**HW\_5**

2016707072 전자통신공학과 신민정

HW4에서 다룬 fashion-MNIST data를 가지고 model을 설계해 보았습니다.

HW4에서는 단순히 NN으로 설계했다면 이번과제에서는 image data에 맞게 Convolutional layer로 model을 구성해보았습니다.

**EarlyStopping, ModelCheckpoint, ReduceLRonPlateau**

학습 데이터가 너무 많다면 10 epcoh을 돌리는데에도 시간이 오래걸립니다. 학습시간을 고려하여 적은 epoch을 돌리기엔 최적의 model을 찾을 수 없기때문에, loss 또는 accuracy가 어느정도 수렴할 때 까지 backpropagation을 진행하며 weight를 업데이트 해주어야 합니다. Epoch을 크게 잡으면 학습시간이 너무 오래걸리며 overfitting에 빠질 위험이 있고, epoch을 작게 잡으면 underfitting이 된다는 단점이 있기 때문에 epcoh은 중요한 하이퍼파라미터입니다.

loss또는 accuracy가 어느정도 수렴했다면 학습을 중단하는 ***EarlyStopping***으로 이 문제를 해결할 수 있습니다.

또한 컴퓨팅 파워나 메모리의 문제로 학습이 중단되었을 때 ***ModelCheckpoint***을 사용하여 학습을 이어나갈 수 있습니다.

Objective function의 global minimum을 찾아나갈 때, initialization의 영향으로 local minimum에 빠질 위험이 있는데, ***ReduceLROnPlateau***을 사용하여 해결할 수 있습니다.

***ModelCheckpoint*** : 모델이 학습하면서 정의한 조건을 만족했을 때 Model의 weight 값을 중간 저장. 학습시간이 오래걸린다면, 모델이 개선된 validation score를 도출해낼 때마다 weight를 중간 저장함으로써, 혹시 중간에 memory overflow나 crash가 나더라도 다시 weight를 불러와서 학습을 이어나갈 수 있기 때문에, 시간을 save해 줄 수 있다.

***EarlyStopping*** : model의 성능 지표가 설정한 epoch동안 개선되지 않을 때 조기 종료

***ReduceLROnPlateau*** : Local Minima에 빠져 더이상 학습률이 개선되지 않고 정체되었을 때, learning rate를 늘리거나 줄여주는 방법으로 빠져나온다.

위의 기능은 Keras의 내장 라이브러리에서 import해올 수 있습니다.

지정해야하는 하이퍼파라미터에 대한 설명은 주석으로 달아놓았습니다.

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

다음과 같은 세개의 call back function은 모든 모델에 동일하게 적용하였습니다.

**Model\_1**

Convoultion layer와 FC만으로 model을 쌓아보았습니다. 3-Conv, 2-FC model입니다.

**스크린샷이(가) 표시된 사진

자동 생성된 설명**

**스크린샷이(가) 표시된 사진

자동 생성된 설명텍스트, 지도이(가) 표시된 사진

자동 생성된 설명**

**스크린샷, 조류이(가) 표시된 사진

자동 생성된 설명**

높은 test accuracy가 나왔지만, train accuracy와 차이가 큰 것으로 보아 overfitting이 일어났고, loss가 수렴하지 않고 발산하는 것으로 보아 Gradient Exploding문제가 일어났다고 판단을 하였습니다.

