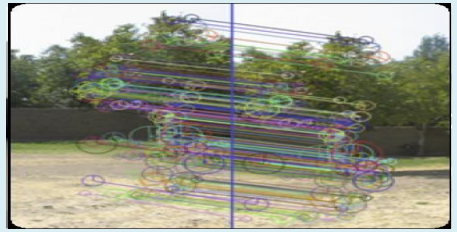
Deep Learning Basics
(#xx: Keras-based Convolutional Neural Network Practice-Part2)

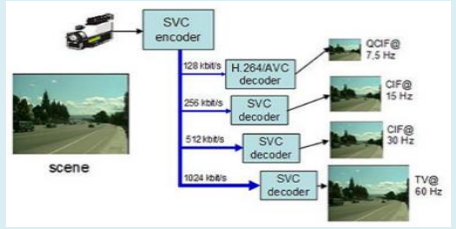
2023 Autumn

Prof. Byung-Gyu Kim
Intelligent Vision Processing Lab. (IVPL)
http://ivpl.sookmyung.ac.kr
Dept. of IT Engineering, Sookmyung Women's University
E-mail: bg.kim@ivpl.sookmyung.ac.kr

Gaol of this lecture

- Understand structure and how to develop my Convolutional Neural Network (CNN)
  - MNIST-based exercises: Digits Recognition – **Make a different model**



