**딥러닝과 인공지능 2차과제**

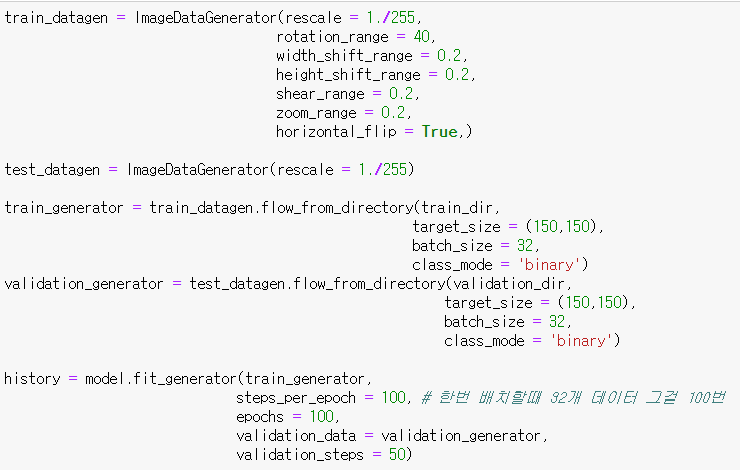
20152640

김태현

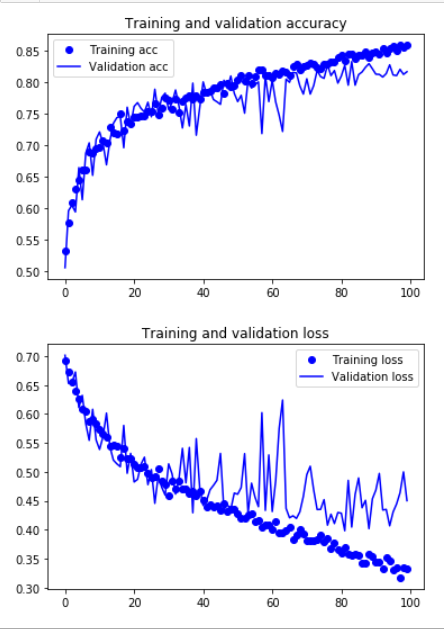
기본 CNN Model의 구조형태 해당 구조를 변경하여 성능을 향상시킬 예정이다.



데이터 증식 및 model fitting



기본적으로 주어져 있는 cats and dogs 데이터 증식 후 100epochs동안 학습시킨 accuracy와 loss

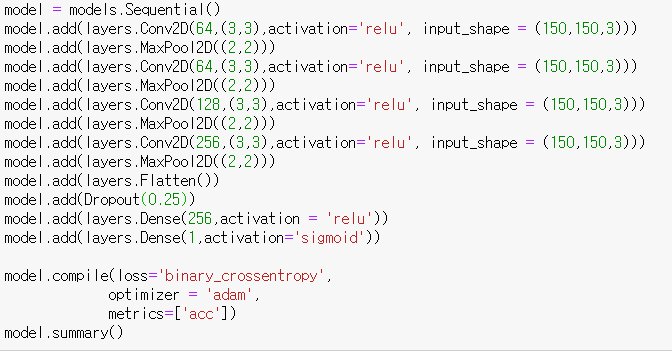


Validation의 accuracy가 가장 높을 때의 epochs의 값을 찾아보면 84epochs 일 때이다. 이때 accuracy는 0.8325, loss는 0.4045이다.

이제 Model의 layer의 층과 featuremap의 개수 등을 변경하여 성능을 조금 향상시켜 보고자 한다. 나는 이번 과제에서 3가지 항목 변화를 통해 성능을 향상시키고자 한다.

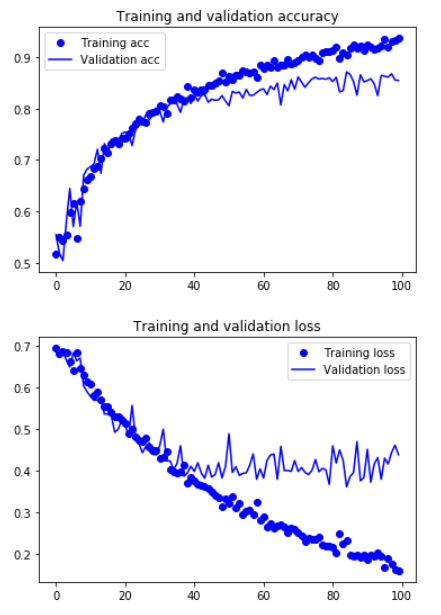
1. Compile을 할 때 RMSprop을 현재 사용하고 있는데 이를 adam과 비교해보고자 한다.
2. Dropout의 비율을 조정해 dropout의 비율이 성능에 영향을 주는지 파악하고자 한다.
3. Convolusion network의 피쳐맵의 개수 조정, Dense Layer의 개수 조정을 해보고자 한다.

다양한 방법을 시도해 본 결과 나온 CNN의 구조는 다음과 같다.



