 토치 \*.t7 또는 \*.net "torch"    5. 다크넷 \*.weights \*.cfg "darknet"    6. DLDT \*.bin \*.xml "dldt"    7. ONNX \*.onnx "onnx" 5. 2. 네트워크 입력 블롭(blob) 만들기 - cv2.dnn.blobFromImage    1. 입력 영상을 블롭(blob) 객체로 만들어서 추론을 진행해야 합니다.    2. 주의할 점은 인자들을 입력할 때 모델 파일이 어떻게 학습되었는지 파악하고 그에 맞게 입력을 해줘야 합니다.    3. 하나의 영상을 추론할 때는 cv2.dnn.blobFromImage 함수를 이용하여 1개의 블롭객체를 받고    4. 여러 개의 영상을 추론할 때는 cv2.dnn.blobFromImags 함수로 여러 개의 블롭 객체를 받아서 사용합니다. 6. cv2.dnn.blobFromImage(image, scalefactor=None, size=None, mean=None, swapRB=None, crop=None, ddepth=None) -> retval    1. • image: 입력 영상    2. • scalefactor: 입력 영상 픽셀 값에 곱할 값. 기본값은 1.    3. • size: 출력 영상의 크기. 기본값은 (0, 0).    4. • mean: 입력 영상 각 채널에서 뺄 평균 값. 기본값은 (0, 0, 0, 0).    5. • swapRB: R과 B 채널을 서로 바꿀 것인지를 결정하는 플래그. 기본값은 False.    6. • crop: 크롭(crop) 수행 여부. 기본값은 False.    7. • ddepth: 출력 블롭의 깊이. CV\_32F 또는 CV\_8U. 기본값은 CV\_32F.    8. • retval: 영상으로부터 구한 블롭 객체. numpy.ndarray. shape=(N,C,H,W). dtype=numpy.float32.    9. scalefactor은 딥러닝 학습을 진행할 때 입력 영상을 0~255 픽셀값을 이용했는지, 0~1로 정규화해서 이용했는지에 맞게 지정해줘야 합니다. 0~1로 정규화하여 학습을 진행했으면 1/255를 입력해줘야 합니다.    10. size : 학습할 때 사용한 영상의 크기를 입력합니다. 그 size로 resize를 해주어 출력합니다.    11. mean : 학습할 때 mean 값을 빼서 계산한 경우 그와 동일한 mean 값을 지정합니다.    12. swapRB : RGB에서 R값과 B값을 바꿀것인지 결정합니다.    13. corp : 학습할 때 영상을 잘라서 학습하였으면 그와 동이하게 입력해야 합니다.    14. ddept : 대부분의 경우 CV\_32F를 사용합니다.    15. 반환값의 shape=(N,C,H,W)인데 N은 갯수, C는 채널 갯수, HW는 영상 크기를 의미합니다. 7. 3. 네트워크 입력 설정하기 - cv2.dnn\_Net.setInput    1. readNet으로 만든 객체에 .setInput 함수로 적용할 수 있습니다.    2. name은 입력 레이어 이름을 지정할 수 있지만 보통 스킵합니다.    3. scalefactor, mean은 블롭을 생성할 때 지정해주었으므로 기본값을 이용합니다. 8. cv2.dnn\_Net.setInput(blob, name=None, scalefactor=None, mean=None) -> None    1. • blob: 블롭 객체    2. • name: 입력 레이어 이름    3. • scalefactor: 추가적으로 픽셀 값에 곱할 값    4. • mean: 추가적으로 픽셀 값에서 뺄 평균 값 9. 4. 네트워크 순방향 실행(추론) - cv2.dnn\_Net.forward    1. 추론을 진행할 때 이용하는 함수입니다.    2. 네트워크를 어떻게 생성했냐에 따라 출력을 여러 개 지정할 수 있습니다. (outputNames) 10. cv2.dnn\_Net.forward(outputName=None) -> retval 11. cv2.dnn\_Net.forward(outputNames=None, outputBlobs=None) -> outputBlobs     1. • outputName: 출력 레이어 이름     2. • retval: 지정한 레이어의 출력 블롭. 네트워크마다 다르게 결정됨.     3. • outputNames: 출력 레이어 이름 리스트     4. • outputBlobs: 지정한 레이어의 출력 블롭 리스트 |
| 오후 | * 1. # 입력 영상을 불러오는 파일 이름   2. filename = 'space\_shuttle.jpg'   3. # 명령 프롬프트에 임시파일 이름을 입력하여 파일을 불러올 수 있도록 하는 코드   4. if len(sys.argv) > 1:   5. filename = sys.argv[1]   7. # 입력 영상 불러오기   8. img = cv2.imread(filename)   9. if img is None:   10. print('Image load failed!')   11. sys.exit()   13. # 네트워크 불러오기   14. # Caffe 설치된 파일 불러오기   15. model = 'googlenet/bvlc\_googlenet.caffmodel'   16. config = 'googlenet/deploy.prototxt'   17. # dnn 객체 생성   18. net = cv2.dnn.readNet(model, congfig)   19. if net.empty():   20. print('Network load failed!')   21. sys.exit()   23. # 클래스 이름 불러오기   24. # 이전에 다운 받아온 클래스 이름 불러오기   25. # classNames에 1000개의 클래스 이름 등록   26. classNames = None   27. with open('googlenet/classification\_classes\_ILSVRC2012.txt', 'rt') as f:   28. classNames = f.read().rstrip('\n').split('\n')   30. # 추론   31. # (1) : 0~255 픽셀값 그대로 이용, (224,224): 입력 영상 크기, (104,117,123): 평균 영상 크기   32. blob = cv2.dnn.blobFromImage(img, 1, (224, 224), (104, 117, 123))   33. net.setInput(blob)   34. prob = net.foward() # 확률에 대한 ndaaray 리턴   35. # 추론 결과 확인 & 화면 출력   36. out = prob.flatten()   37. classId = np.argmax(out) # 가장 큰 값 저장   38. confidence = out[classId] # 확률 값   39. # 확률 값을 보여주기 위해 text값을 제너레이션   40. text = f'{classNames[classId]} ({confidence \* 100:4.2f}%)'   41. cv2.putText(img, text, (10, 30), cv2.FONT\_HERSHEY\_SIMPLEX, 0.8, (0, 0, 255), 1, cv2.LINE\_AA)   42. cv2.imshow('img', img)   43. cv2.waitKey()   44. cv2.destroyAllWindows() |