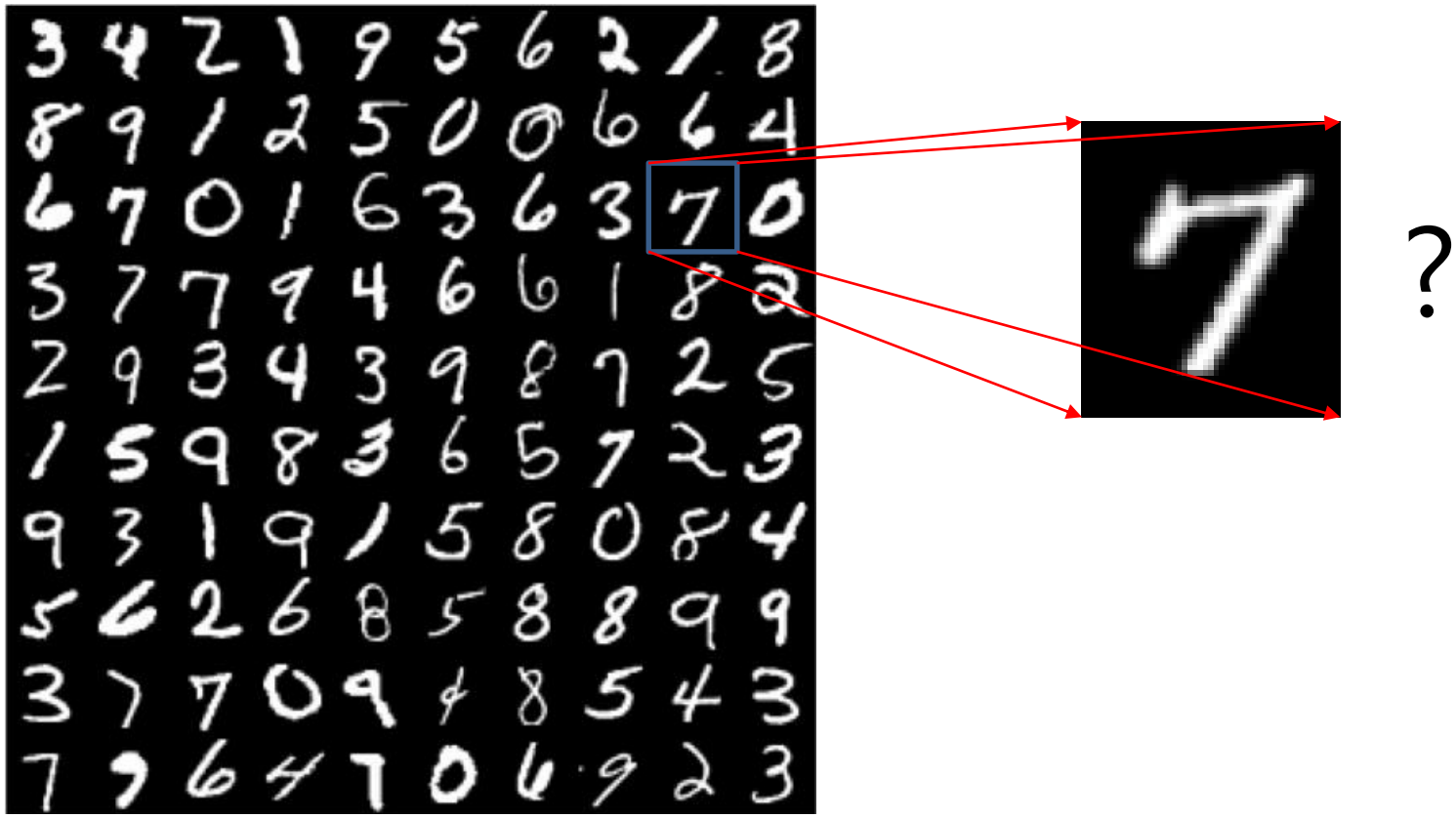


Contents

- **CNN implementation and its structure**

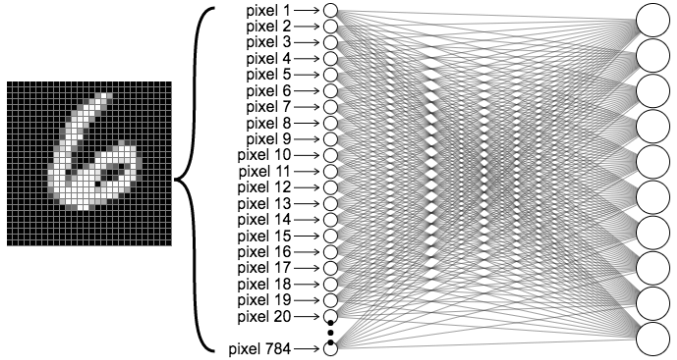
Keras Convolutional Neural Network Tutorial: MNIST (1)

