DNN을 이용한 Cifar-100

201520991

곽명섭

Keras를 이용하여 Deep Neural Network의 Cifar-100의 accuracy를 높이는 과제를 이번에 해보았습니다. 우선 cifar 100이란 100개의 class가 존재하며 총 60000개의 이미지 데이터를 50000개는 training set, 나머지 10000개는 test set으로 사용하게 됩니다. 일단 저는 9월 10일 오후 실습시간에 하였던 deep neural network를 통하여 cifar – 10을 학습하는 코드를 수정하여 실시하였습니다.

맨처음으로 해주었던 것은 cifar-10을 불러오는 코드를 cifar-100 데이터셋을 불러오기 위해서 수정을 2군데정도 해주었습니다

Data\_func()함수의

(X\_train, y\_train), (X\_test, y\_test) = datasets.cifar100.load\_data()

으로 수정을 해주었고

number\_of\_class = 100

으로 수정을 해주었습니다. 이렇게 아무런 수정없이 cifar-100을 load만 해주는 과정으로 나온 accuracy는 대략 19%정도까지 올라갔습니다.

그리고 여기서 바로 다음에 해준 것은 hidden layer를 추가해주는 과정을 하였습니다

일단은 hidden layer을 많이 추가하면 더 좋을 것이라는 생각에 코드에 hidden layer을 아주 적게는 3~4개, 또 조금 늘려서 8개정도를 추가해서도 돌려보고 극단적으로 20개 까지도 추가해서 돌려보았습니다. 이 과정에서 activation 함수는 relu함수로 추가를 해주었습니다. 그런데 여기서는 딱히 18~22퍼 정도를 맴돌 뿐 큰 차이가 없었습니다. 그리고 혹시나 하여 수업시간에 배웠던 activation함수인 sigmoid함수를 써보았지만 accuracy는 0.09로 반 토막이 나버릴 뿐 큰 효과가 없다는 것을 깨닫고 다시 relu함수로 바꾸어 주었습니다. 밑의 사진이 원래 코드에서 hidden layer을 늘렸을 때의 accuray입니다. 가장 잘 나온 accuracy인데 이때는 hidden layer을 4개를 추가해주었을 때의 사진입니다. epoch값은 100으로 설정해주었습니다

스크린샷이(가) 표시된 사진

자동 생성된 설명

그리고 또 이제는 어떻게 하면 더 accuracy를 올릴 수 있을지 조사를 하다보니 activation function중에 elu라는 function이 있다는 것을 알게 되었습니다. 가장 최근에 나온 함수라는 글을 보고 elu함수에 대해서 이론적인 것을 더 조사를 해보았습니다.

f(x)=x if x>0

f(x)=α(ex−1) if x≤0

이것이 elu함수의 식이고 relu함수의 거의 모든 장점을 포괄하며 출력값이 zero-centered에 달하며 relu함수와는 다르게 exp함수를 계산하는 비용이 발생한다는 특징을 가집니다.

그래서 처음에는 우선 8개의 hidden layer에 4개는 relu함수를 써주었고 나머지 4개는 elu함수를 써주었습니다. 그리고 노드의 개수도 늘려주어 보았습니다. 노드의 개수는 원래 100 혹은 50으로 되어있던 것을 전부 10배로 1000, 500으로 설정해 보았습니다. 또한 dropout도 1~2개씩 넣어주었습니다. 그 결과 지금까지에 비해서 확연히 눈에 띄는 변화가 있었습니다. Epoch는 20으로 해준 상태에서 밑의 사진처럼 accuracy가 0.2577까지 올라가는 변화가 있었습니다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

그리고 이전에 해보았던 결과 같은 layer설정에서 epoch를 늘려주면 조금은 더 나은 결과가 나왔기에 epoch를 100으로 올려 주었습니다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

하지만 기대와는 달리 epoch값을 100으로 늘려도 큰 차이없이 26%정도의 accuracy정도로 상승할 뿐이었습니다. 여기에서 split을 늘리거나 줄이기도 해보고 batch size도 늘리고 해보아도 accuracy가 줄어들기만 할 뿐 늘어나지가 않아서 정체가 생기게 되었습니다. 그러다가 또 1퍼센트정도가 올라가게 된 계기가 모든 activation function을 elu로 바꿔주고 hidden layer을 12개까지 추가해주었더니 약간의 증가가 있었습니다.

