**인공지능 통계분석 CNN 실습**

20190917 통계학과 신효민

**수업내용 실습**

텍스트이(가) 표시된 사진

자동 생성된 설명

텐서플로우의 케라스 패키지를 이용하여 MNIST 숫자 이미지 분류에 사용한다. 먼저, 간단한 컨브넷 모델을 만들기 위해 Conv2D와 MaxPooling2D 층을 쌓아 올린다. MNIST 숫자 이미지는 1채널 28X28 사이즈의 이미지이기 때문에 input shape을 위와 같이 설정한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

ConvNet의 마지막 층의 출력 텐서를 완전 연결 네트워크에 넣어 훈련시킨다. 그러기 위해서 이미지를 1차원으로 펼쳐야 하고, Flatten을 사용한다. 그 다음 몇 개의 Dense층을 추가한다. 0~9까지 10개의 클래스를 분류하기 위해서 softmax는 10개의 출력을 내야 한다. 전체 네트워크의 구조를 확인하면 다음과 같다.

테이블이(가) 표시된 사진

자동 생성된 설명

(3,3,64) 출력이 (576,) 크기의 벡터로 펼쳐지고 Dense층으로 주입되는 것을 볼 수 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명

테이블이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

훈련시킨 후 평가해보면 간단한 모델로 굉장히 높은 정확도를 보이는 것을 알 수 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

테이블이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명

ConvNet을 위한 예제로 강아지와 고양이 데이터셋을 통해 이중 분류를 수행하고자 한다. 캐글에서 데이터를 받아와 훈련, 검증, 테스트 세트와 강아지 고양이를 분류하여 적절하게 파일을 폴더 내에 정리한다.

테이블이(가) 표시된 사진

자동 생성된 설명

훈련용 고양이, 강아지 이미지가 각각 1000개씩, 검증용 고양이, 검증용 강아지, 테스트용 고양이, 테스트용 강아지 이미지가 각각 500개씩 준비된 것을 확인한다. 여기서 강아지와 고양이 사이에 데이터 수의 불균형이 없는 것을 확인할 수 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명

이미지가 크고 복잡하기 때문에 Conv2D와 MaxPooling2D 층을 반복적으로 사용하여 특성맵의 크기를 줄인 후 Flatten으로 펼쳐 완전 연결 네트워크에 주입한다. 이진분류 문제이므로 사이즈가 1인 Dense층에 sigmoid 활성화 함수로 모델을 끝낸다. 모델의 구조는 다음과 같다.

테이블이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

RMSprop 옵티마이저를 선택하고, 마지막 층의 활성화 함수가 시그모이드 함수이기 때문에 loss로 binary\_crossentropy 함수를 선택한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

ImageDataGenerator를 통해 이미지 전처리를 실시한다. 데이터를 스케일링 하고 target size와 batchsize를 지정해주면 조건에 맞는 데이터 배치를 무한정 만들어준다.

텍스트이(가) 표시된 사진

자동 생성된 설명

무한히 배치가 생성되는 것을 막기 위해 반복문 안에 break를 사용한다.

테이블이(가) 표시된 사진

자동 생성된 설명

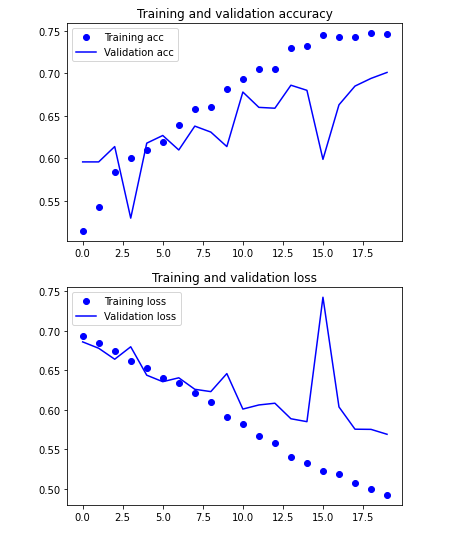
텍스트이(가) 표시된 사진

자동 생성된 설명

Steps\_per\_epoch 하이퍼 파라미터에서 제너레이터에서 얼마나 많은 샘플을 뽑을 것인지 알려준다. 20개의 샘플이 하나의 배치로 2000개의 샘플을 모두 처리해야하기 때문에 100개의 배치를 뽑는다고 설정한다. Validation data에 제너레이터를 지정해준 경우, 마찬가지로 무한히 배치가 반환되는 것을 막기 위해 validation steps를 지정한다. 또한 훈련이 끝난 후 모델을 저장해준다.