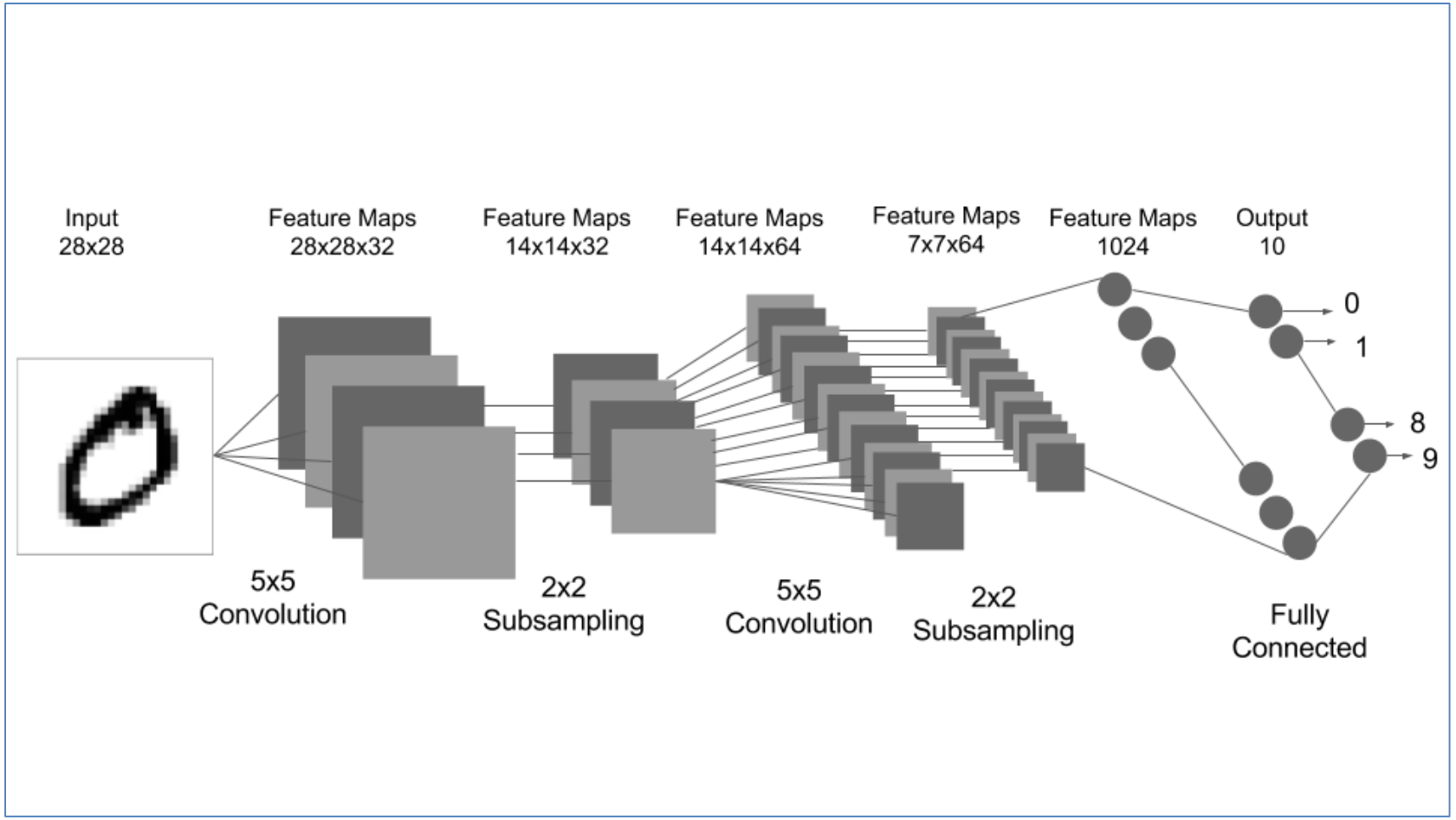
- Digits Recognition: "Just like programming has Hello World, machine learning has MNIST".
  - Goal: Handwritten Digit Prediction (Recognition) using Convolutional Neural Networks



Keras Convolutional Neural Network Tutorial: MNIST (2)

❖ Network Structure (another version)





Keras Convolutional Neural Network Tutorial: MNIST-different model (1)

- CNN Implementation

```
from __future__ import print_function
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout,
Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras import backend as K
from keras.utils import np_utils
from PIL import Image
import numpy as np
import os

batch_size = 128
num_classes = 10
epochs = 12

# input image dimensions
img_rows, img_cols = 28, 28
          (continue)
```

```
# the data, split between train and test sets
(x_train, y_train), (x_test, y_test) = mnist.load_data()

if K.image_data_format() == 'channels_first':
    x_train = x_train.reshape(x_train.shape[0], 1, img_rows,
img_cols)
    x_test = x_test.reshape(x_test.shape[0], 1, img_rows, img_cols)
    input_shape = (1, img_rows, img_cols)
else:
    x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols,
1)
    x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
    input_shape = (img_rows, img_cols, 1)

x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples' )

                    (continue)
```

Keras Convolutional Neural Network Tutorial: MNIST-different model (2)

```
# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train,
num_classes)
y_test = keras.utils.to_categorical(y_test,
num_classes)

model = Sequential()
model.add(Conv2D(32, kernel_size=(5, 5),
                 activation='relu',
                 input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (5, 5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation =
'softmax'))
        (contiue)
```

