Google Developer Student Clubs

Week3 Presentation

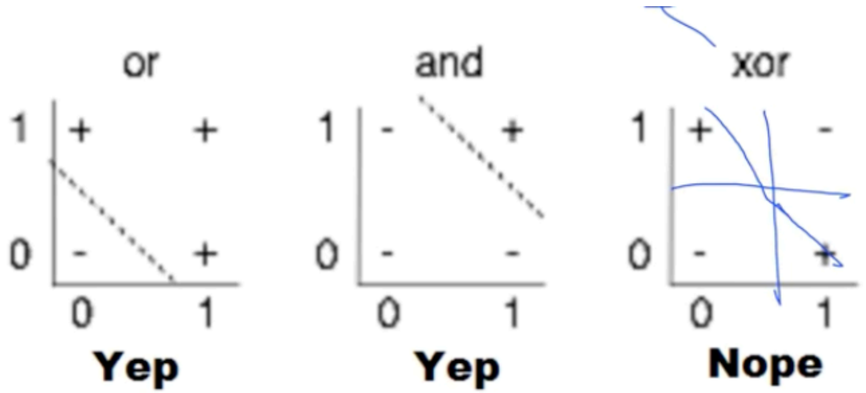
Neural Network, Techniques
Fashion MNIST Classifier

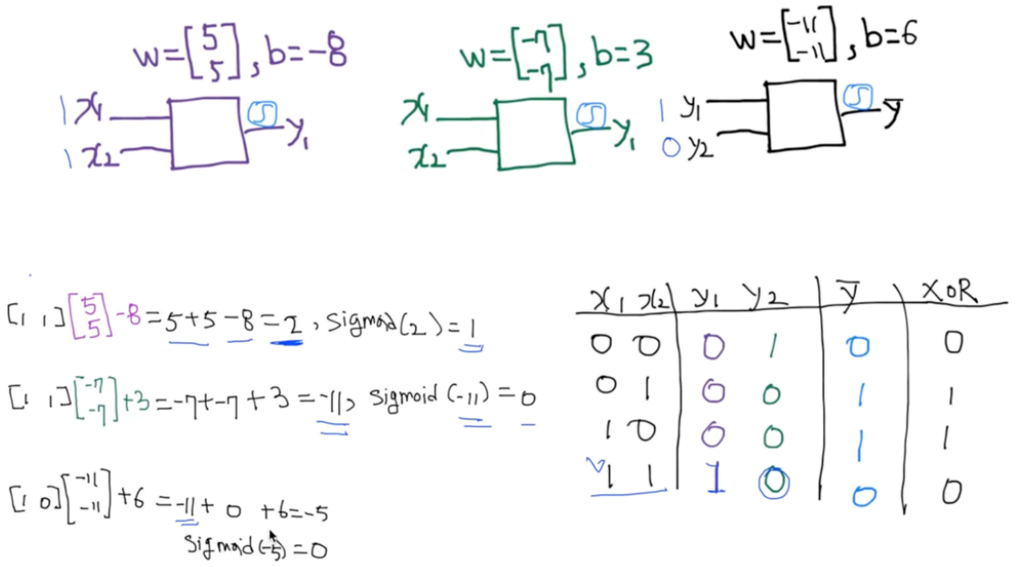
GDSC Hanyang ML/DL : Basic, Jaeseung Lee

