실험 목적

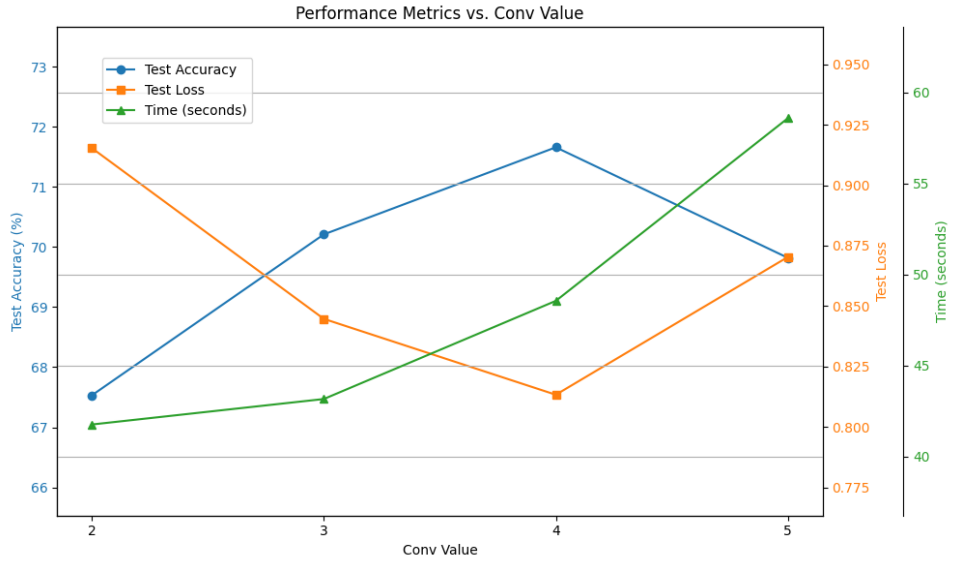
몇 개의 convolution 연산이 RGB이미지에 적합한가 궁금하여 실험을 진행하였다

예상 결과

Convolution 연산이 많은수록 layer가 깊어지므로 성능은 좋아질 것이다. 하지만 연산 과정이 길어지므로 layer가 증가함에 따라 소요 시간 역시 증가할 것이다.

결과

	Time	Test Accuracy	Test Loss
Conv = 2	00:41.77	67.53	0.9154
Conv = 3	00:43.17	70.21	0.8447
Conv = 4	00:48.56	71.66	0.8133
Conv = 5	00:58.59	69.82	0.8703

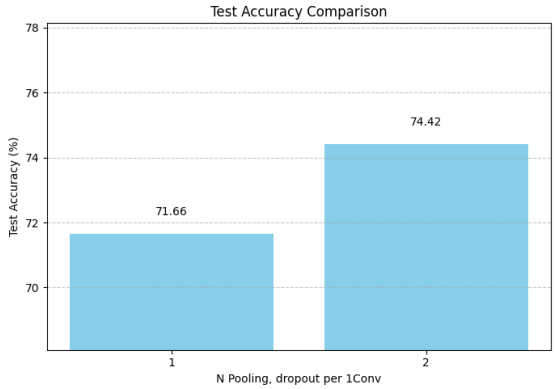


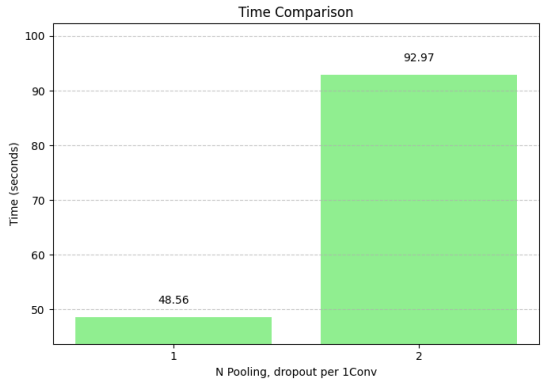
Convolution 연산 수에 따라 소요 시간은 증가하였지만 layer가 4개 일 때 가장 좋은 성능을 보이고 그 이후에는 감소하였다. 이유에 대해 분석해보자면 pooling과 dropout를 convolution 연산이 진행될 때마다 하여 feature가 많이 소실된 것이 문제가 된 것 같다.

