2. CNN 문제

1번째 문제는 CNN 채널을 모두 4배 늘려보는 문제인데 모르겠습니다.

2번째 문제는 화소 데이터 [-1, 1] 정규화를 생략해 보는 문제입니다. 저는 데이터 스케일링 파트를 없애보았습니다. 그랬더니 최종 정확도는 생략하기 전이랑 별차이가 없었지만 최종 손실도가 더 안좋아졌습니다.

### Keras CNN으로 패션 아이템 구분하기

1. 패키지 수입 및 파라미터

# 패키지 수입

import matplotlib.pyplot as plt

import numpy as np

from time import time

from keras.datasets import fashion\_mnist

from keras.models import Sequential

from keras.utils import np\_utils

from keras.layers import Flatten

from keras.layers import Dense, MaxPool2D

from keras.layers import Conv2D, InputLayer

# 파라미터

MY\_EPOCH = 10

MY\_BATCH = 500

2. 데이터 처리

# 데이터 불러오기

(X\_train, Y\_train), (X\_test, Y\_test) = fashion\_mnist.load\_data()

print('학습용 입력 데이터:', X\_train.shape)

print('학습용 출력 데이터:', Y\_train.shape)

print('평가용 입력 데이터:', X\_test.shape)

print('평가용 출력 데이터:', Y\_test.shape)

# 데이터 샘플 출력

print('학습용 데이터 첫번째 화소 정보')

print(X\_train[0])

plt.figure(figsize=(10, 10))

plt.imshow(X\_train[0], cmap='gray')

print('학습용 데이터 첫번째 이미지의 라벨:',Y\_train[0])

# 데이터 스케일링

# X\_train = X\_train/255.0

# X\_test = X\_test/255.0

# 이미지 채널 정보 추가

X\_train = X\_train.reshape(60000, 28, 28, 1)

X\_test = X\_test.reshape(10000, 28, 28, 1)

# 라벨 정보 수정

Y\_train = np\_utils.to\_categorical(Y\_train, 10)

print('학습용 데이터 첫번째 이미지의 라벨:', Y\_train[0])

Y\_test = np\_utils.to\_categorical(Y\_test, 10)

print('학습용 입력 데이터:', X\_train.shape)

print('학습용 출력 데이터:', Y\_train.shape)

print('평가용 입력 데이터:', X\_test.shape)

print('평가용 출력 데이터:', Y\_test.shape)

3. 인공신경망 구현

# CNN 구현

model = Sequential()

model.add(InputLayer(input\_shape=(28, 28, 1)))

# 첫번째 합성곱 블럭

model.add(Conv2D(filters=32,

kernel\_size=2,

padding='same',

activation='relu'))

model.add(MaxPool2D(pool\_size=2))

# 두번째 합성곱 블럭

model.add(Conv2D(filters=64,

kernel\_size=2,

padding='same',

activation='relu'))

model.add(MaxPool2D(pool\_size=2))

# DNN 입성

model.add(Flatten())

model.add(Dense(units=128,

activation='relu'))

model.add(Dense(units=10,

activation='softmax'))

print('CNN 요약')

model.summary()

4. 인공신경망 학습

# 학습 방식 설정

model.compile(optimizer='adam',

loss='categorical\_crossentropy',

metrics=['acc'])

# CNN 학습

print('학습 시작')

begin = time()

model.fit(x=X\_train,

y=Y\_train,

epochs=MY\_EPOCH,

batch\_size=MY\_BATCH,

verbose=0)

end = time()

print('총 학습 시간: {:.2f}초'.format(end-begin))

5. CNN 평가

# CNN 평가

score = model.evaluate(x=X\_test,

y=Y\_test,

verbose=1)

print('최종 손실값: {:.2f}'.format(score[0]))

print('최종 정확도: {:.2f}'.format(score[1]))

6. CNN 예측

# 샘플 이미지 지정

image = X\_test[1234]

print(image.shape)

# batch 정보 추가

image = image.reshape(1, 28, 28, 1)

pred = model.predict(image)

# 정보 출력

print('CNN 예측 값:', pred)

print('정답:', Y\_test[0])

plt.figure(figsize=(5, 5))

tmp = X\_train[1234].reshape(28, 28)

tmp = tmp \* 255

plt.imshow(tmp, cmap='gray')

3번째 문제는 정확도를 최대화하는 optimizer와 loss 함수를 찾는 문제입니다. Optimizer는 adam이고 loss는 categorical\_crossentropy입니다.