**Model\_2**

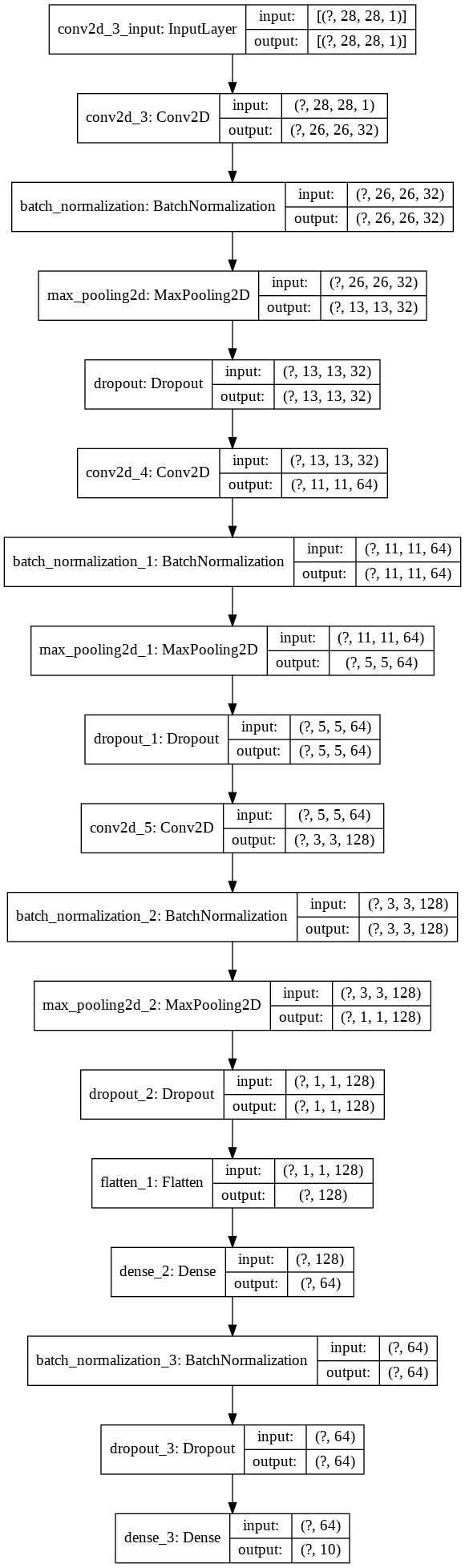
**Model\_1**과 같은 layer에 각각 Batch normalization과 Drop out을 추가하였습니다.

Batch Normalization은 Model 1에서 일어난 Gradient Exploding 문제를 막아주고,

Drop out은 Overfitting을 방지해줍니다.

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**텍스트, 지도이(가) 표시된 사진

자동 생성된 설명**

**스크린샷, 조류이(가) 표시된 사진

자동 생성된 설명**

Train accurac와 test accuracy가 거의 동일한 선상에서 수렴하는 것으로 보아 fit하게 학습된 것을 알 수 있고, loss가 일정값에 수렴하는 것을 보아 Gradient Vanishing/Exploding문제가 일어나지 않았음을 알 수 있습니다.

**Model\_param**

**Model\_2**와 같은 layer들에서 하이퍼파라미터 튜닝을 하기 위해 GridsearchCV를 하였습니다. **Model\_1과 Model\_2**를 비교한 결과, 단순히 Dense를 쌓는 것 보다 Batch Normalization과 Drop Out을 추가하는 것이 효과적임을 알 수 있었습니다.

GridsearchCV를 통해 Batch Normalization의 여부, Drop out의 정도, activation function의 종류, learning rate, opimizer의 종류, batch size를 바꾸어가며 비교해 보겠습니다.

조류이(가) 표시된 사진

자동 생성된 설명

스크린샷이(가) 표시된 사진

자동 생성된 설명

Test accuracy가 0.9038이 나왔다. Model\_2보다 accuracy가 낮은 이유는 학습의 속도를 위해 batch\_size를 지정해주었기 때문이다. 최적의 모델은 model\_2와 거의 동일함을 알 수 있다.