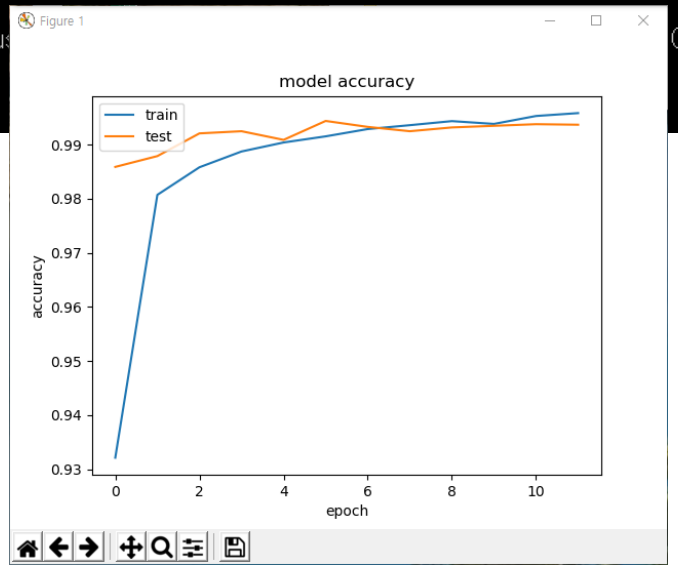
```
model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])

model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

Keras Convolutional Neural Network Tutorial: MNIST-different model (3)

- Run this CNN code…!!! (epoch = 12) → Accuracy is almost 99.40%.

```
2019-07-29 13:36:45.132301: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1326] Created TensorFlow device (/job:localhost/repli
-> physical GPU (device: 0, name: GeForce GTX 1070 Ti, pci bus id: 0000:01:00.0, compute capability: 6.1)
60000/60000 [==============================] - 6s 96us/step - loss: 0.2137 - acc: 0.9321 - val_loss: 0.0440 - val_acc: 0.9859
Epoch 2/12
60000/60000 [==============================] - 3s 56us/step - loss: 0.0631 - acc: 0.9807 - val_loss: 0.0338 - val_acc: 0.9879
Epoch 3/12
60000/60000 [==============================] - 3s 56us/step - loss: 0.0455 - acc: 0.9858 - val_loss: 0.0257 - val_acc: 0.9921
Epoch 4/12
60000/60000 [==============================] - 3s 56us/step - loss: 0.0362 - acc: 0.9888 - val_loss: 0.0231 - val_acc: 0.9925
Epoch 5/12
60000/60000 [==============================] - 3s 55us/step - loss: 0.0302 - acc: 0.9904 - val_loss: 0.0244 - val_acc: 0.9909
Epoch 6/12
60000/60000 [==============================] - 3s 56us/step - loss: 0.0263 - acc: 0.9915 - val_loss: 0.0188 - val_acc: 0.9944
Epoch 7/12
60000/60000 [==============================] - 3s 56us/step - loss: 0.0230 - acc: 0.9929 - val_loss: 0.0197 - val_acc: 0.9933
Epoch 8/12
60000/60000 [==============================] - 3s 55us/step - loss: 0.0196 - acc: 0.9936 - val_loss: 0.0221 - val_acc: 0.9925
Epoch 9/12
60000/60000 [==============================] - 3s 56us/step - loss: 0.0182 - acc: 0.9944 - val_loss: 0.0192 - val_acc: 0.9932
Epoch 10/12
60000/60000 [==============================] - 3s 56us/step - loss: 0.0180 - acc: 0.9939 - val_loss: 0.0168 - val_acc: 0.9935
Epoch 11/12
60000/60000 [==============================] - 3s 56us/step - loss: 0.0157 - acc: 0.9953 - val_loss: 0.0190 - val_acc: 0.9938
Epoch 12/12
60000/60000 [==============================] - 3s 56us/step - loss: 0.0141 - acc: 0.9956 - val_loss: 0.0199 - val_acc: 0.9937
Test loss: 0.019895846759876577
Test accuracy: 0.9937
```



Keras Convolutional Neural Network Tutorial: MNIST-different model (4)

- Actual test result:

```
Test loss: 0.019895846759876577
Test accuracy: 0.9937

----Actual test for digits----
7

[7]
2

[2]
1

[1]
0

[0]
4

[4]
1

[1]
4

[4]
9

[9]
5

[5]
9

[9]

Final test accuray: 1.000000
(BGKim) C:₩Users₩vicl₩practices₩cnn>
```

⟸ For 10 test images , 100% of accuracy..!!!

Thank you for your attention.!!!
QnA

http://ivpl.sookmyung.ac.kr