2018 FALL
FINANCIAL ENGINEERING
FINAL PROJECT

Doodle Recognition Challenge

with Convolutional Neural Network

Start!

2조
박수경 이하경 임지연

목차

Contents Map

2조
박수경 이하경 임지연

Part 1

데이터 선별 및 탐색

Data Exploration

데이터 선별 및 탐색

Doodle Recognition Challenge

kaggle

데이터 선별 및 탐색

Quick, Draw! 게임 소개

https://quickdraw.withgoogle.com/

데이터 선별 및 탐색

팀원들의 게임 참여 결과

Draw by SK

신경망이 낙서 5개를 맞췄습니다.
하지만 1개는 알아보지 못했어요.
낙서를 선택하여 신경망이 무엇으로 인식했는지 알아보세요.


✓ 액자


✓ 백조


✓ 자동차


✕ 샌들


✓ 낙하산


✓ 양초

Draw by JY

신경망이 낙서 3개를 맞췄습니다.
하지만 3개는 알아보지 못했어요.
낙서를 선택하여 신경망이 무엇으로 인식했는지 알아보세요.


✓ 안경


✓ 나비


✕ 욕조


✕ 원숭이


✓ 베개


✕ 배낭

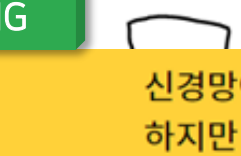
데이터 선별 및 탐색

팀원들의 게임 참여 결과

Draw by SK

신경망이 낙서 5개를 맞췄습니다.
하지만 1개는 알아보지 못했어요.
낙서를 선택하여 신경망이 무엇으로 인식했는지 알아보세요.


✓ 액자

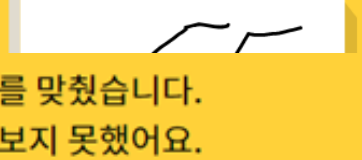



✕ 샌들


✓ 낙

Draw by JY

신경망이 낙서 3개를 맞췄습니다.
하지만 3개는 알아보지 못했어요.
낙서를 선택하여 신경망이 무엇으로 인식했는지 알아보세요.




나비


✕ 욕조


베개


✕ 배낭

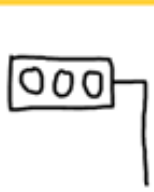
Draw by HG

신경망이 낙서 3개를 맞췄습니다.
하지만 3개는 알아보지 못했어요.
낙서를 선택하여 신경망이 무엇으로 인식했는지 알아보세요.


✓ 교회


✓ 토끼


✕ 손톱


✓ 신호등


✕ 성냥개비


✕ 전화

데이터 선별 및 탐색

데이터 불러들이기

[1]
```
# 필요한 패키지 다운로드
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt

import os
import glob
import keras

from tensorflow.keras import layers
from sklearn.model_selection import train_test_split
from sklearn.utils import shuffle
```

[2]
```
classes = ['guitar', 'fish', 'laptop', 'pencil', 'scissors',
           'sheep', 'soccer_ball', 'teddy-bear', 'The_Eiffel_Tower', 'The_Mona_Lisa']

!mkdir dataset    # make directory

import urllib.request

def download():

  base = 'https://storage.googleapis.com/quickdraw_dataset/full/numpy_bitmap/'

  for c in classes:
    class_url = c.replace('_', '%20')
    path = base + class_url + '.npy'

    print(path)
    urllib.request.urlretrieve(path, 'dataset/' + c + '.npy')
```

[3]
```
download() # web data download
```

```
https://storage.googleapis.com/quickdraw_dataset/full/numpy_bitmap/guitar.npy
https://storage.googleapis.com/quickdraw_dataset/full/numpy_bitmap/fish.npy
https://storage.googleapis.com/quickdraw_dataset/full/numpy_bitmap/laptop.npy
https://storage.googleapis.com/quickdraw_dataset/full/numpy_bitmap/pencil.npy
https://storage.googleapis.com/quickdraw_dataset/full/numpy_bitmap/scissors.npy
https://storage.googleapis.com/quickdraw_dataset/full/numpy_bitmap/sheep.npy
https://storage.googleapis.com/quickdraw_dataset/full/numpy_bitmap/soccer%20ball.npy
https://storage.googleapis.com/quickdraw_dataset/full/numpy_bitmap/teddy-bear.npy
https://storage.googleapis.com/quickdraw_dataset/full/numpy_bitmap/The%20Eiffel%20Tower.npy
https://storage.googleapis.com/quickdraw_dataset/full/numpy_bitmap/The%20Mona%20Lisa.npy
```