스크린샷이(가) 표시된 사진

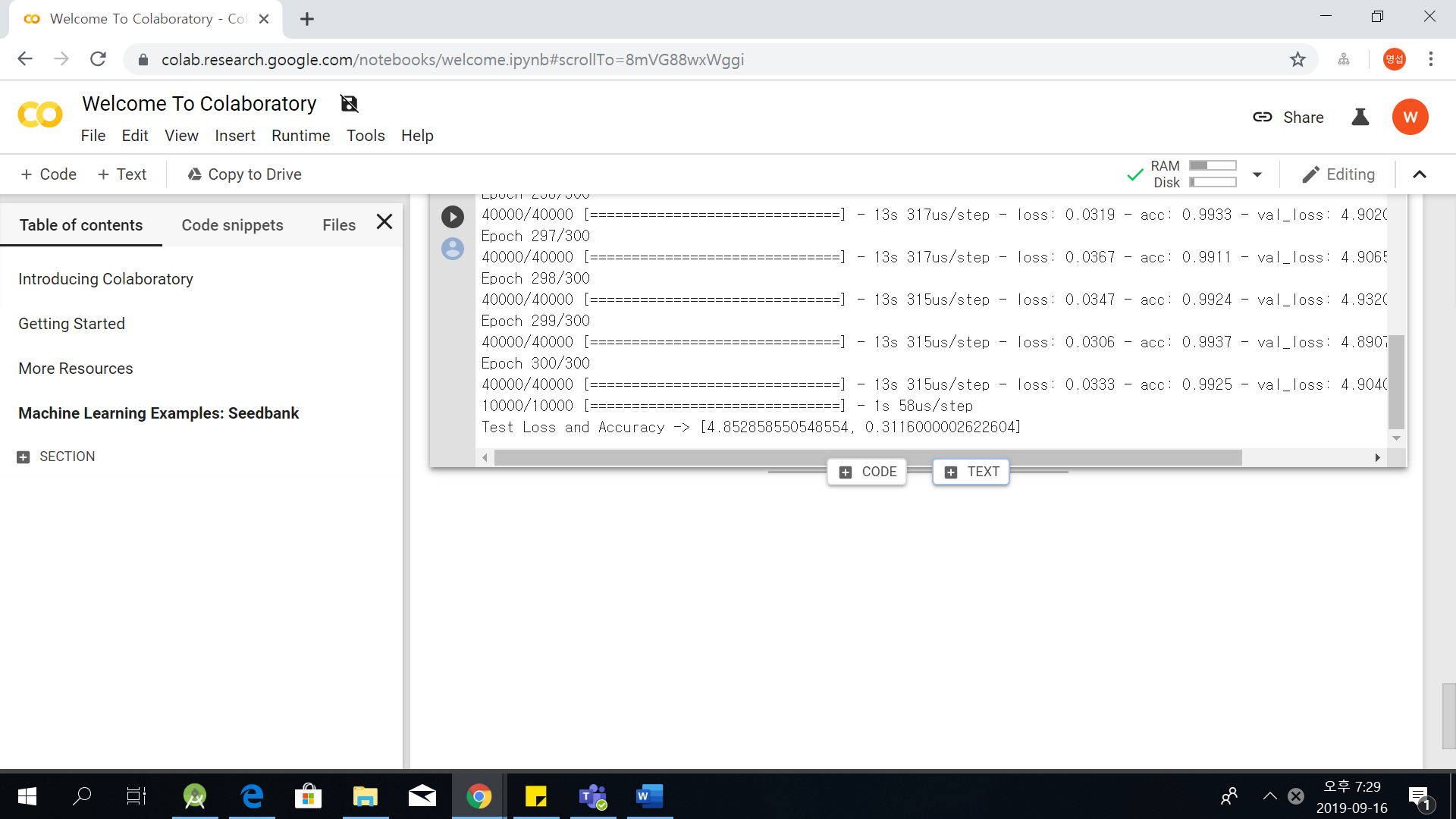
자동 생성된 설명

여기서 hidden layer을 약간 높인결과 확률이 1퍼센트가 올라간 것을 깨닫고 한번 극단적으로 더 hidden layer을 23개까지 늘려보았지만 accuracy는 턱없이 내려가기만 할 뿐이었습니다. 그래서 여기서 너무 accuracy가 올라가지않아서 이쯤하고 포기해야 하나 할 때 쯤 조교님께 메일을 보내어서 도움을 요청하게 되었는데 그때 조교님께서 optimizer을 adam으로 바꾸고 learning rate를 바꾸어 보라고 말 해주셔서 optimizer=Adam(lr=0.0001, beta\_1=0.9, beta\_2=0.999)로 learning rate를 많이 낮추고 여기에 제가 조사했던 마지막 hidden layer는 tanh로 설정을 해주라는 것과 hidden layer 수도 4개까지 줄여본 결과 여기에서 또 한번의 도약이 있었는데 이때 29~30까지 오르는 확연한 변화가 있었습니다.

