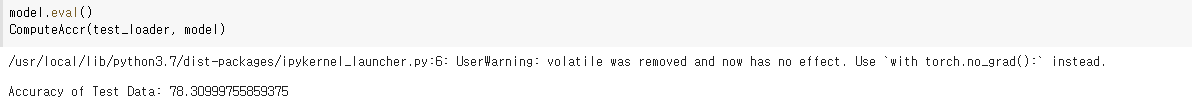
최종 정확도 78.3



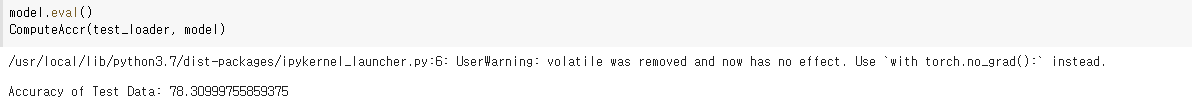
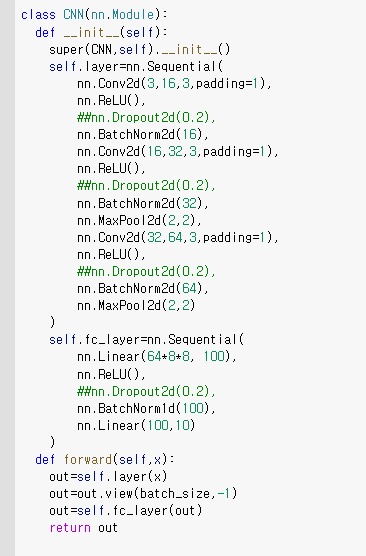
작업환경은 GPU 가속을 위해 구글 colab 을 사용하였다. 우선 아무런 개선 요소를 넣지 않은 기본 상태에서 약 10% 대의 정확도를 기록하였다.

1. Adam optimizer



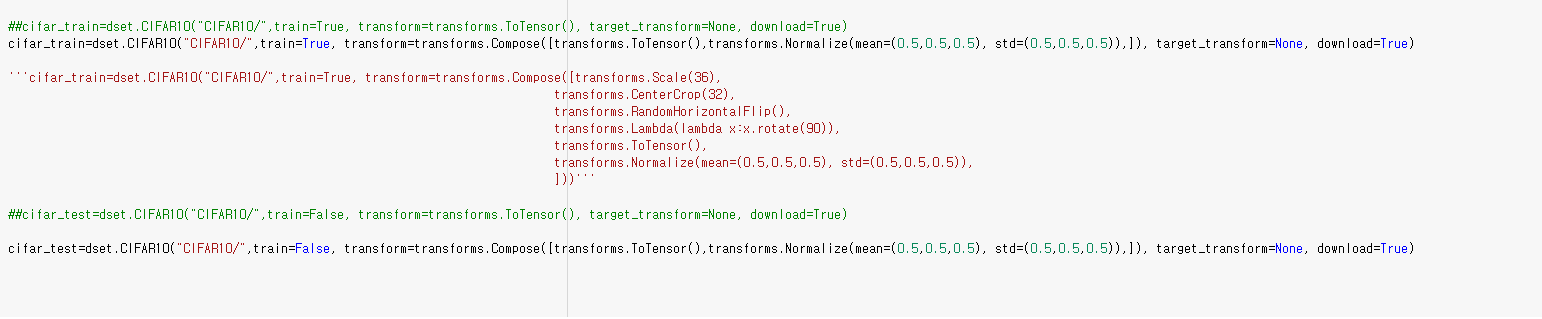
이것저것 하나씩 실험해 보던 내가 처음 높은 정확도 상승률을 본 방법은 Adam optimizer였다. 다른 기법 없이 58~63% 사이의 정확도를 보여주는 것을 보고 나는 Adam optimizer를 기본으로 다른 기법들을 추가 해 보기로 결심했다.

1. Dropout, batch normalization



Dropout의 경우 일부 노드들을 끄면서 진행하는 기법인데 내 epoch 수가 적어서 인지 오히려 정확도가 떨어지는 모습을 보여줬다. 때문에 향후 실험에 있어서 사용하지 않겠다는 생각을 가지게 되었다. Batch normalization의 경우 adam과 같이 사용하였을 때 꽤 유의미한 정확도 상승폭을 보여주었다. 이 시점에서 나는 Adam optimizer, batch normalization 두 가지 기법을 가지고 가기로 결심했다.

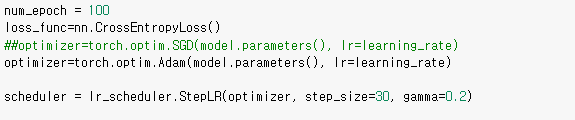
1. Data augmentation, Data Normalization



Data augmentation의 경우 큰 기대를 하고 추가했던 기법이었지만 dropout과 마찬가지로 처참한 정확도만을 기록하며 사용할 기술 후보에서 제외되게 되었다.

그에 반해 Data Normalization의 경우 1~2% 정도 꽤 괜찮은 성능 상승폭을 보이며 사용할 기술에 추가하게 되었다. 다만 정말 Data Normalization 의 힘인지 다른 요소에 의한 것인지, 오차 범위인지는 좀 헷갈린다는 생각이 들어 향후 실험에서 배제한 채로 실험을 할 수도 있다는 여지를 넘겨 두기로 했다.

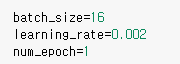
1. Learning rate decay



내가 좋은 조합이다 라고 생각한 기법을 가지고 (당시 Data normalization, batch normalization, weight initialization, adam optimizer 였다. Weight initialization의 경우 뺀 것과 넣은 것에서 유의미한 차이를 발견하지 못 하여 실험일지에선 누락하기로 결정했다.)

epoch수를 늘리기로 결정했다. Epoch의 수가 10,20…100까지 유의미한 성능 상승이 있었지만 일정 수를 넘어가자 별 차이가 없거나 정확도가 떨어지는 모습을 보였다. 이것이 수업 시간에 배운 과대 적합인가? 하는 생각이 들은 나는 100 이상의 epoch 값은 주지 않기로 하였다.

1. Batch 사이즈 조절



Batch size와 learning rate에 대한 식이 기억난 나는 두 수치를 만져 보았다. Learning rate의 경우 무슨 차이인지 잘 모르겠지만 batch size의 경우 실험의 속도가 확연히 빨라졌음을 느꼈다. 다만 성능이 감소하거나 큰 차이가 없어 초기값 16으로 놔두기로 하였다.

1. Activation function 교체, fully connected layer 증가



마지막으로 가장 큰 효과를 보았던 방법이었다. ReLU를 대신해 성능이 더 좋은 PReLU를 사용하고 Linear의 개수를 늘리자 가장 좋은 효과를 보였다. 이로써 내 최종 실험은 data normalization, batch normalization, weight initialization, learning rate decay, PReLU, Linear 늘리기, epoch 100 으로 결정 되었다.