데이터 선별 및 탐색

데이터셋 만들기

[5]
```
def load_data(root, max_items_per_class = 10000): #데이터 10000 개사용
  all_files = glob.glob(os.path.join(root, '*.npy'))

  # initialize variables
  x = np.empty([0, 784])
  y = np.empty([0])
  class_names = []

  # load each data file
  for idx, file in enumerate(all_files):
    data = np.load(file)
    data = data[0:max_items_per_class, :]
    labels = np.full(data.shape[0], idx)

    x = np.concatenate((x, data), axis = 0)
    y = np.append(y, labels)

    class_name, ext = os.path.splitext(os.path.basename(file))
    class_names.append(class_name)

  data = None
  labels = None

  # randomize the dataset
  x, y = shuffle(x, y, random_state = 100)

  # train(80%), test(20%) split
  x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 100)

  return x_train, x_test, y_train, y_test, class_names
```

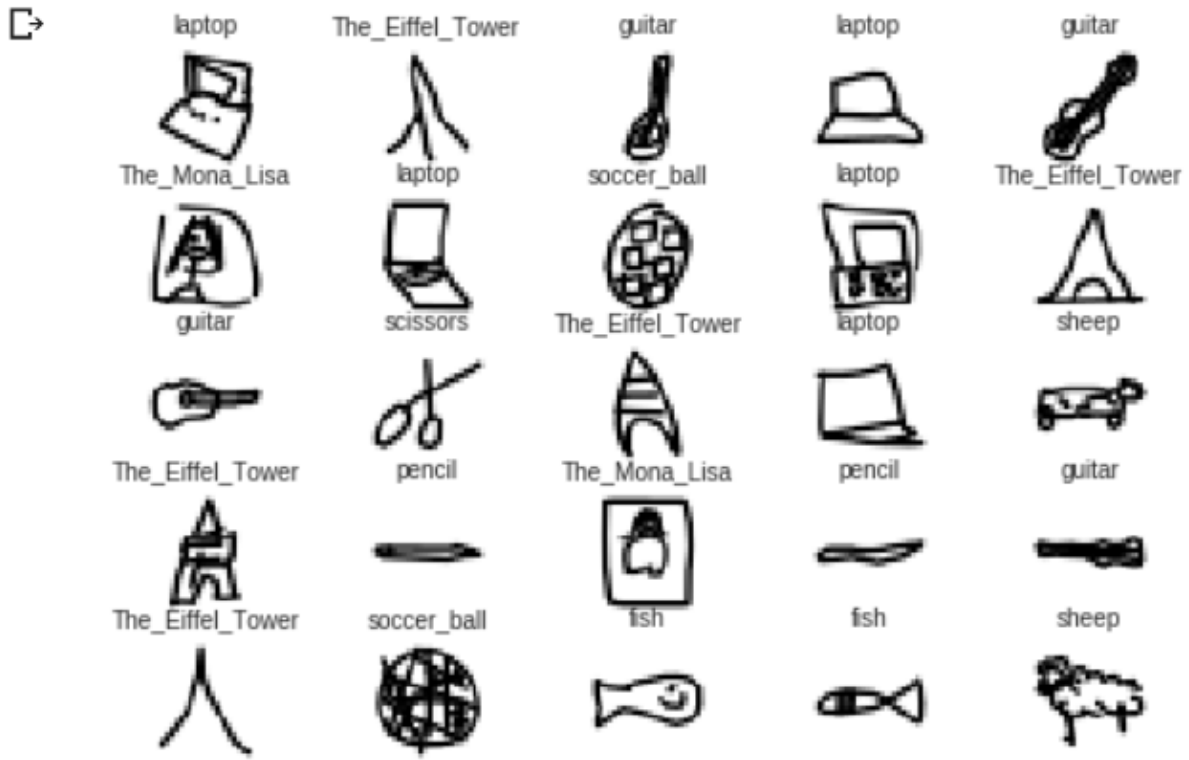
[6]
```
x_train, x_test, y_train, y_test, class_names = load_data('dataset')
n_classes = len(class_names) # 10 classes
image_size = 28

print(x_train.shape, y_train.shape, x_test.shape, y_test.shape, n_classes)
```

```
(80000, 784) (80000,) (20000, 784) (20000,) 10
```

데이터 선별 및 탐색

데이터셋 확인

```
# 사용할 데이터가 어떻게 생겼을까?
randomid = np.random.randint(0, len(x_train), size = 25)
fig = plt.figure()
for i in range(25):
  plt.subplot(5, 5, i +1)
  plt.axis('off')
  plt.imshow(x_train[randomid[i]].reshape(28, 28))
  plt.title(class_names[int(y_train[randomid[i]].item())], fontsize = 10)
plt.show()
```



데이터 선별 및 탐색

Reshape

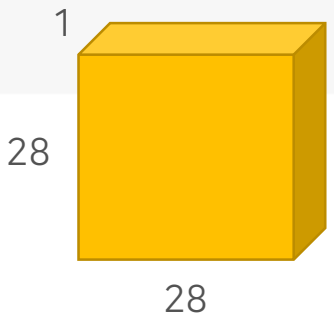
[8]
```
# 기존 shape
x_train.shape, x_test.shape
```

```
((80000, 784), (20000, 784))
```

