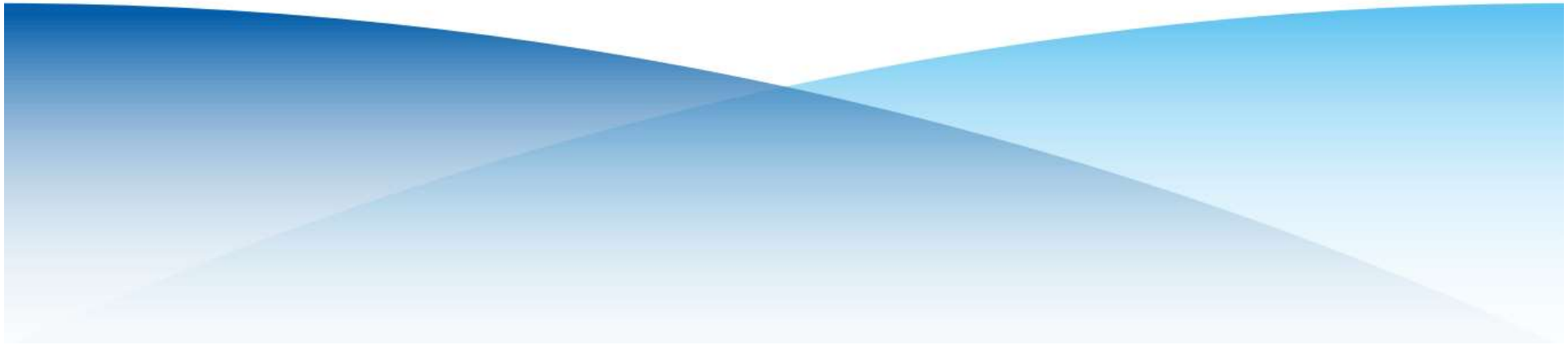
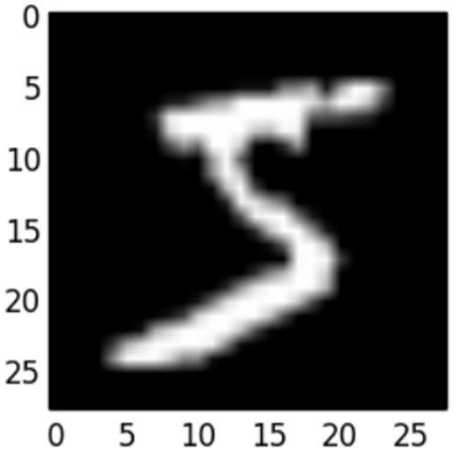
ECE30007 Intro to AI Project