텍스트이(가) 표시된 사진

자동 생성된 설명



모델의 손실과 정확도를 그래프로 나타낸다. 그래프를 확인하면 훈련 정확도가 점점 높은 성능을 보여도 검증 데이터는 70%를 겨우 도달하는 것을 볼 수 있다. 즉, 과대적합이 발생하고 있다. 손실 또한 비슷한 형태를 보인다. 이러한 문제를 해결하기 위하여 Data Augmentation을 사용한다.

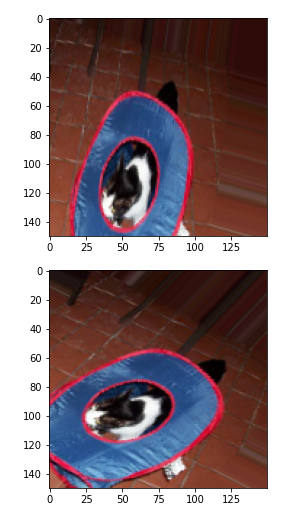
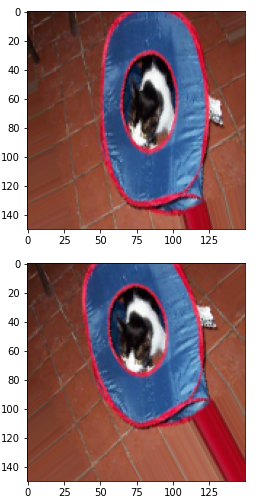
텍스트이(가) 표시된 사진

자동 생성된 설명

데이터 제너레이터에 여러가지 이미지를 여러가지 랜덤 변환하도록 지정한다. 예를 들면 이미지를 회전하거나, 기울이거나 확대하고 뒤집는 등 여러가지 변환을 실시할 수 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명



위와 같이 같은 이미지 데이터를 조금씩 변환하여 다른 이미지를 만들고, 데이터 증식은 모델에 더 많은 데이터를 훈련시키고 싶을 때나, 과대적합을 어느정도 막고 싶을 때 쓸 수 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명

데이터 증식 또한 기존 이미지의 변형이기 때문에 과대적합을 막기에 충분하지 않을 수 있어 DropOut 층을 추가한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

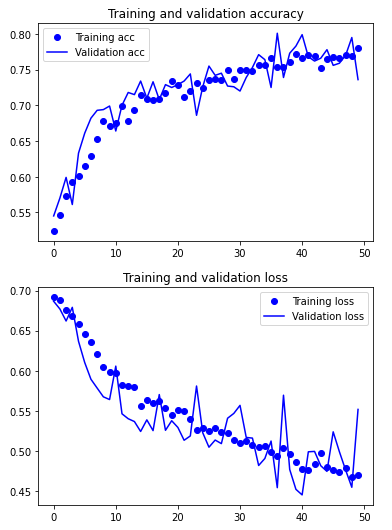
자동 생성된 설명



데이터 증식과 드롭아웃을 이용해 모델을 훈련하고, 해당 모델을 저장한다.

텍스트이(가) 표시된 사진

자동 생성된 설명



그래프를 사용하여 훈련 세트와 검증 세트의 정확도와 손실을 보면, 전보다 과대적합이 줄고 성능도 조금 더 좋아진 것을 볼 수 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명

테이블이(가) 표시된 사진

자동 생성된 설명

데이터 개수가 적기 때문에 Pretrained network 중 하나인 VGG16을 사용하여 모델을 훈련시킨다.