[9]
```
# Reshape

trainX = x_train.reshape(x_train.shape[0], image_size, image_size, 1).astype('float32')
testX = x_test.reshape(x_test.shape[0], image_size, image_size, 1).astype('float32')

# Normalize Colors (0-1)
trainX /= 255.0
testX /= 255.0

# Convert Class vectors to class matrices
trainY = keras.utils.to_categorical(y_train, n_classes)  # One-hot 포맷 변환
testY = keras.utils.to_categorical(y_test, n_classes)
```

[10]
```
# shape 잘 변환되었다
trainX.shape, testX.shape
```

```
((80000, 28, 28, 1), (20000, 28, 28, 1))
```



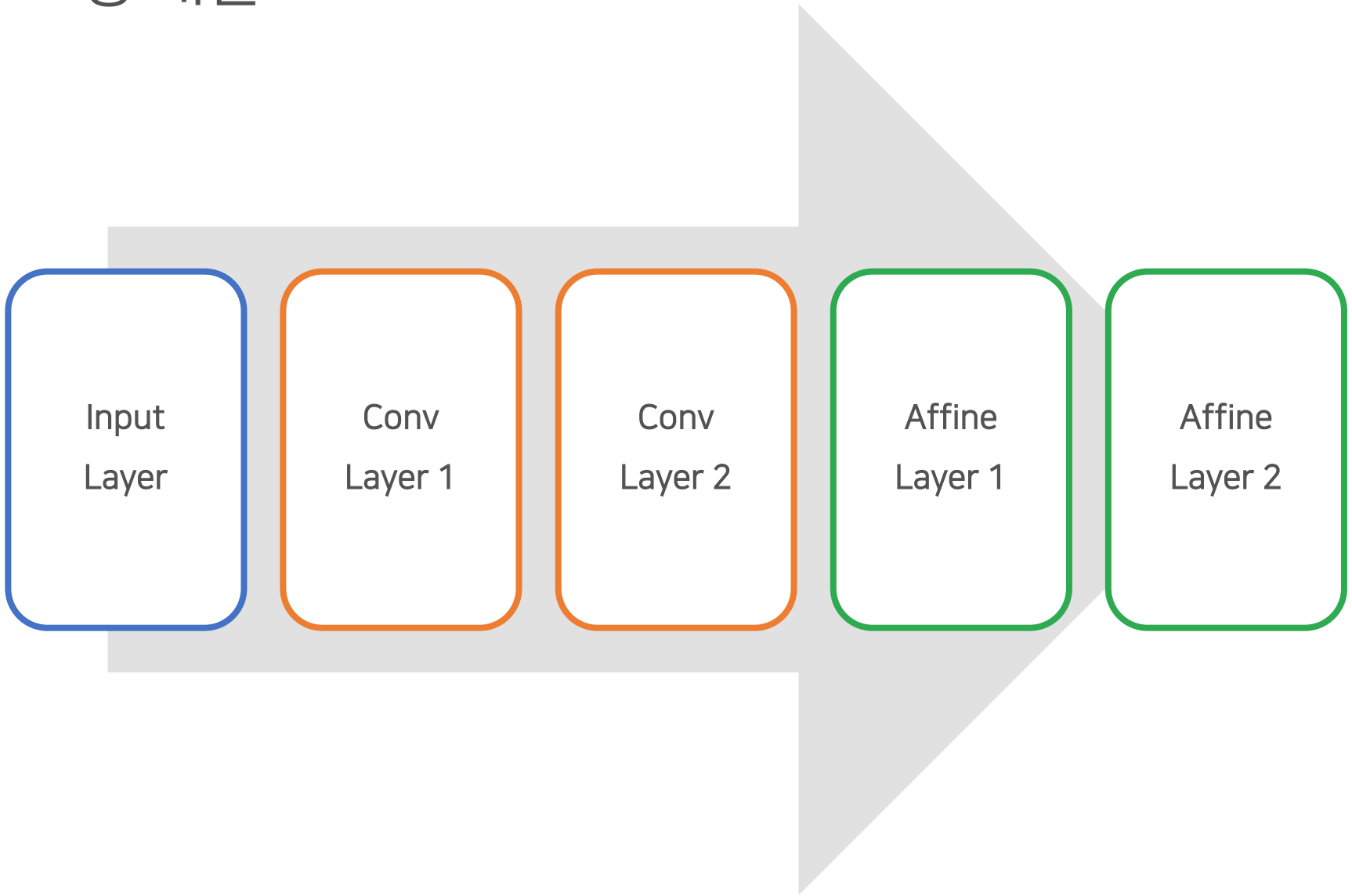