Index

Neural Network

XOR Problem

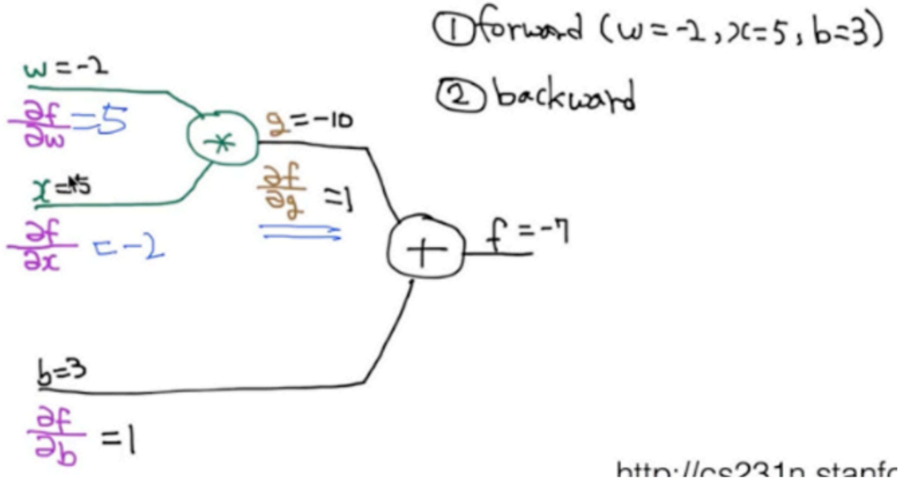


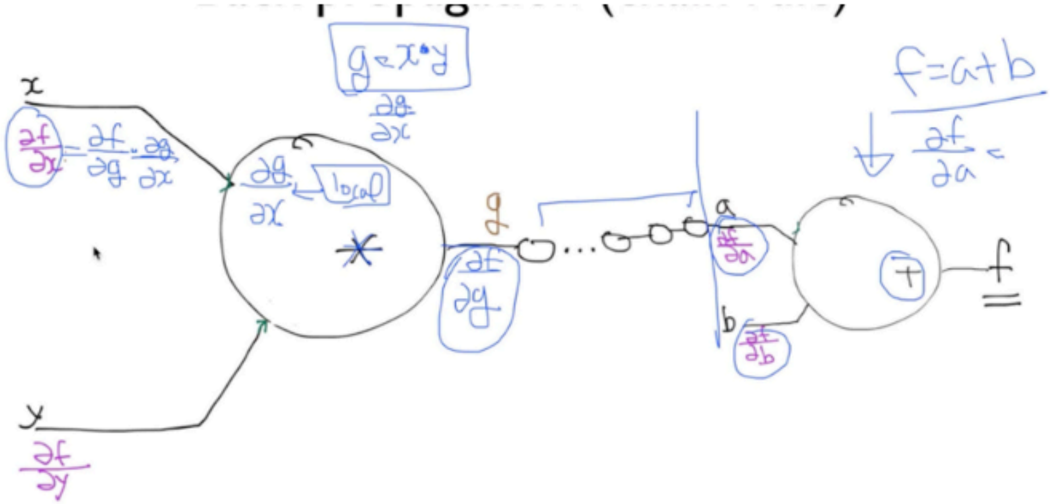


- Neural Network의 성장을 저해한 유명한 Problem by Marvin Minsky
- 한 개의 모델으로는 XOR을 예측할 수 없음. => 위 그림은 3개의 모델을 사용하여 해결.