**최종결과**

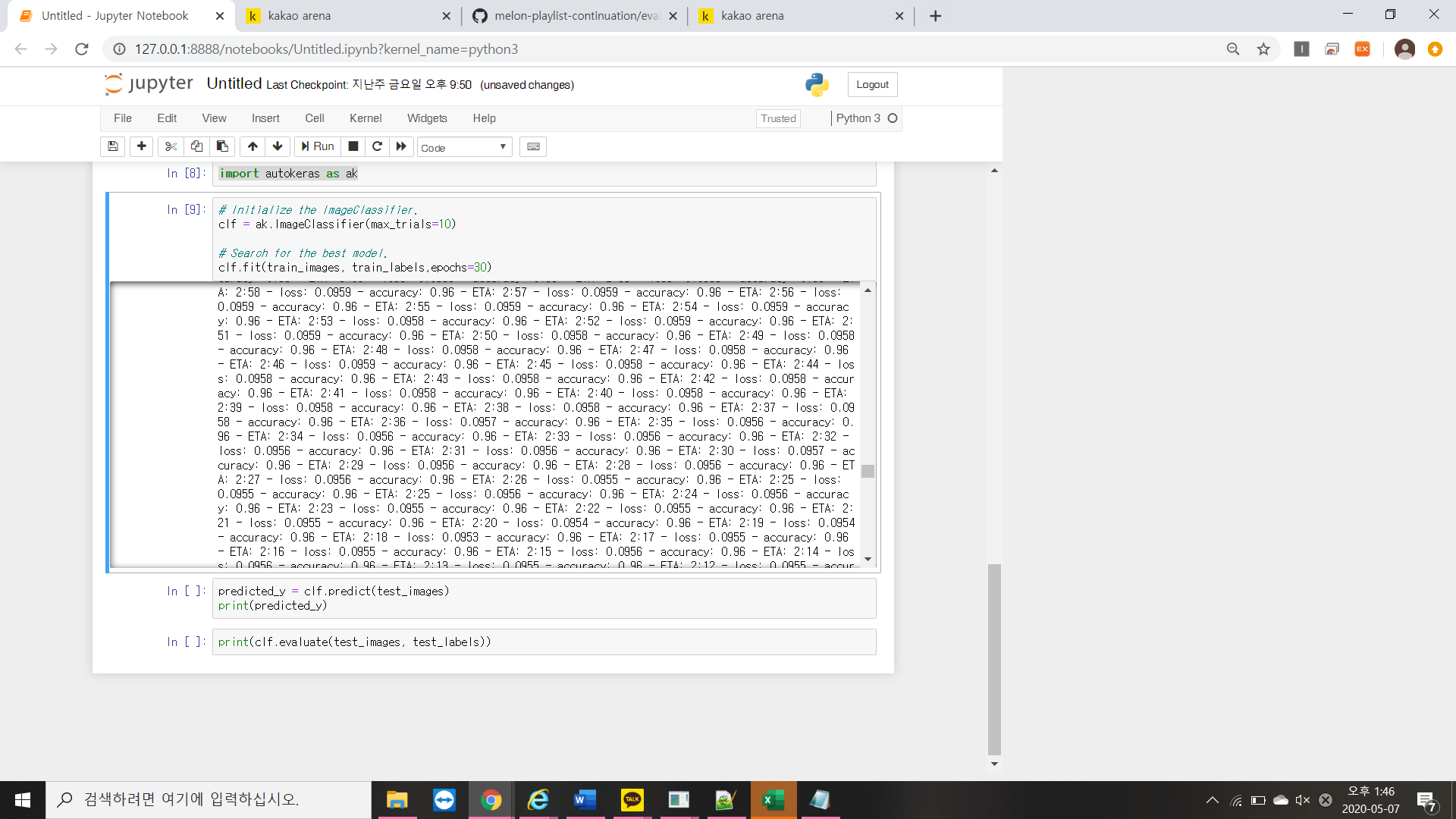
이번과제는 **HW\_4**에서 동일한 data를 사용였지만, 단순히 NN을 사용한 **HW\_4**와 달리 Convolution layer를 추가하여 image data의 feature를 추출하며 학습을 진행하였습니다.

NN을 사용한 **HW\_4**의 best accuracy(0.89)보다 높은 accuracy를 얻을 것으로 보아, image data에서는 FC보다 Conv layer가 효과적임을 알 수 있었습니다.