Part 2

CNN 학습 모형 개발

Convolutional Neural Network with Keras

CNN 학습 모형 개발

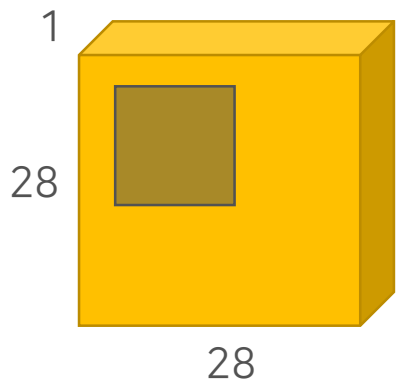
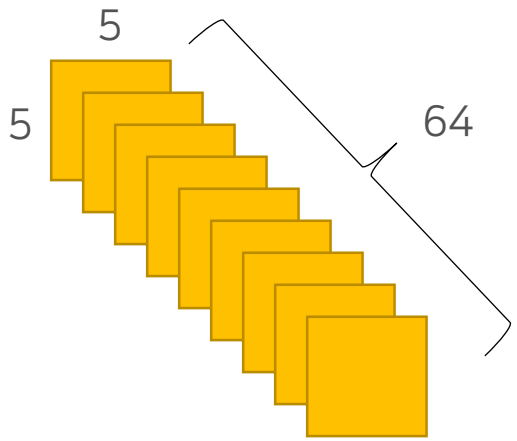
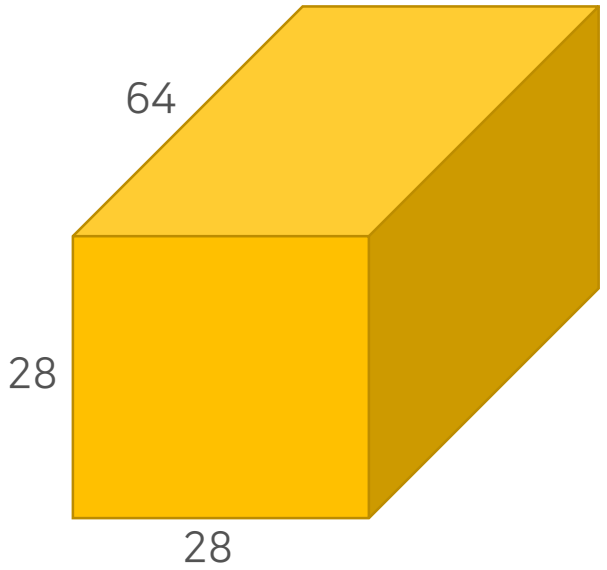
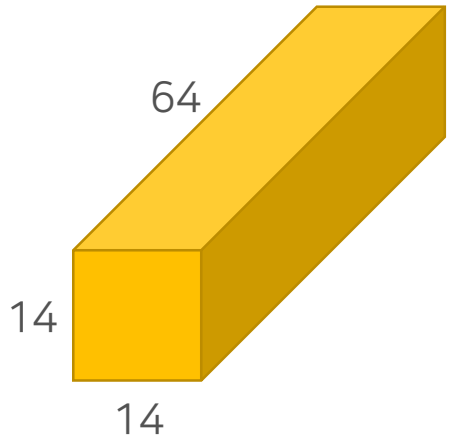
Model



CNN 학습 모형 개발

Convolution layer 1

```
#Convolution layer 1
model.add(Conv2D(64, kernel_size = (5, 5), activation = "relu", input_shape = (28, 28, 1), padding = "same"))
model.add(MaxPooling2D(pool_size = (2, 2)))
```

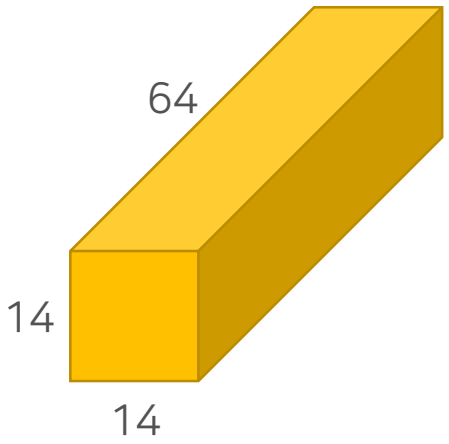
Conv2D			MaxPooling2D
Input	Filter	ReLu	Pooling
			