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

ImageDataGenerator를 활용하여 이미지 레이블을 넘파이 배열로 추출한다. 추출된 특성의 크기는 (sample, 4, 4, 512)이고, (sample, 8192)크기로 Flatten 해준다. 그 다음 과적합을 막기 위해 드롭아웃을 사용하고, 훈련을 진행한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

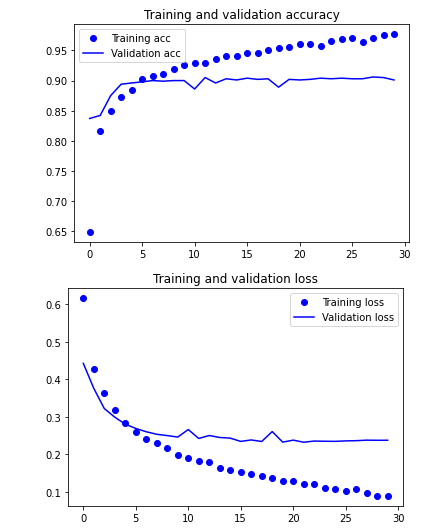
텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

이전과 같은 방식으로 훈련한 뒤 손실과 정확도 곡선을 그려본다.



90%의 정확도까지 도달했다. 이전보다 더 나아진 성능을 볼 수 있다. 그러나 과대적합 문제가 초반부터 심하게 생기는 것을 볼 수 있다. 데이터 증식을 사용하지 않았기 때문이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

이전보다 비용이 많이 들지만 좋은 성능을 보일 것으로 기대되는 conv\_base 모델의 확장을 실시한다. 모델의 구조는 다음과 같다.

테이블이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

Conv\_base.trainable을 False로 지정하여 모델의 Convolution layer를 동결한다. 사전 훈련된 모델을 사용하기 때문에 기존의 학습된 내용이 훈련하는 동안 수정되는 것을 막는다.

텍스트이(가) 표시된 사진

자동 생성된 설명

앞의 코드와 마찬가지로 훈련을 실시한다.

테이블이(가) 표시된 사진

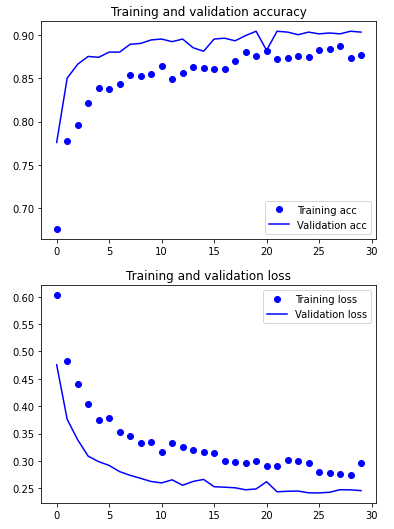
자동 생성된 설명



모델을 저장한 후 정확도 손실 곡선을 그린다.

텍스트이(가) 표시된 사진

자동 생성된 설명



전보다는 과대적합이 많이 줄어든 것을 확인할 수 있다.

테이블이(가) 표시된 사진

자동 생성된 설명

마지막 3개의 Convolution Layer를 미세 조정한다. 마지막 세 개의 층은 완전히 학습을 동결시켰던 아까의 모델과는 다르게 다시 학습이 진행될 것이다. 마지막의 층들은 일반적이고 재사용 가능한 특성들을 인코딩 하기 때문에 조정하여도 괜찮지만, 상위 층들은 조금 더 구체적이고 데이터에 특화된 특성을 학습하기 때문에 하위의 층을 학습에 사용한다. 또한 훈련하는 파라미터가 많으면 과대적합의 위험이 있기 때문에 적은 개수의 층만 학습에 사용한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

이와 같이 미세 조정 층을 설정한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

테이블이(가) 표시된 사진

자동 생성된 설명

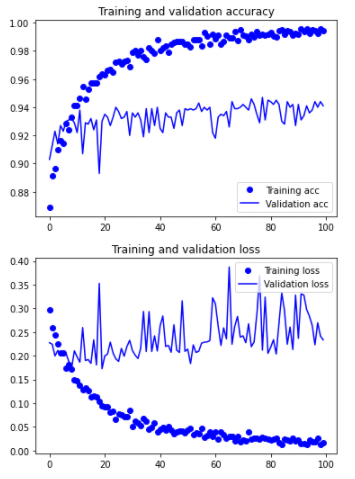


세 개의 층의 파라미터를 조금씩 수정하기 위해서 learning rate를 낮춘 RMSprop를 사용한다. 그리고 모델을 저장한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