Neural Network

Back Propagation

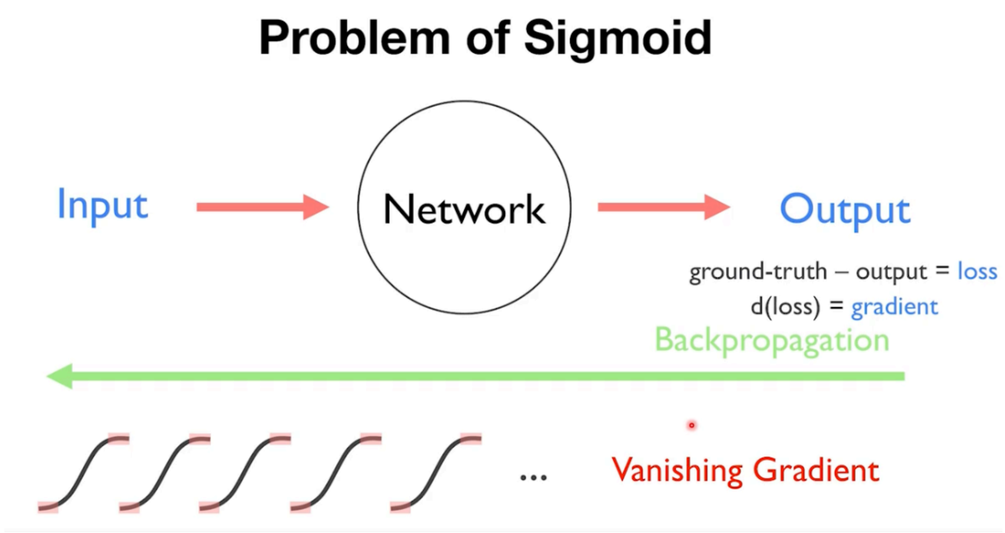


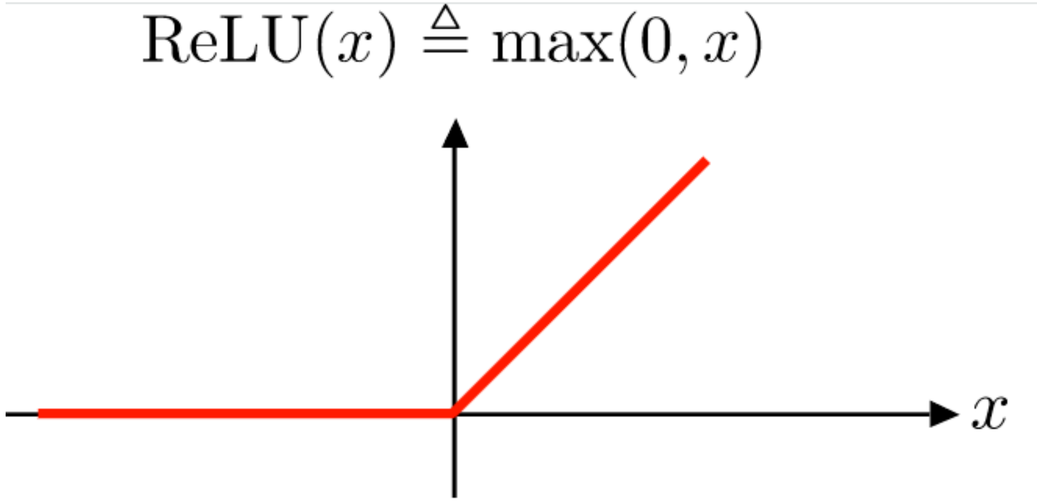


- Backpropagation은 복잡한 인공 신경망의 학습 불가 문제를 어느 정도 해결함.
- Chain Rule을 활용하여 앞에서 넘어오는 미분 값들을 활용하여 Weight 학습에 사용.
- 한계점이 존재하여 이를 보완하기 위한 여러 테크닉들이 등장하게 됨.

Techniques

ReLU

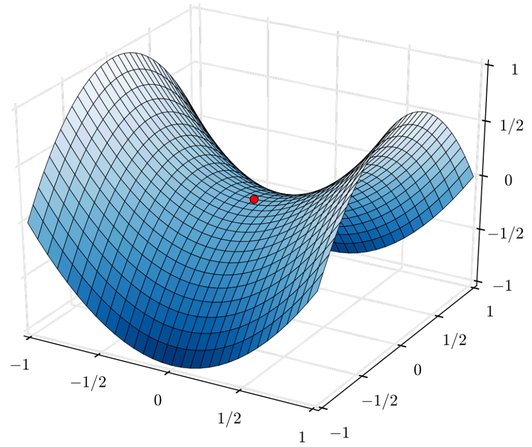




- ReLU : 양수 입력값의 기울기가 1, 음수, 0 입력값의 기울기는 0인 함수.
- Why ? : Sigmoid 함수는 양 극단의 Gradient가 매우 작아 Vanishing Gradient 발생.
- Leaky ReLU : 기존의 ReLU 음수값에서 Backpropagation 미전달 문제 해결