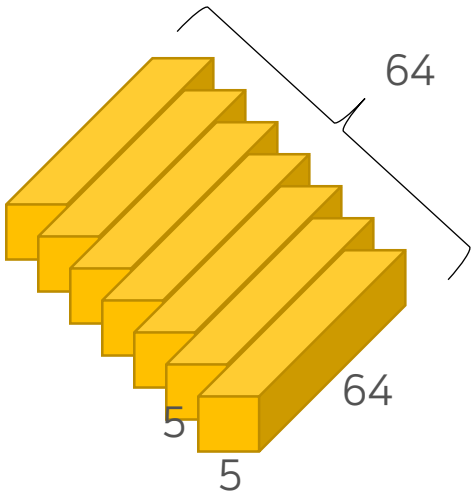
CNN 학습 모형 개발

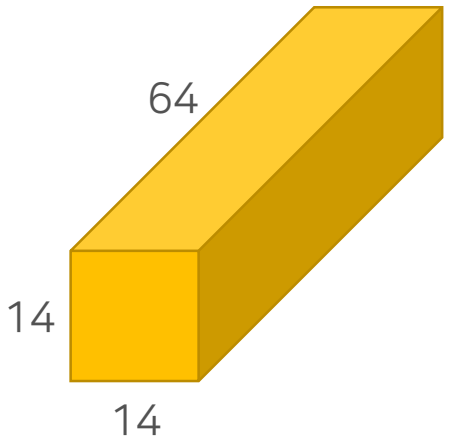
Convolution layer 2

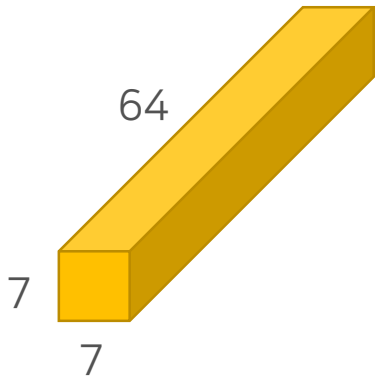
```
#Convolution layer 2
model.add(Conv2D(64, kernel_size = (5, 5), activation = "relu", input_shape = (14, 14, 64), padding = "same"))
model.add(MaxPooling2D(pool_size = (2, 2)))
```

Conv2D			MaxPooling2D
Input	Filter	ReLu	Pooling







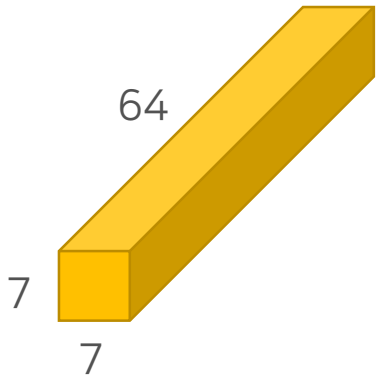
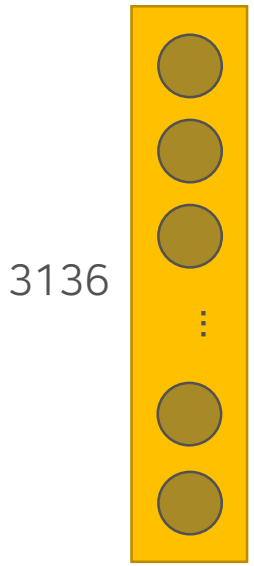
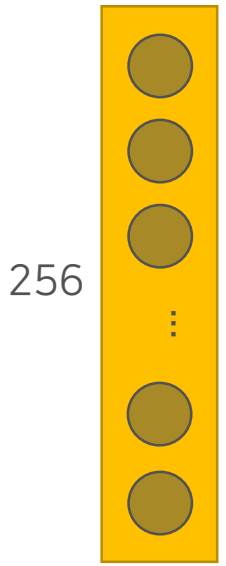
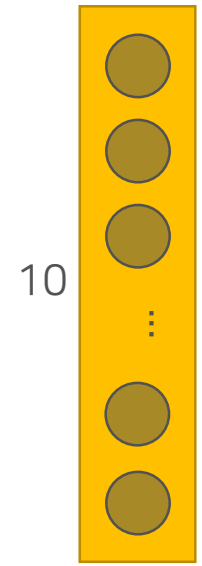


CNN 학습 모형 개발

Affine layer

```
#reshape
model.add(Flatten())
```

```
#Affine layer 1
model.add(Dense(256, activation = "relu"))

#Affine layer 2
model.add(Dense(10, activation = "softmax"))
```

Reshape		Affine	
Input	Flatten	ReLu	Softmax
			

CNN 학습 모형 개발

Model Summary

```
print(model.summary())
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_11 (Conv2D)           (None, 28, 28, 64)        1664
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 14, 14, 64)        102464
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 7, 7, 64)          0
_________________________________________________________________
flatten_6 (Flatten)          (None, 3136)              0
_________________________________________________________________
dense_11 (Dense)             (None, 256)               803072
_________________________________________________________________
dense_12 (Dense)             (None, 10)                2570
=================================================================
Total params: 909,770
Trainable params: 909,770
Non-trainable params: 0
_________________________________________________________________
None
```

CNN 학습 모형 개발

Cross Entropy & Training

```
model.compile(loss = 'categorical_crossentropy', optimizer = tf.train.AdamOptimizer(), metrics = ["accuracy"])
```

```
batch_size = 128

model.fit(trainX, trainY, batch_size = batch_size, epochs = 10, verbose = 1)
```

```
Epoch 1/10
80000/80000 [==============================] - 11s 133us/step - loss: 0.3562 - acc: 0.8952
Epoch 2/10
80000/80000 [==============================] - 10s 128us/step - loss: 0.1794 - acc: 0.9488
Epoch 3/10
80000/80000 [==============================] - 10s 128us/step - loss: 0.1374 - acc: 0.9604
Epoch 4/10
80000/80000 [==============================] - 10s 128us/step - loss: 0.1047 - acc: 0.9691
Epoch 5/10
80000/80000 [==============================] - 10s 129us/step - loss: 0.0787 - acc: 0.9761
Epoch 6/10
80000/80000 [==============================] - 10s 129us/step - loss: 0.0556 - acc: 0.9835
Epoch 7/10
80000/80000 [==============================] - 10s 129us/step - loss: 0.0371 - acc: 0.9879
Epoch 8/10
80000/80000 [==============================] - 10s 128us/step - loss: 0.0276 - acc: 0.9916
Epoch 9/10
80000/80000 [==============================] - 10s 129us/step - loss: 0.0225 - acc: 0.9925
Epoch 10/10
80000/80000 [==============================] - 10s 128us/step - loss: 0.0182 - acc: 0.9944
<keras.callbacks.History at 0x7f9ae8c0a278>
```

