제가 이번에 설명할 주제는 torchvision에서의 객체 검출 미세조정입니다. 여기서는 미리 학습된 Mask R-CNN 모델을 미세조정하는 것에 대해 설명하겠습니다.

간단하게 목차는 다음과 같습니다. 먼저 데이터셋과 모델을 정의한 뒤에 forward 메소드를 테스트해보는 과정을 통해 미세조정과정이 어떻게 이루어지는 지 알아보도록 하겠습니다.

먼저 데이터 셋을 정의하는 과정입니다. 데이터 셋은 Penn-Fudan 데이터 베이스를 사용하고 이 데이터 셋에는 보행자 인스턴스가 있는 이미지들이 포함되어 있습니다. 우리는 이 이미지를 사용하여 인스터스를 분할하는 모델을 학습할 것입니다. 인스턴스가 분할되면 오른쪽 사진처럼 각 인스턴스마다 다른 색으로 표현되는 형태가 만들어집니다.

먼저 데이터를 불러온 뒤에 압축을 풀고 첫번째 이미지를 예시로 가져오겠습니다. 첫번째 이미지로 불러온 결과가 가운데에 위치한 사진입니다.

코드를 통해 데이터셋을 정의해보도록 하겠습니다. 먼저 init 메소드를 통해 클래스의 기본적인 요소들을 확인합니다. Self.root는 루트 디렉토리의 주소를 갖고, 아래의 self.imgs 와 self.masks 는 해당 주소에 있는 디렉토리 내의 파일과 디렉토리를 정렬하여 리스트의 형태로 갖게 됩니다.

그 다음에는 getitem 메소드를 정의하는데, 먼저 이미지와 마스크를 주소를 통해 읽어온다. 이때 이미지만 RGB로 변환하고 마스크는 변환하지 않는데 이유는 마스크의 각 색상은 다른 인스턴스를 의미하기 때문이라고 합니다. 이후 mask\_path를 통해 PIL 이미지를 numpy array로 변환합니다. 이후 mask에 대한 정보를 object ids 에 저장합니다. 컬러 인코딩 되어있는 마스크를 바이너리 마스크 세트로 나눈 뒤에 각 마스크의 바운딩 박스 좌표를 모아 boxes 에 저장합니다. 저장이 끝나면 boxes 배열과 레이블, masks를 모두 텐서 타입으로 변환합니다. 앞에서 구한 바운딩 박스의 좌표를 이용하여 바운딩 박스 영역의 크기를 구하고 target array의 각 항목에 구한 값들을 모두 저장합니다. 이때 iscrowd는 군중상태인지에 대한 정보인데, 저는 사람이 겹쳐있는지에 대한 정보라고 이해했습니다. 위의 과정이 모두 끝난 뒤에 transform 정보가 None이 아니면 transform 정보를 바탕으로 이미지와 타겟을 변환한 뒤에 return 하게 됩니다.

데이터 셋에 대한 정의가 끝났기 때문에 이제 모델을 정의하도록 하겠습니다. 모델은 Faster R-CNN에 기반한 Mask R-CNN을 사용했습니다. Faster R-CNN은 이미지에 존재할 수 있는 객체에 대한 바운딩 박스와 클래스 점수를 모두 예측하는 모델이고, Mask R-CNN은 각 인스턴스에 대한 분할 마스크를 예측하는 레이어를 Faster R-CNN에 추가한 모델입니다

torchvision에서 모델을 수정하는 방법에는 크게 두 가지가 있는데 하나는 미리 학습된 모델을 이용하면서 마지막 레이어의 수준만 미세 조정하는 것이고, 다른 하나는 모델의 백본을 다른 것으로 교체하는 것이라고 합니다. 여기서 백본이란 입력 이미지를 feature map으로 변형시켜주는 부분으로 이전 발표에서 등장했던 ResNet과 VGG16과 같은 것이 있습니다.

미리 학습된 모델로부터 미세 조정을 하기 위해서는 torchvision.models.dectection을 통해 이미 학습되어있는 모델을 불러오고, classfier를 교체 합니다. 이때 클래스의 개수를 사용자가 정의 합니다. 이번 데이터 셋에서는 특정 부분이 사람인지 아닌지를 구별하기 원하기 때문에 사람과 나머지 배경의 2가지 클래스가 필요하기 때문에 num classes의 값은 2가 됩니다. 이후 classfier에서 사용할 input feature의 수를 얻어 저장한 뒤에 model의 box predictor를 사용자가 정의한 num class를 가진 predictor로 바꿉니다. 이때 FastRCNNPredictor를 사용하는 이유는 불러왔던 model이 fast R CNN을 기반으로 하는 faster R CNN 모델이기 때문으로 이해했습니다.