Techniques

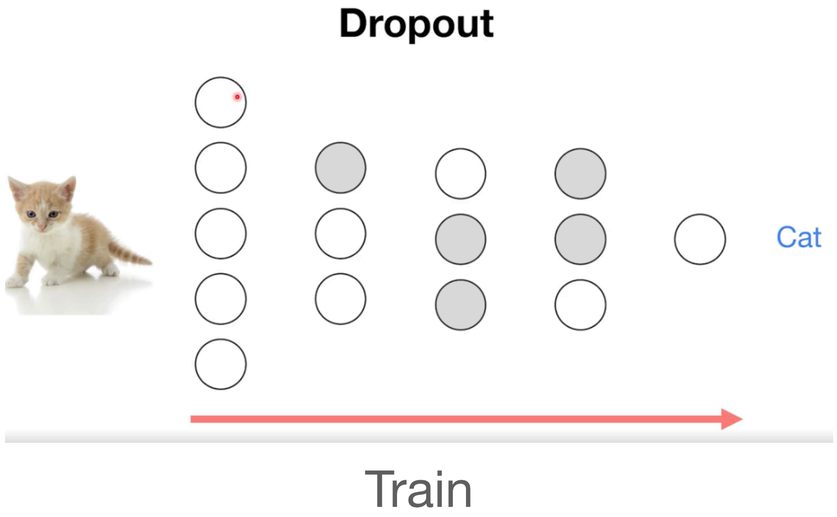
Weight Initialization

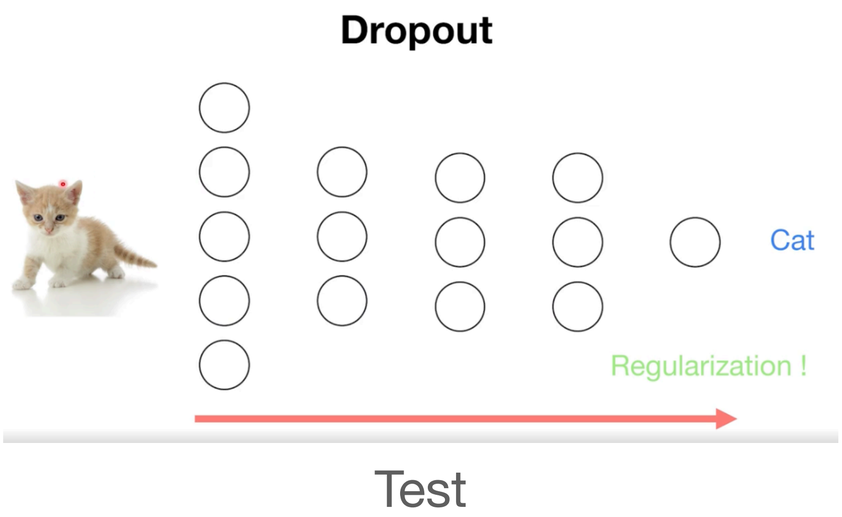


- 서론
  - 실제 Loss function은 복잡하여 Local minimum, Saddle point 등에 빠질 수 있음.
  - 따라서 적절한 초기값 세팅이 매우 중요함.
- Xavier Initialization (Glorot Initialization)
  - 평균은 0이며, 분산 = 2 / Channel_in + Channel_out. (노드 개수)
- He Initialization (for ReLU)
  - ReLU 함수에 특화되어 있으며, 분산 = 4 / Channel_in + Channel_out