Part 3

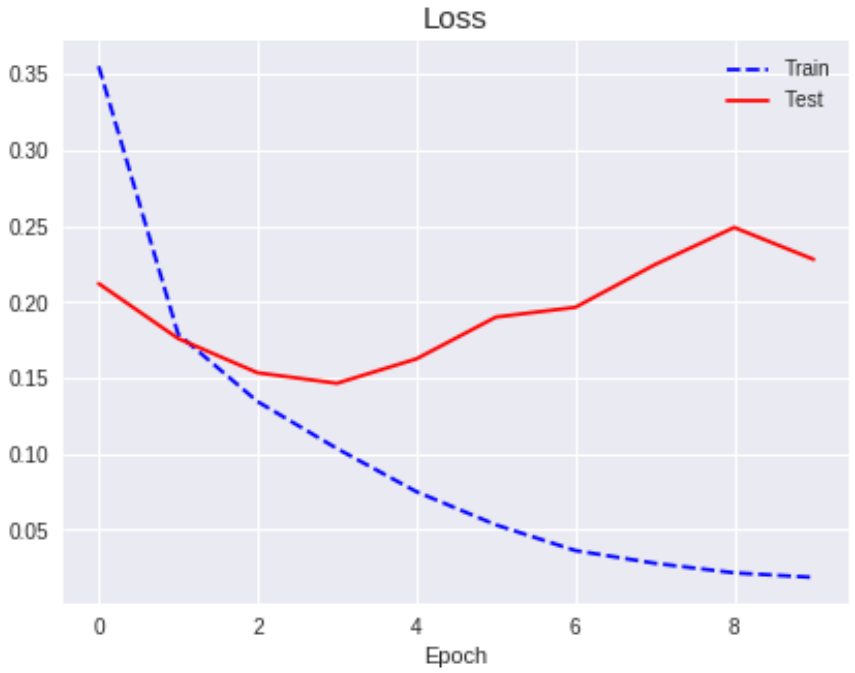
모형의 평가 및 예측

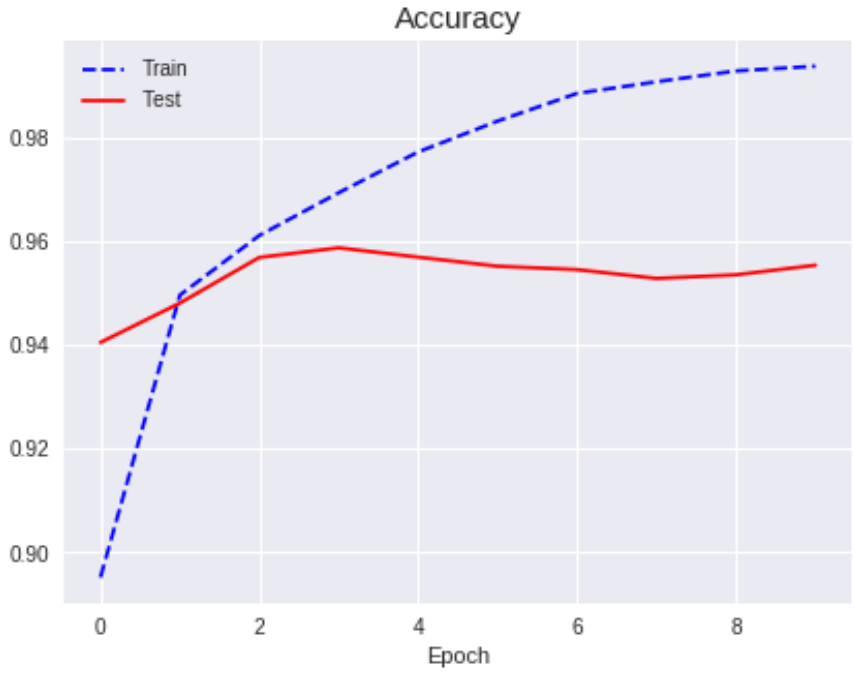
Model Evaluation & Prediction

모형의 평가 및 예측

Model Evaluation

```
my_learning = model.fit(trainX, trainY,
                        batch_size = batch_size,
                        epochs = epochs,
                        validation_data = (testX, testY))

train_loss = my_learning.history['loss']
train_accuracy = my_learning.history['acc']
test_loss = my_learning.history['val_loss']
test_accuracy = my_learning.history['val_acc']
```