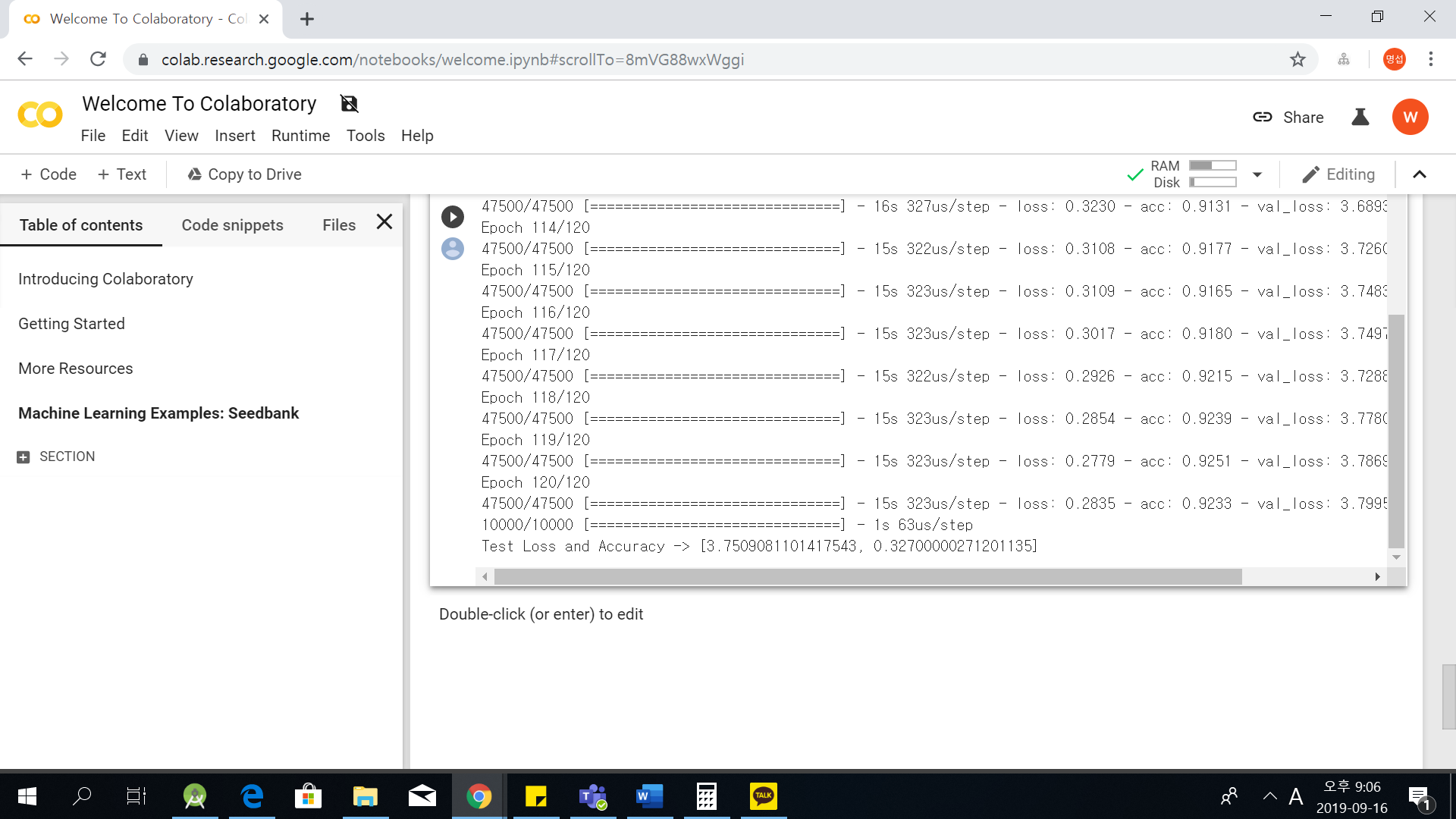
스크린샷이(가) 표시된 사진

자동 생성된 설명

여기에서 이제 무언가 새로운 것을 추가할 수 있는 것은 한계가 왔으며 제가 해볼수 있는 것은 이제 learning rate값과 epoch, batch size조절 그리고 layer의 갯수를 조절하는 정도일 뿐이었습니다. 여기서 learning rate값을 극단적으로 낮추고 epoch값을 300까지 올려본 결과 31%까지 올라가는 것을 알 수 있었습니다.