CNN의 층은 총4개 그리고 1개의 Denselayer로 앞서 기본적으로 사용했던 층과 같다. 여기서 나는 CNN의 featuremap의 개수를 64개, 64개, 128개, 256개로 변환시킨 결과 성능이 가장 우수했다. 이후 Dropout의 비율을 0.5에서 0.25로 비율을 낮추었을 때 성능이 약간 상승하였다. 그리고 compile을 할 때 optimizer를 RMSprop에서 adam으로 변경하였더니 성능이 올라 갔다.

다음은 해당 모델을 학습시킨 결과 나온 accuracy와 loss 값에 대한 그래프이다.

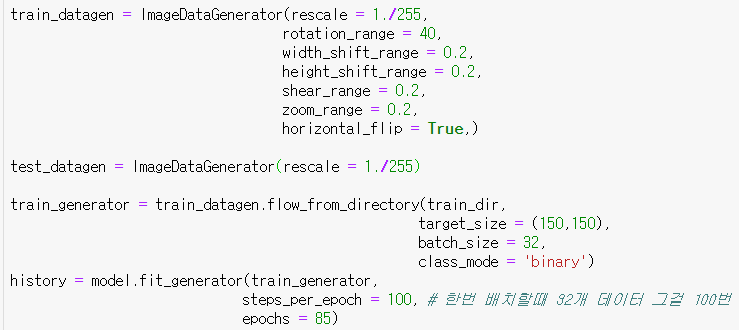


모델의 구조를 변경시킨 결과 모델의 accuracy와 loss 전부 개선하기 이전에 비해 성능이 좋아지는 효과가 나타났다. 특히 accuracy의 경우 85epoch에서 최대 0.8715 이때 loss값의 경우 0.3618이 나오게 되면서 앞서 성능이 가장 우수했던 결과와 비교해 보았을 때 대략 5%의 accuracy가 상승하였고 약 10%의 loss값이 감소하는 효과를 가져왔다.

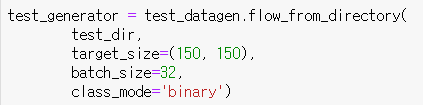
이번 보고서를 작성하며 느낀점은 Compile할 때 optimizer를 무엇을 선택하는가에 따라 네트워크의 구조를 수정하는 것에 비해 더 좋은 성능을 끌어올릴 수 있다는 점이었다. 처음에 default코드에서 adam으로 변경한 결과 최고 accuracy가 0.84가 나온 것을 보며 이러한 생각을 갖게 되었다. 다음으로 배운 것은 Dropout의 비율을 높게 두었을 때 과적합을 막을 수 있다는 장점이 있지만 오히려 최대의 성능을 끌어내지 못하는 문제가 발생할 수 있다는 사실을 알게 되었다. 이것은 내가 Layer구조를 변경시킬 때마다 진행해 보았는데 Dropout의 비율이 매 Network 구조변경마다 바뀌는 모습을 보고 일단 내가 생각하는 최선의 Network구조를 만든 후 Dropout의 비율을 조정하는 것이 좋다는 생각을 갖게 되었다. Network에서 featuremap의 개수를 여러 방향을 가지고 변경해 보았다. 처음에 많은 featuremap을 가지고 점차 줄여가는 식으로 진행하기도 하고 제시한 것처럼 점점 증가시켜가는 방식으로 구현을 해보았다. 결과적으로 Dogs and Cats 데이터의 경우 적은 feature map에서 큰 feature map의 수로 증가시켜 가는 것이 좋다는 결론을 내었다. 또한 layer의 수를 증가시켜 보기도 했는데 추가됨에 따라 학습이 안되는 경우가 발생하기도 하였다. 이는 왜 그런지는 잘 모르겠다. Feature map의 개수를 줄여보았는데 성능이 많이 떨어지는 경향을 보였다. 이것은 Layer의 개수가 줄어듦에 따라 학습이 제대로 되지 않기 때문이라 결론지었다.

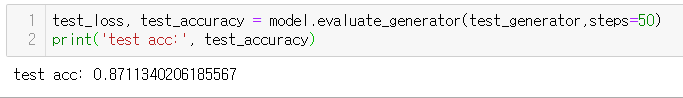
마지막으로 해당 data의 train data와 valid data를 모두 사용하여 85 epoch동안 데이터를 학습시키고 test data로 결과를 산출한 후 보고서를 마무리하고자 한다. **(Train\_2폴더에 valid data까지 삽입)**





데이터를 학습시킨 후 test\_generator를 만들어 model.evaluate\_generator를 활용 step=50을 넣고 accuracy를 확인한다.





Test data를 넣어 성능을 확인한 결과 accuracy가 0.8711이 나왔다. 해당 모델을 validation data로 평가했을 때 0.8715였다. 결과적으로 많은 차이가 나지 않은 것으로 볼 때 학습이 잘 되었음을 알 수 있다.