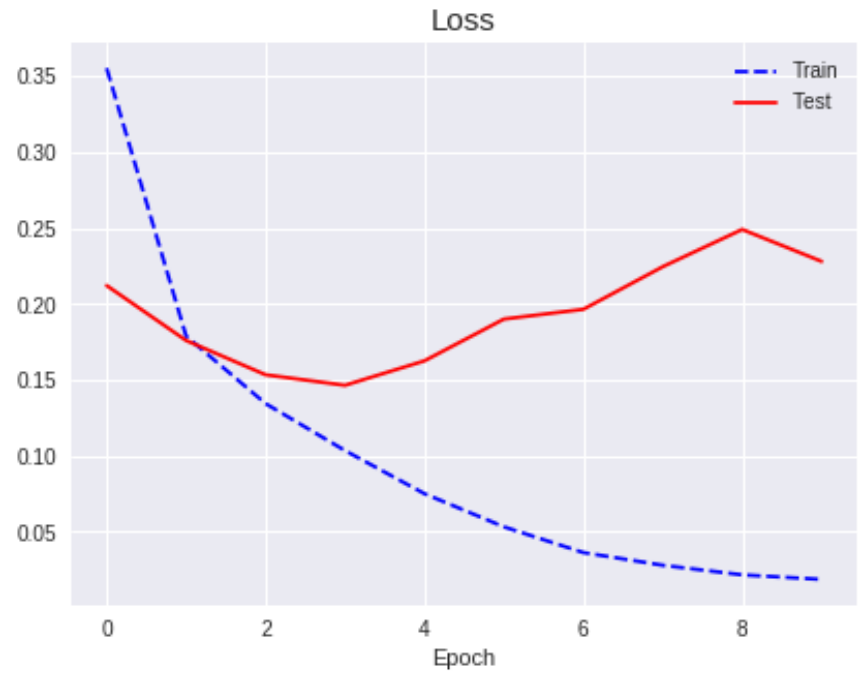
모형의 평가 및 예측

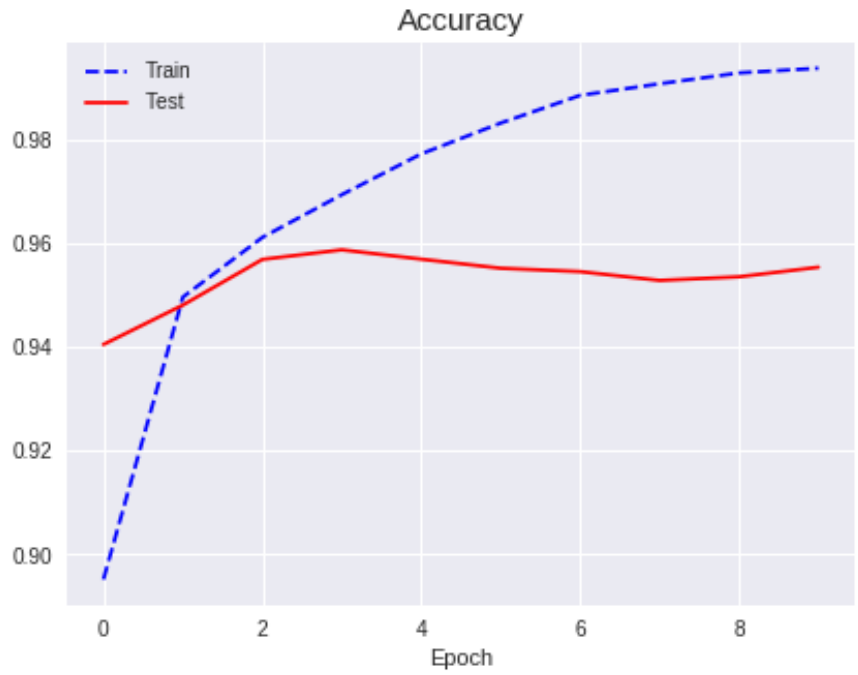
Model Evaluation

```
my_learning = model.fit(trainX, trainY,
                        batch_size = batch_size,
                        epochs = epochs,
                        validation_data = (testX, testY))

train_loss = my_learning.history['loss']
train_accuracy = my_learning.history['acc']
test_loss = my_learning.history['val_loss']
test_accuracy = my_learning.history['val_acc']
```

```
train_score = model.evaluate(trainX, trainY, verbose = 0)
test_score = model.evaluate(testX, testY, verbose = 0)

print('Train Accuracy: {}%'.format(train_score[1] * 100))
print('Test Accuracy: {}%'.format(test_score[1] * 100))
```

```
Train Accuracy: 99.575%
Test Accuracy: 95.54%
```





모형의 평가 및 예측

Model Prediction 1 - Sample

Index	Class Name
0	Pencil
1	Fish
2	Teddy-Bear
3	The Eiffel Tower
4	Scissors
5	Guitar
6	The Mona Lisa
7	Sheep
8	LapTop
9	Soccer Ball