두번째 방법으로 다른 backbone을 추가하는 방법에 대해서 설명하도록 하겠습니다. 먼저 분류 목적으로 미리 학습된 model인 mobilenet\_v2를 로드하고 feature값을 리턴받습니다. Faster R CNN의 경우에 출력 채널 수를 알아야 하기 때문에 out\_channels 값을 추가합니다.

이후 anchor\_generator를 생성하는데 하나의 anchor를 기준으로 해서 5개의 서로 다른 사이즈와 3개의 서로 다른 비율로 anchor box를 생성하도록 설정합니다. 이때 aspect\_ratios는 화면 비를 의미합니다. 이렇게 설정하면 입력으로 들어오는 feature map에 대해서 수많은 anchor box를 생성하게 됩니다.

Anchor generator 설정이 끝나면 RPN을 통해 얻은 각각의 region proposals 을 동일한 크기로 처리하기 위한 roi pooler 를 설정합니다. Output size와 함께 sampling ratio 을 설정한 뒤에 위에서 만들었던 설정과 함께 Faster R CNN모델로 합치게 됩니다.

여기서 다루는 데이터 셋의 경우 샘플의 수가 작기 때문에 첫번째 방법이었던 미세 조정 방식을 사용합니다. 먼저 미리 학습된 인스턴스 분할 모델을 읽어오고, 위에서와 같이 분류를 위한 input feature의 수를 얻습니다. 그리고 head의 box predictor를 앞에서 정의한 것으로 바꿉니다. 그 다음 mask classfier를 위해 input feature의 수를 얻습니다. 이때 mask classfier에 대한 정보를 얻는 이유는 위에서 설명했던 것처럼 Mask R CNN이 Faster R CNN에 mask branch 이 추가된 것이기 때문입니다. Mask predictor까지 새로운 것으로 변경이 끝나면 model을 리턴하여 사용합니다.

이것은 데이터 셋의 데이터가 부족하기 때문에 사용하는 함수로 학습 시에 50%의 확률로 이미지를 좌우 반전 함으로써 다른 이미지를 학습한 것과 비슷한 효과를 얻게 해줍니다.

이제 앞에서 설정한 모델들을 이용해 테스트를 진행하는 과정입니다. 앞에서처럼 dataset이 가지는 클래스의 수를 설정하고, 데이터 셋을 불러옵니다. 이때 앞에서 정의했던 get transform을 사용하여 가져옵니다. 그리고 데이터셋을 학습용과 테스트용으로 나누는데 예시에서는 전체의 50개를 테스트에 사용했고 나머지를 학습에 사용했습니다. 데이터로더 또한 학습용과 테스트용으로 정의합니다. 앞에서 정의 했던 get model instance segmentation을 통해 모델을 가져오고 그것을 cpu 또는 gpu로 옮긴 뒤에 optimizer를 생성하고 learning rate scheduler를 만듭니다. 이때 사용하는 StepLR은 일정한 스텝마다 learning rate에 gamma값을 곱해주는 스케줄러의 종류라고 합니다.

예시에는 10 epoch만큼 학습했고 각 에포크 마다 학습률을 업데이트 하면서 결과를 출력하면 다음과 같이 출력됩니다. 학습된 모델에 예시 이미지를 넣어서 결과를 확인하면 다음과 같이 인스턴스를 분리해서 예측하게 됩니다.

다음 주제는 컴퓨터 비전을 위한 transfer learning입니다. 실제로 충분한 크기의 데이터 셋을 갖추기가 힘들기 때문에, 기존에 매우 큰 데이터 셋에 대해 학습된 모델을 가져와서 저희가 원하는 작업에 사용하게 됩니다. 이 학습 방식은 두가지 형태로 이루어지는데 하나는 합성곱 신경망을 미세조정하는 것이고, 다른 하나는 fixed feature를 추출하는 형태로 이루어집니다.

합성곱 신경망을 미세조정하는 것은 초기 값을 무작위로 주는 것이 아닌 미리 학습한 신경망으로 초기화 하는 것이고, fixed feature를 추출하는 것은 마지막의 Full connected layer를 제외한 모든 신경망의 가중치를 고정해서 사용하는 것입니다. 이때 마지막에 존재하는 fully connected layer는 무작위의 가중치를 갖는 layer로 대체되어 이 layer만 학습하게 됩니다.