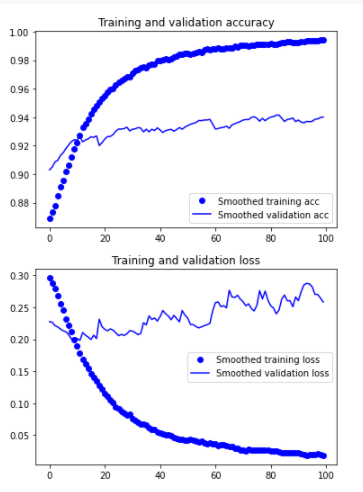
정확도 손실 그래프를 그리면 다음과 같다.



그래프가 불규칙하게 보이기 때문에 지수 이동 평균을 이용하여 정확도와 손실을 부드럽게 해준다.

텍스트이(가) 표시된 사진

자동 생성된 설명



검증 정확도 곡선이 훨씬 잘 보인다. 정확도가 아주 조금 상승하였다. 그러나 손실 함수는 의미 있는 변화를 얻지 못했다.

텍스트이(가) 표시된 사진

자동 생성된 설명

테스트 데이터에 이 모델을 사용한 결과 93%의 정확도를 얻었다. 꽤 높은 결과이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

MobileNetV2를 이용한 Transfer Learning을 실시한다. Conv\_base를 MobileNetV2 메서드로 지정한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

하위 층을 훈련 가능하게 조정한 후 완전 연결 신경망을 conv\_base에 추가하여 훈련을 진행한다.

테이블이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

94%라는 아까보다 더 나아진 정확도를 얻을 수 있다.

테이블이(가) 표시된 사진

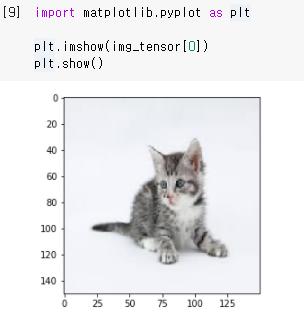
자동 생성된 설명

저장해두었던 고양이 강아지 분류 모델을 불러온다.

텍스트이(가) 표시된 사진

자동 생성된 설명

고양이 이미지를 불러온 후 규격에 맞게 이미지의 사이즈를 변환한다.



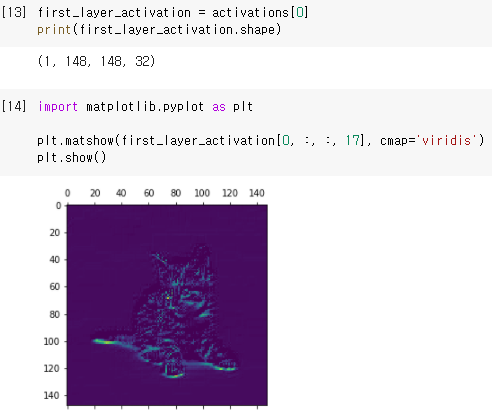
불러온 고양이 이미지는 위와 같다.

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명



확인하고 싶은 층의 filter, 그러니까 고양이 이미지에 대한 첫 번째 합성곱 층의 활성화 값을 이용하여 특성맵을 그리면 위와 같은 결과가 나온다.

텍스트이(가) 표시된 사진

자동 생성된 설명

네트워크의 모든 특성맵을 확인할 수 있는 코드이다.

테이블이(가) 표시된 사진

자동 생성된 설명

지도이(가) 표시된 사진

자동 생성된 설명

확인한 특성맵은 요약하자면 위와 같은 형태이다. 초기의 특성맵은 사진에 있는 정보가 거의 유지되지만 후반부로 갈수록 점점 더 이미지가 추상적으로 되고 더 구체적이고 작은 단위의 개념을 학습하게 된다. 즉, 이미지 자체의 정보보다 이미지 클래스의 정보가 더 강조되는 것이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

각 필터가 반응하는 시각적 패턴을 그려 컨브넷이 학습한 필터를 조사한다. 특정 합성곱 층의 필터 값을 최대화하는 손실함수를 정의하고, 이 활성화 값을 최대화하기 위해 확률적 경사 상승법을 사용한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

결과 텐서를 이미지로 출력하기 위해 후처리를 실시한다.