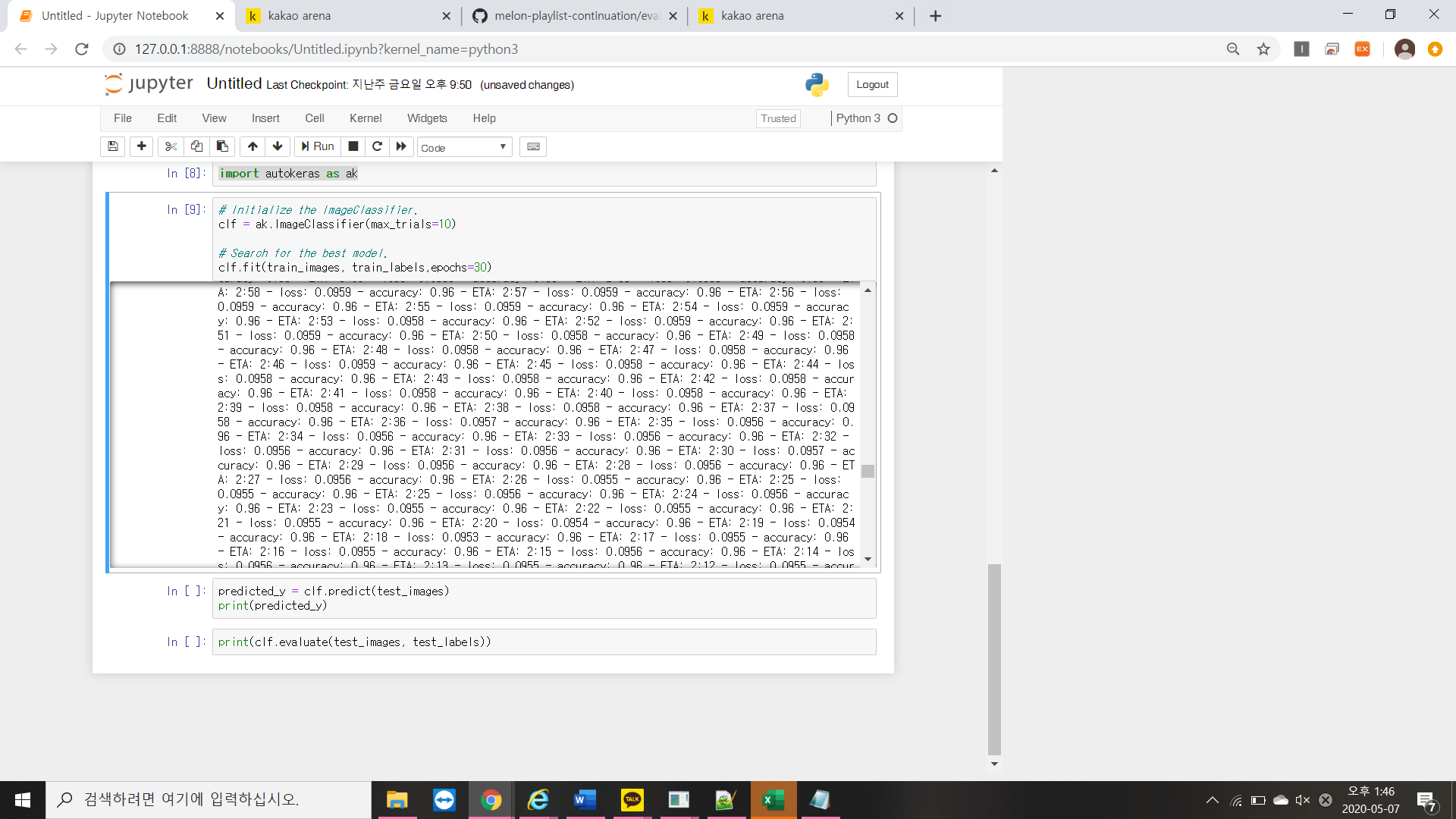
또한 단순히 layer를 쌓는 것 보다 Drop out과 Batch Normalization을 추가하는 것이 overfitting과 gradient vanishing/exploding을 방지할 수 있었습니다.

**HW\_4**에 대한 고찰 추가

(HW\_4제출 시 학습이 완료되지 않아 제출하지 못했던 AutoML에 관한 결과 및 고찰입니다. CPU 2core로 학습시 3일 이상 소요되어, 높은 acc를 얻었다고 판단하여 중간에 중단하였습니다.



Fit의 default로 지정된 validation\_split =0.25로train set을 split하여accuray를측정하였다.

 0.96이라는 높은 accuracy를 얻을 수 있었다.

학습시간과 컴퓨팅 파워가 충분하다면 AutoML로 학습해보는 것이 좋을 것 같다.