Techniques

Dropout



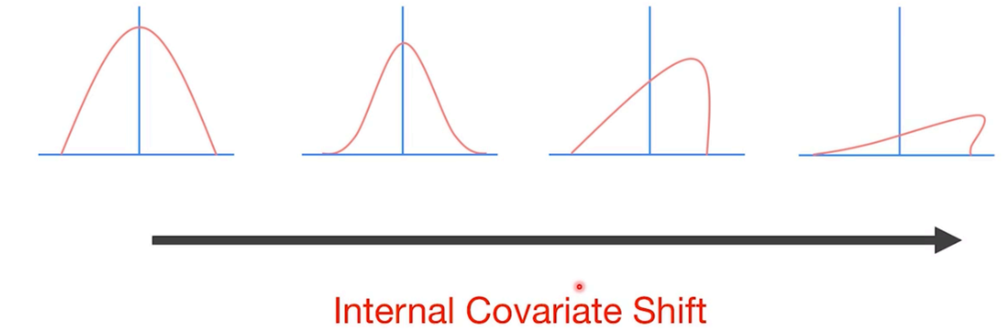


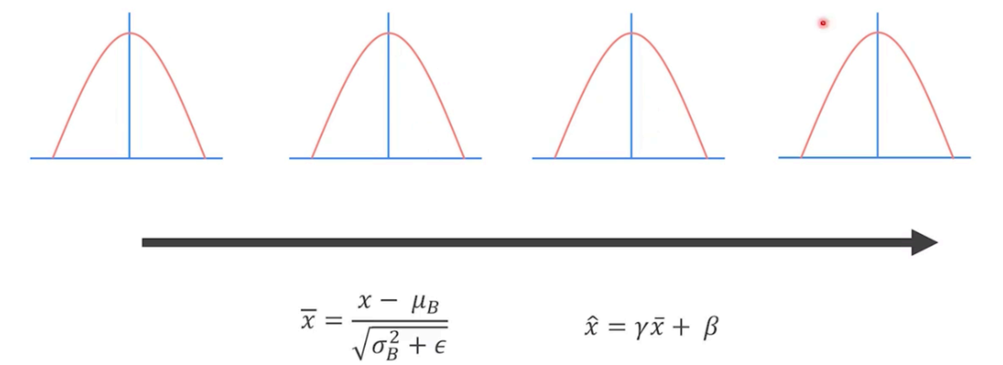
- Neural Network에서 Overfitting을 막기 위해 일부 Node를 끄고 학습하는 방법.
- 매 epoch 마다 다른 일부의 Node를 끄고, 테스트 시에는 전부 켜고 테스트

Techniques

Batch Normalization

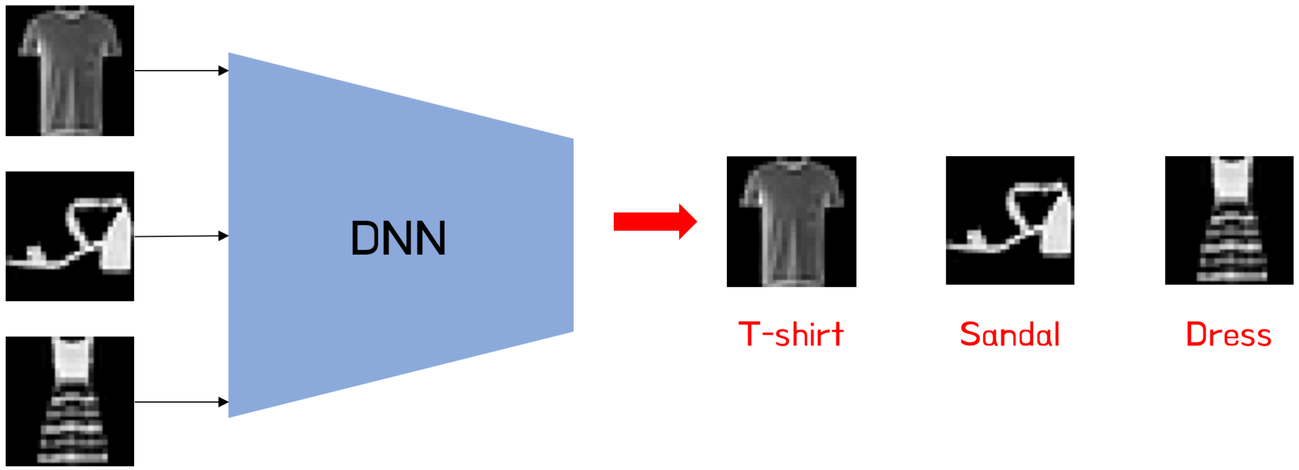
- Internal Covariate Shift
  - Neural Network Layer를 지날 수록, Distribution이 계속 변형되어 제대로 학습이 잘 안 되는 현상 발생.
- Batch Normalization
  - 이를 해결하기 위해 어떤 Term을 두어 Batch마다 Distribution이 변형되는 것을 막음.
- 모델의 순서
  - Layer -> Batch Norm -> Activation