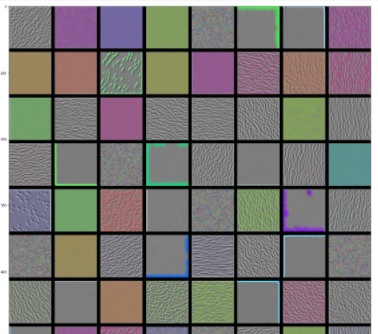
텍스트이(가) 표시된 사진

자동 생성된 설명

함수가 필터의 반응을 최대화시켜 그 활성화 된 이미지를 출력한다. 위와 같이 필터 0번은 점 패턴에 반응하는 것을 알 수 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명

쇼지, 건물이(가) 표시된 사진

자동 생성된 설명

합성곱 블록의 첫 번째 층의 필터가 반응해 활성화된 특성맵의 일부를 살펴보면 위와 같다.

테이블이(가) 표시된 사진

자동 생성된 설명

클래스 활성화 맵을 시각화하는 방법이다. 입력 이미지에 대한 클래스 활성화의 히트맵을 만든다. 입력 이미지가 각 클래스로 분류될 때 어느 부분의 어떤 모습이 중요하게 작용하는 지를 알 수 있다. 먼저 이를 직접 보기 위해 아프리카 코끼리 이미지를 불러온다.

텍스트이(가) 표시된 사진

자동 생성된 설명

모델에 이미지를 넣기 위해서 이미지의 크기를 조정하고 전처리 함수를 적용한다. 그 다음 이미지를 사전 훈련된 모델에 적용시키고 결과를 이해하기 쉽게 디코딩한다.

이 이미지에 대해 아프리카 코끼리일 확률이 90.9%, 코끼리일 확률이 8%, 인도 코끼리일 확률이 0.5%라고 인식했다. 예측 벡터에서 최대로 활성화된 항목은 아프리카 코끼리에 대한 것이며, 386번 인덱스이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

이 이미지에서 가장 아프리카 코끼리 다운 부위를 시각화하면 위와 같다. 이를 원본 이미지와 겹치면 다음과 같다.

텍스트, 잔디이(가) 표시된 사진

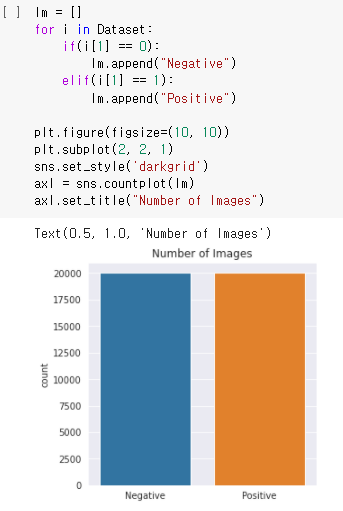
자동 생성된 설명

**다른 예제에 적용시키기**

텍스트이(가) 표시된 사진

자동 생성된 설명

균열이 있는 데이터와 없는 데이터가 각각 20000개씩 있는 데이터셋을 불러온 후 Color 채널이 분류에 큰 영향을 끼치지 않을 것으로 판단하여 Gray Scale로 변환한다.



균열이 있는 데이터와 없는 데이터가 각각 20000개씩, 데이터 개수의 불균형 없이 있는 것을 볼 수 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명

정규화를 거친 후 실제 이미지의 모습을 보면, 균열이 없는 이미지는 왼쪽과 같고 있는 이미지는 오른쪽과 같다.

테이블이(가) 표시된 사진

자동 생성된 설명

Conv2D와 MaxPooling2D를 반복한 후 과적합을 방지하기 위해 드롭아웃을 실시하고 네트워크에 넣어 softmax 활성화 함수로 마지막에 확률을 출력하도록 한다.

텍스트이(가) 표시된 사진

자동 생성된 설명



Adam 옵티마이저를 이용하고, sparse categorical crossentropy 손실 함수를 활용하여 훈련시킨 후 정확도 손실 그래프를 그려보면 위와 같으며 간단한 CNN 모델로도 굉장히 높은 정확도를 도출해내는 것을 알 수 있다.