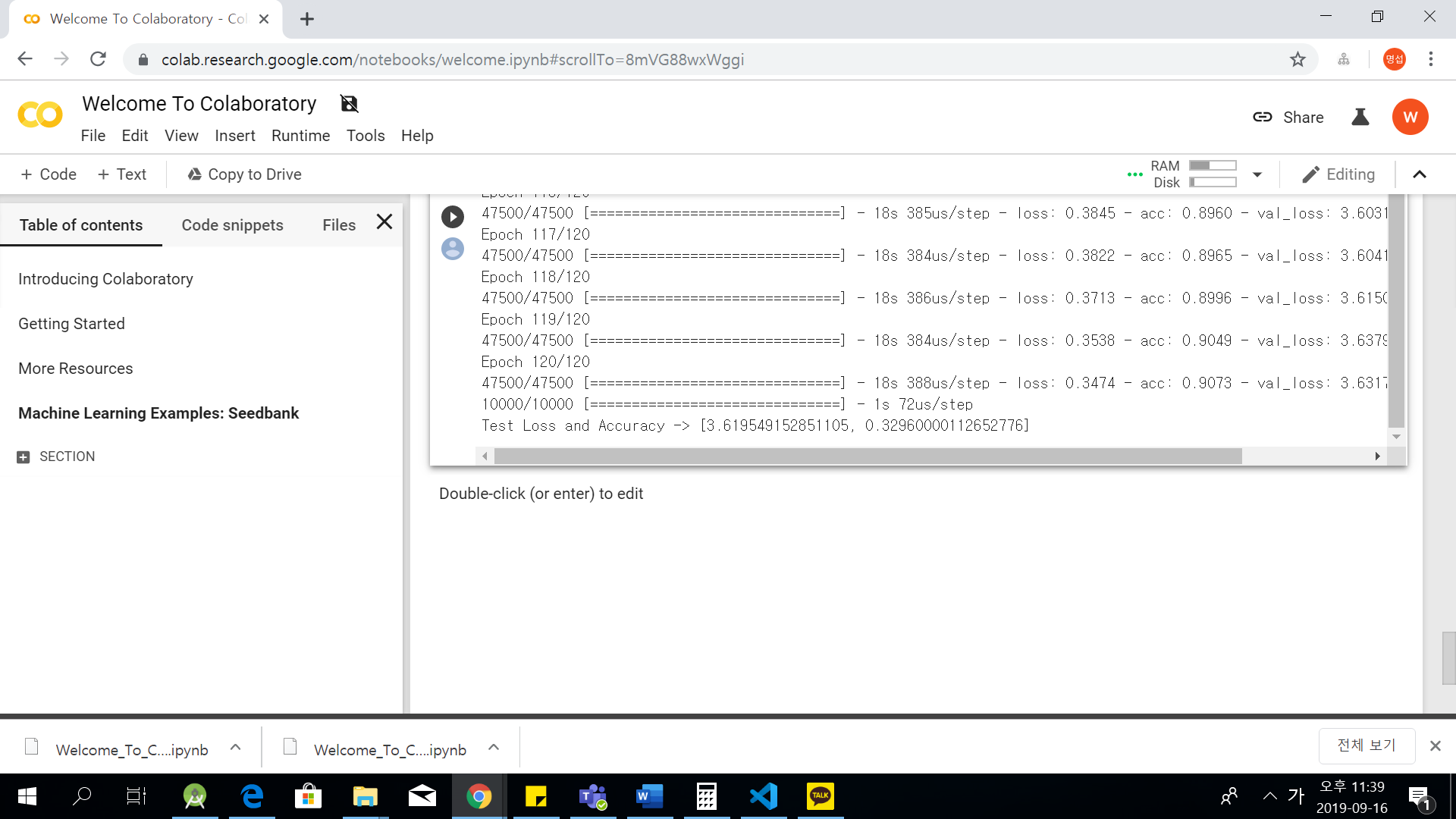


여기서 이제 끝을 낼까 하였지만 혹시나 아직 제출시간이 남아있기에 epoch, batch size, learning rate를 조절하며 accuracy를 높이기 위해서 최대한 많이 여러번의 시도를 해보았습니다.



그 결과 32.7%까지 올릴 수 있었습니다.

추가



32.96% 추가

느낀점

사실 이번 과제를 하면서 가장 크게 느낀점은 교수님이 항상 수업시간에 하시는 말이 인공지능 분야에서는 이게 왜 accuracy가 올랐는지는 모르지만 해보니 오르더라 라는 말씀을 자주 해주셨는데 이 이야기를 과제를 통해서 몸소 느껴 볼 수 있었던 것 같았습니다. 뿐만 아니라 왜 계속 시간을 많이 투자해야하는 과제라고 말씀하신 것 인지도 많이 와 닿았던 과제였던 것 같습니다. accuracy를 늘려가는 과정에서 왜 이것이 늘어나는지에 대해서 알기가 매우 어려웠고 이것저것 많이 바꾸다 보니 이유는 정확히 모르지만 확률이 늘어나는 경우가 매우 많았습니다. 그래서 보고서도 약간은 제가 했던 시행착오를 위주로 보고서를 작성하였습니다. 그래도 여러가지 1%라도 올리기 위해서 꾸준히 이것저것 바꾸기 위해서 이런저런 조사도 해보고 논문도 읽으며 확실히 조금씩은 올라가고 많이 배우게 된 과제였던 것 같습니다