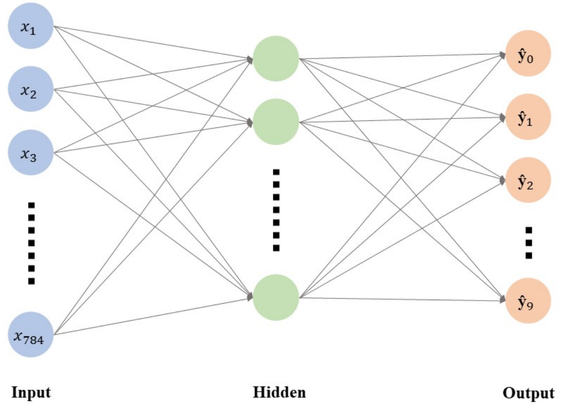




Fashion MNIST Classifier

Intro





Multilayer Perceptron (MLP)

- 옷 사진 Data를 DNN을 통해 분류하는 모델을 만드는 것이 목표
- 사진은 28x28x1 , 28x28 size이며 RGB가 아닌 밝기로만 이루어진 데이터
- Local에서 진행

Fashion MNIST Classifier

Code Review

- 데이터 셋 설정
  - tf.data.Dataset을 활용성 좋게 데이터를 셋
  - shuffle은 data를 섞어서 뽑음
    - 그래야 학습의 다양성이 증가.
  - 결과를 보면 784(28*28)으로 잘 들어갔음.
  - 맨 위에 보면 / 255로 normalization 됨.

```
# Load training and eval data from tf.keras
(train_data, train_labels), (test_data, test_labels) = \
    tf.keras.datasets.fashion_mnist.load_data()

train_data = train_data / 255.
train_data = train_data.reshape([-1, 28 * 28])
train_data = train_data.astype(np.float32)
train_labels = train_labels.astype(np.int32)

test_data = test_data / 255.
test_data = test_data.reshape([-1, 28 * 28])
test_data = test_data.astype(np.float32)
test_labels = test_labels.astype(np.int32)
```

```
# for train
N = len(train_data)

## 코드 시작 ##
train_dataset = tf.data.Dataset.from_tensor_slices((train_data, train_labels))
train_dataset = train_dataset.shuffle(N)
train_dataset = train_dataset.batch(batch_size)
train_dataset = train_dataset.repeat()
## 코드 종료 ##
```

```
# for test
## 코드 시작 ##
test_dataset = tf.data.Dataset.from_tensor_slices((test_data, test_labels))
test_dataset = test_dataset.batch(batch_size)
test_dataset = test_dataset.repeat()
## 코드 종료 ##
```

```
<_RepeatDataset element_spec=(TensorSpec(shape=(None, 784), dtype=tf.float32, name=None),
<_RepeatDataset element_spec=(TensorSpec(shape=(None, 784), dtype=tf.float32, name=None),
```

```
checker.train_dataset_check(train_dataset)
checker.test_dataset_check(test_dataset)
```

```
train_dataset을 잘 구현하셨습니다! 이어서 진행하셔도 좋습니다.
test_dataset을 잘 구현하셨습니다! 이어서 진행하셔도 좋습니다.
```

Fashion MNIST Classifier

Code Review

- 모델 구성 제작
  - tf.keras.Input을 통해 Input size 28*28 지정
  - 요구조건에 맞게 세팅
    - 첫번째 Layer 출력 feature 512
    - ReLU와 1st Layer 사이 Batch Normalization
    - 마지막 출력 Layer는 num_classes로 설정
    - 마지막 Layer에는 softmax 설정

```
model = tf.keras.Sequential() # Sequential 모델 생성
```

```
tf.keras.layers API 를 이용해 모델 코드를 작성해보세요! "## 코드 시작 ##"과 "## 코드 종료 ##
```

```
## 코드 시작 ##
model.add(tf.keras.Input(shape=(28*28,)))
model.add(tf.keras.layers.Dense(units=512))
model.add(tf.keras.layers.BatchNormalization())
model.add(tf.keras.layers.ReLU())
model.add(tf.keras.layers.Dense(units=num_classes, activation='softmax'))
## 코드 종료 ##
```