4번째는 정확도 개선하는 문제입니다. 저는 원래대로가 최선이라 생각이듭니다….

### Keras CNN으로 패션 아이템 구분하기

1. 패키지 수입 및 파라미터

# 패키지 수입

import matplotlib.pyplot as plt

import numpy as np

from time import time

from keras.datasets import fashion\_mnist

from keras.models import Sequential

from keras.utils import np\_utils

from keras.layers import Flatten

from keras.layers import Dense, MaxPool2D

from keras.layers import Conv2D, InputLayer

# 파라미터

MY\_EPOCH = 10

MY\_BATCH = 500

2. 데이터 처리

# 데이터 불러오기

(X\_train, Y\_train), (X\_test, Y\_test) = fashion\_mnist.load\_data()

print('학습용 입력 데이터:', X\_train.shape)

print('학습용 출력 데이터:', Y\_train.shape)

print('평가용 입력 데이터:', X\_test.shape)

print('평가용 출력 데이터:', Y\_test.shape)

# 데이터 샘플 출력

print('학습용 데이터 첫번째 화소 정보')

print(X\_train[0])

plt.figure(figsize=(10, 10))

plt.imshow(X\_train[0], cmap='gray')

print('학습용 데이터 첫번째 이미지의 라벨:',Y\_train[0])

# 데이터 스케일링

X\_train = X\_train/255.0

X\_test = X\_test/255.0

# 이미지 채널 정보 추가

X\_train = X\_train.reshape(60000, 28, 28, 1)

X\_test = X\_test.reshape(10000, 28, 28, 1)

# 라벨 정보 수정

Y\_train = np\_utils.to\_categorical(Y\_train, 10)

print('학습용 데이터 첫번째 이미지의 라벨:', Y\_train[0])

Y\_test = np\_utils.to\_categorical(Y\_test, 10)

print('학습용 입력 데이터:', X\_train.shape)

print('학습용 출력 데이터:', Y\_train.shape)

print('평가용 입력 데이터:', X\_test.shape)

print('평가용 출력 데이터:', Y\_test.shape)

3. 인공신경망 구현

# CNN 구현

model = Sequential()

model.add(InputLayer(input\_shape=(28, 28, 1)))

# 첫번째 합성곱 블럭

model.add(Conv2D(filters=32,

kernel\_size=2,

padding='same',

activation='relu'))

model.add(MaxPool2D(pool\_size=2))

# 두번째 합성곱 블럭

model.add(Conv2D(filters=64,

kernel\_size=2,

padding='same',

activation='relu'))

model.add(MaxPool2D(pool\_size=2))

# DNN 입성

model.add(Flatten())

model.add(Dense(units=128,

activation='relu'))

model.add(Dense(units=10,

activation='softmax'))

print('CNN 요약')

model.summary()

4. 인공신경망 학습

# 학습 방식 설정

model.compile(optimizer='adam',

loss='categorical\_crossentropy',

metrics=['acc'])

# CNN 학습

print('학습 시작')

begin = time()

model.fit(x=X\_train,

y=Y\_train,

epochs=MY\_EPOCH,

batch\_size=MY\_BATCH,

verbose=0)

end = time()

print('총 학습 시간: {:.2f}초'.format(end-begin))

5. CNN 평가

# CNN 평가

score = model.evaluate(x=X\_test,

y=Y\_test,

verbose=1)

print('최종 손실값: {:.2f}'.format(score[0]))

print('최종 정확도: {:.2f}'.format(score[1]))

6. CNN 예측

# 샘플 이미지 지정

image = X\_test[1234]

print(image.shape)

# batch 정보 추가

image = image.reshape(1, 28, 28, 1)

pred = model.predict(image)

# 정보 출력

print('CNN 예측 값:', pred)

print('정답:', Y\_test[0])

plt.figure(figsize=(5, 5))

tmp = X\_train[1234].reshape(28, 28)

tmp = tmp \* 255

plt.imshow(tmp, cmap='gray')