튜토리얼의 예시에서는 이미지에서 개미와 벌을 분류하는 모델을 학습하는 것에 대해 소개하고 있습니다. 사용되는 데이터 셋은 120장의 학습용 이미지와 75장의 테스트용 이미지가 있습니다. 일반적으로는 아주 작은 양의 데이터 셋이지만 전이학습을 사용하기 때문에 큰 문제가 없습니다.

먼저 데이터를 로드한 뒤에 학습을 효율적이고 정확하게 하기 위해 데이터를 증가시키고, Normalize 시킵니다. 먼저 train 데이터에 대해서는 randomresizedcrop 과 randomhorizontalflip을 사용하는데, random resized crop 은 이미지를 지정한 크기로 랜덤하게 잘라서 다른 이미지인것처럼 활용하는 방법이고, horizontalflip은 50%의 확률로 이미지를 뒤집음으로서 다양한 데이터를 얻게 합니다. 해당 데이터들을 텐서로 변환한 후에 normalize 시키는데 앞의 세 값은 평균이고 뒤의 세 값은 표준변차입니다. 테스트용 데이터 셋에도 transform을 적용시키는데, 이미지의 사이즈를 변화시키는 resize와 이미지의 가운데를 지정크기만큼 잘라내는 center crop을 사용했습니다. 이후 train data에서처럼 텐서로 바꾼뒤에 normalize를 해줍니다.

transform설정이 끝나면 이미지를 불러오기 위한 dataset을 설정합니다. 해당 주소로부터 이밎를 불러온뒤에 dataloader와 datasize, class\_names를 생성합니다. 이것은 일부 이미지를 시각화 해서 데이터를 제대로 가져왔는지, 확인하기 위한 코드 입니다.

다음은 모델을 학습하기위한 train model 함수입니다. 이 함수의 scheduler parameter는 이전 주제에서 나왔던 learning rate scheduler를 의미합니다. 먼저 deepcopy를 통해 model의 state dict를 불러옵니다. 그리고 parameter로 입력된 에폭 만큼 학습을 진행합니다. 각 에폭은 학습 단계와 검증단계를 거지는데 학습단계일 경우에만 연산 기록을 추척하여 역전파와 최적화를 진행합니다. 위의 과정을 수행하면 각 에폭마다 loss값과 정확도 값이 나오는데 이것이 이전에 구해졌던 best값들보다 성능이 좋다면 best model에 현재 모델이 적용되게 됩니다. 주어진 에폭 만큼 학습이 완료되면 best\_model\_wts에는 가장 나은 모델의 가중치가 저장 되어있고, 그것을 load state dict를 통해 불러와서 return 합니다.

다음 코드는 일부 이미지에 대한 예측값을 보여주는 일반화된 함수입니다. 주어진 이미지의 개수만큼 이미지를 가져와 예측한 결과를 프린트합니다. 결과는 오른쪽과 같이 프린트됩니다.

다음은 Convolution Network를 미세조정하는 부분을 설명하겠습니다. 먼저 미리 학습되어있는 모델을 불러온 뒤에 아까 다루었던 것처럼 input feature값을 저장하고, model의 끝부분에 있는 fully connected layer를 초기화 합니다. 다음으로 loss function 과 optimizer를 정의한 뒤에 스케줄러의 형태로 정의합니다. 여기에선 7 스텝마다 학습률에 0.1을 곱한다고 이해하시면 될 것 같습니다.

위에서 만든 모델을 train model의 parameter로 전달한 뒤에 학습을 진행하면 아래와 같은 결과가 나오게 됩니다.

다음은 Convolution Network에서 위에서와는 다르게 마지막 layer를 제외한 모든 부분을 고정해서 학습하는 방법입니다. 이미 학습된 resnet을 불러온 뒤에 그 parameter를 고정시켜서 backward중에 gradient가 계산되지 않도록 합니다. 그리고 위에서 했던 것처럼 fully connected layer에 input feature를 저장하고 새로 fully connected layer를 정의합니다. 이때 새로 생기는 layer는 requires\_grad 값이 true이기 때문에 backward 시에 gradient가 계산됩니다.

loss함수 역시 정의한 뒤에 optimizer를 정의할 때는 위에서와 다르게 fully connected layer의 parameter만 최적화시킵니다. Learning rate 스케줄러 역시 위와 같게 설정한 뒤에 학습을 진행시킵니다.

결과는 다음과 같이 나오게 됩니다. 여기까지 해서 2가지의 방법으로 전이학습을 하는 것에 대해 설명 드렸니다. 질문 있으시면 해주시면 될 것 같습니다.