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_2 (Dense)             (None, 512)               401920

 batch_normalization_1 (Bat  (None, 512)               2048
 chNormalization)

 re_lu_1 (ReLU)              (None, 512)               0

 dense_3 (Dense)             (None, 10)                5130

=================================================================
Total params: 409098 (1.56 MB)
Trainable params: 408074 (1.56 MB)
Non-trainable params: 1024 (4.00 KB)
_________________________________________________________________
```

```
checker.model_check(model)
```

```
네트워크를 잘 구현하셨습니다! 이어서 진행하셔도 좋습니다.
```

Fashion MNIST Classifier

Code Review

- 모델 컴파일 및 학습, 테스트
  - 요구조건에 맞게 세팅
    - optimizer : adam optimizer 사용
      - learning_rate 0.001에서 0.01으로 변경
    - loss : sparse categorical cross entropy
      - Sparse : one-hot encoded label
    - Metrics : accuracy (모델 성능 평가에 쓰임)

```
# model compile with optimizer, loss, metrics

## 코드 시작 ##
model.compile(optimizer = tf.keras.optimizers.legacy.Adam(learning_rate=learning_rate),
              loss = 'sparse_categorical_crossentropy',
              metrics = ['accuracy'])
## 코드 종료 ##
```

```
checker.compile_check(model)
```

```
compile을 잘 정의하셨습니다! 이어서 진행하셔도 좋습니다.
```

Fashion MNIST Classifier

Code Review

- 모델 학습 및 테스트
  - 학습
    - model.fit을 활용
      - steps_per_epoch : 몇 번 batch 학습 시킬 것인지
      - Batch size : 한번에 데이터 얼마만큼 읽어들일 것인지
    - Epoch 기존 5에서 30으로 변경.
  - 테스트
    - Model acc, test acc 기준 통과

```
## 코드 시작 ##
model.fit(train_dataset, steps_per_epoch = len(train_data) / batch_size, epochs=max_epochs)
## 코드 종료 ##
```

```
Epoch 1/30
468/468 [==============================] - 1s 3ms/step - loss: 0.0900 - accuracy: 0.9670
```

```
checker.accuracy_check(model)
```

```
fit을 잘 정의하셨습니다! 이어서 진행하셔도 좋습니다.
```

```
loss,accuracy = model.evaluate(test_dataset, steps = len(test_data)//batch_size)
print('test loss is {}'.format(loss))
print('test accuracy is {}'.format(accuracy))
```

```
78/78 [==============================] - 0s 1ms/step - loss: 0.8617 - accuracy: 0.8783
test loss is 0.8616554141044617
test accuracy is 0.8783053159713745
```

```
checker.test_check(model)
```

```
모델 성능이 기준치를 넘었습니다! 이어서 진행하셔도 좋습니다.
```

Fashion MNIST Classifier

Code Review

- 결과
  - 테스트 셋 25개 중 23개 맞춤. (92% acc)
  - 모든 기준 통과

```
import check_util.submit as submit
submit.process_submit()
```

```
[ Self-Check ] 시스템: Darwin
[ Self-Check ] Submit 파일 생성완료! 위치: 'submit'
[ Self-Check ] submit.zip 생성 완료!
[ Self-Check ] 모든 평가기준을 통과했습니다. 압축파일을 제출해주세요!
```

Fashion MNIST Classifier

Problems

- 문제점 원인
  - Python 3.6 / Tensorflow 2.0.0 버전에서 구동해야 함
  - 이 과제 한 개를 위해 현재 세팅을 지우고 다시 설치해야 함을 우회하고자 했음
- Metrics (https://www.boostcourse.org/ai212/forum/23364)
  - metrics는 model.fit 이후에 설정되는 값이라 fit을 돌린 이후 checker 검사
- Model Accuracy (https://www.boostcourse.org/ai212/forum/25094)
  - 버전 문제로 인해 2.0.0에서는 model.metrics[1]에 acc가 들어있음.

Google Developer Student Clubs

Thank you for listening

Neural Network, Techniques
Fashion MNIST Classifier

GDSC Hanyang ML/DL : Basic, Jaeseung Lee