- Test 데이터 중 10개를 랜덤으로 추출하여 분류된 Label을 확인
- sampleY는 [1, 10]의 array 형태 (ex. Sheep = [0, 0, ⋯, 1, 0, 0]) → np.argmax 이용

```
sampleID = np.random.randint(0, len(testX), 10)
sampleX = np.array([testX[i] for i in sampleID])
sampleY = np.array([testY[i] for i in sampleID])

predicted = model.predict_classes(sampleX, verbose = 0)

targets = np.argmax(sampleY, axis = 1)
predictions = predicted

print(targets)
print(predictions)
```

```
[0 2 5 2 2 7 1 9 4 7]
[0 4 5 2 2 7 5 9 4 7]
```

모형의 평가 및 예측

Model Prediction 1 - Sample

Index	Class Name
0	Pencil
1	Fish
2	Teddy-Bear
3	The Eiffel Tower
4	Scissors
5	Guitar
6	The Mona Lisa
7	Sheep
8	LapTop
9	Soccer Ball

- Test 데이터 중 10개를 랜덤으로 추출하여 분류된 Label을 확인
- sampleY는 [1, 10]의 array 형태 (ex. Sheep = [0, 0, ⋯, 1, 0, 0]) → np.argmax 이용

```
sampleID = np.random.randint(0, len(testX), 10)
sampleX = np.array([testX[i] for i in sampleID])
sampleY = np.array([testY[i] for i in sampleID])

predicted = model.predict_classes(sampleX, verbose = 0)

targets = np.argmax(sampleY, axis = 1)
predictions = predicted

print(targets)
print(predictions)
```

```
[0 2 5 2 2 7 1 9 4 7]
[0 4 5 2 2 7 5 9 4 7]
```

```
model.predict_classes(sampleX)
```

```
array([0, 4, 5, 2, 2, 7, 5, 9, 4, 7])
```

모형의 평가 및 예측

Model Prediction 1 - Sample

```
[0 2 5 2 2 7 1 9 4 7]
[0 4 5 2 2 7 5 9 4 7]
```

```
predicted_prob = model.predict(sampleX, verbose = 0)

print("Sample 2: ", np.round(predicted_prob[1], 3))
print("Sample 7: ", np.round(predicted_prob[6], 3))
```

```
Sample 2:  [0.    0.    0.012 0.    0.988 0.    0.    0.    0.    0.   ]
Sample 7:  [0.    0.    0.    0.    0.009 0.99  0.    0.    0.    0.   ]
```

Teddy Bear

Scissor

Guitar

Target: pencil
Prediction: pencil

Target: guitar
Prediction: guitar

Target: teddy-bear
Prediction: teddy-bear

Target: fish
Prediction: guitar

Target: scissors
Prediction: scissors

Target: teddy-bear
Prediction: scissors

Target: teddy-bear
Prediction: teddy-bear

Target: sheep
Prediction: sheep

Target: soccer_ball
Prediction: soccer_ball

Target: sheep
Prediction: sheep

모형의 평가 및 예측

Model Prediction 2


```
from PIL import Image

im1 = Image.open('monalisa.bmp')
im2 = Image.open('bear.bmp')
im3 = Image.open('eiffel.bmp')

im_title = ["JY's Mona Lisa", "HK's Teddy Bear", "SK's Eiffel Tower"]

for i, im in enumerate([im1, im2, im3]):
  plt.subplot(1, 3, i +1)
  plt.imshow(im)
  plt.axis('off')
  plt.title(im_title[i])

plt.show()
```

모형의 평가 및 예측

Model Prediction 2

JY's Mona Lisa

HK's Teddy Bear

SK's Eiffel Tower

```
ims = np.array([np.array(im1),
                np.array(im2),
                np.array(im3)])
print(ims.shape)

ims = ims.reshape(3, 28, 28, 1)
ims = ims / 255.0
ims = 1- ims

labels = np.array([np.array([0, 0, 0, 0, 0, 0, 1, 0, 0, 0]),  # 6: Mona Lisa
                   np.array([0, 0, 1, 0, 0, 0, 0, 0, 0, 0]),  # 2: Teddy Bear
                   np.array([0, 0, 0, 1, 0, 0, 0, 0, 0, 0])]) # 3: Eiffel Tower
print(labels.shape)
```

```
(3, 28, 28)
(3, 10)
```

모형의 평가 및 예측

Model Prediction 2

JY's Mona Lisa

HK's Teddy Bear

SK's Eiffel Tower

```
predict_class = model.predict_classes(ims, verbose = 0)
predict_prob = model.predict(ims, verbose = 0)
```

Label	Image 1	Image 2	Image 3
True	The Mona Lisa	Teddy Bear	The Eiffel Tower
Predicted	The Mona Lisa	Teddy Bear	The Eiffel Tower
Probability	100%	71.8%	100%

THANK YOU!