Loss가 0.8이상의 값을 가지는 이유 역시 이 이유인 것 같다. 이 결과에 따라 Conv = 4

로 고정한 채로 Conv연산 2번마다 pooling과 dropout을 진행하면 성능향상이 있을 것으로 예상하고 후속실험을 시작하였다.

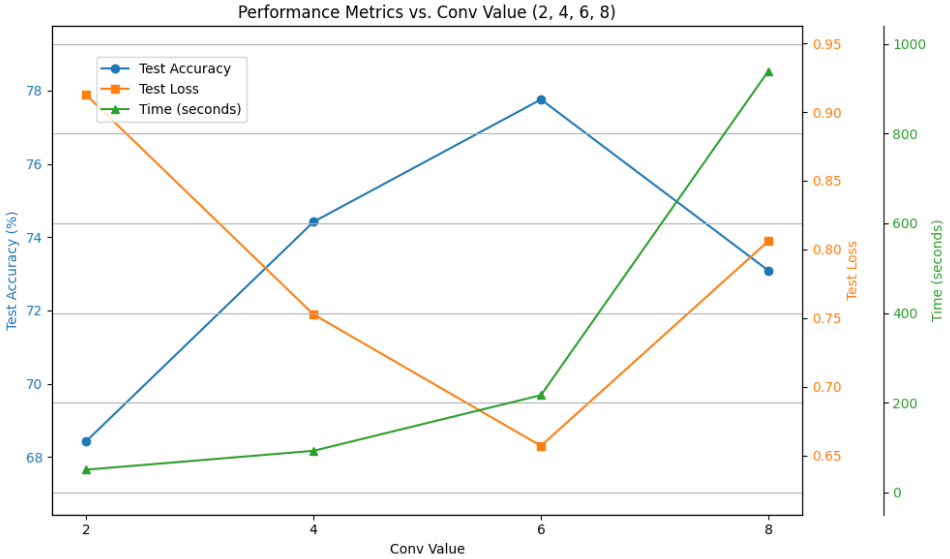
N Pooling,dropout per 1Conv	Time	Test Accuracy	Test Loss
1	00:48.56	71.66	0.8133
2	01:32.97	74.42	0.7476





예상대로 성능은 향상되었지만 시간의 2배가 증가하였다. 원인은 Convolution 연산을 진행할 때 feature map의 크기 차이로 생긴 연산량의 차이로 생각된다. Pooling,dropout per Conv 값을 높이니 feature의 손실이 적어져 성능이 향상됨을 알았으니 이번에는 Conv연산량을 다시 늘려보는 실험을 해볼 것이다. Conv연산량에 따른 pooling,dropout 횟수는 2로 고정하고 진행하였다.

	Time	Test Accuracy	Test Loss
conv = 2	00:50.82	68.43	0.9125
conv = 4	01:32.97	74.42	0.7527
conv = 6	03:36.96	77.76	0.6571
conv = 8	15:39.49	73.09	0.8061



Conv연산 횟수가 늘어날수록 시간은 늘어나지만 시간에 따라서 기대한만큼의 성능 향상이 나오지 않아 conv6일때가 좋은 성능을 보이는 것 같다. 하지만 70퍼대 역시 좋은 성능이 아니기 때문에 수정해야될 부분은 Conv 연산을 진행할 때 OutputSize를 무조건 InputSize의 2배로 잡아 feature map이 데이터에 비해 쓸모없이 커져 좋은 성능이 나오지 않은 것 같다. 따라서 InputSize는 Outputsize의 몇 배가 적합한가로 후속실험을 시작하려한다.