Convolutional Neural Network

이원형 교수
Dr. Lee, WonHyong
whlee@handong.edu

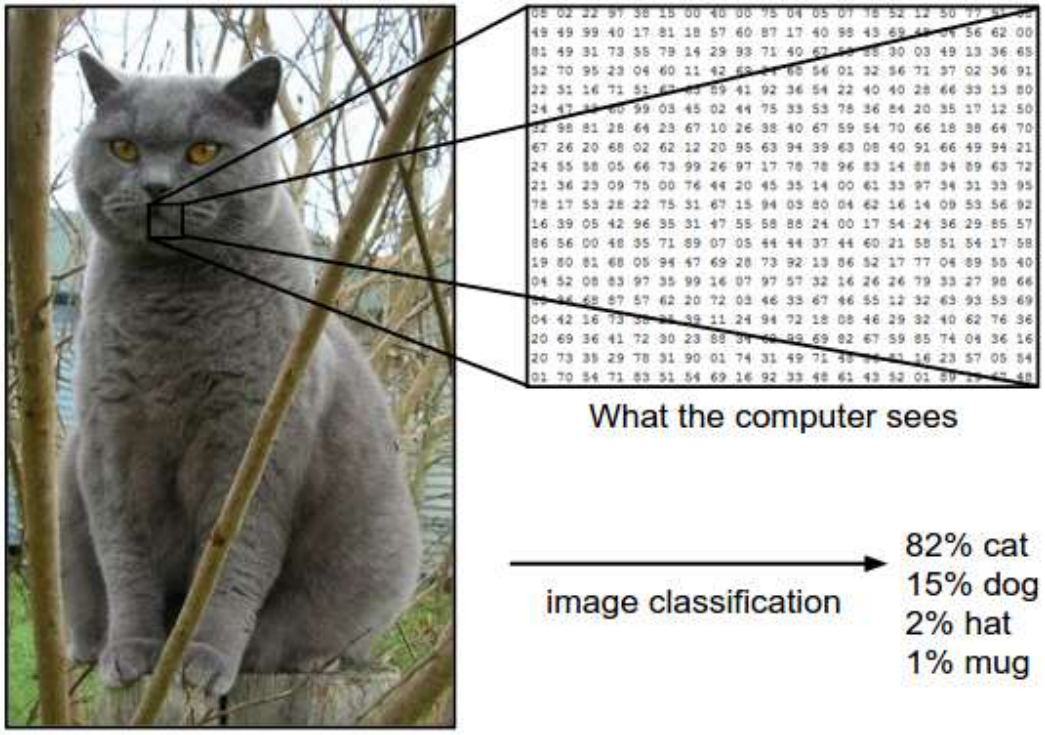


Goal



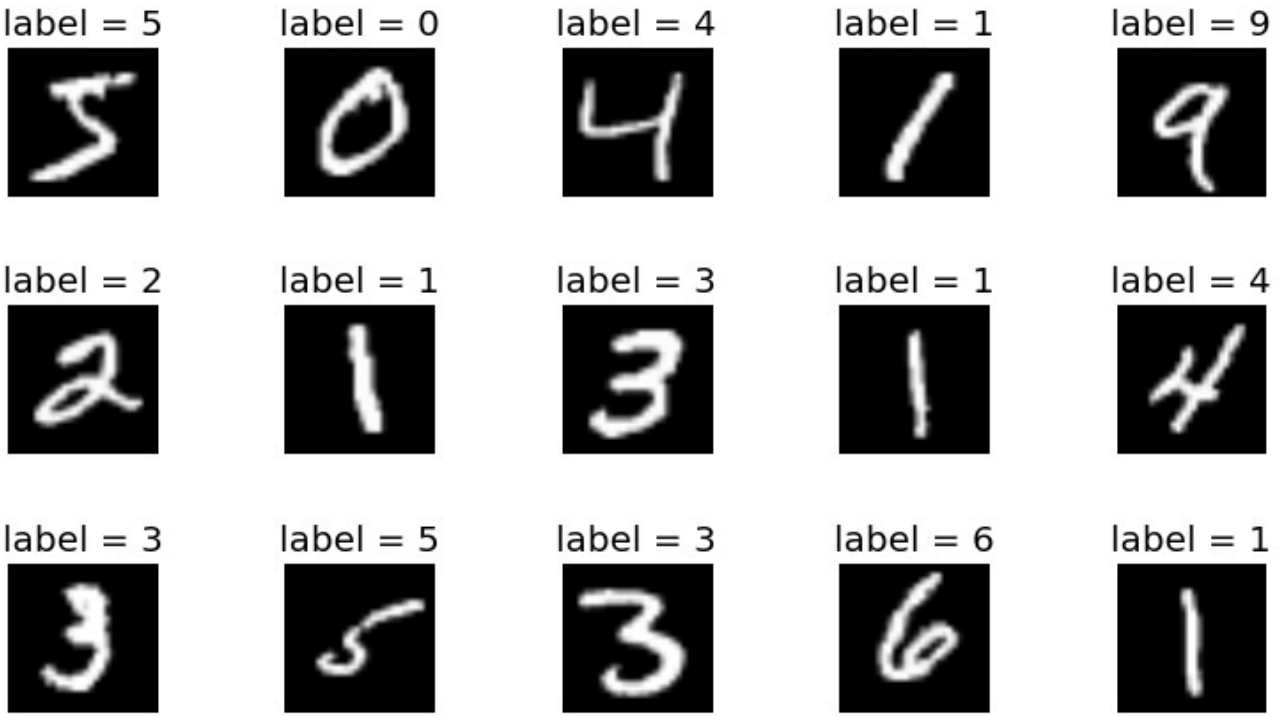
5

Intro to image data



https://3months.tistory.com/512

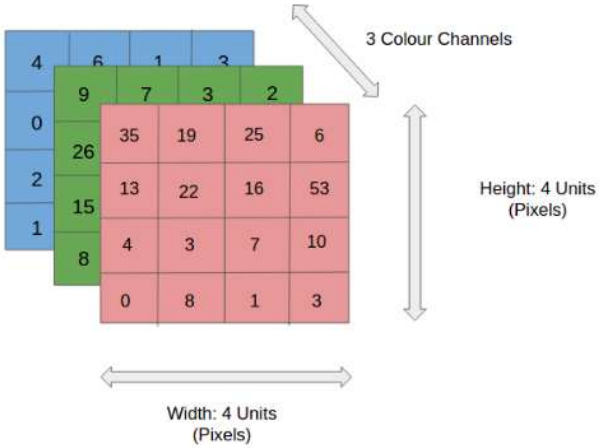
Intro to image data



What is image data

- Gray scale
  - 2-dim array
  - 0 represents black and as the number increases, the brightness increases and becomes white.

- RGB(Red-Green-Blue)
  - 3-dim array
  - It is expressed as a vector of three numbers that mean the brightness of three colors of red, green, and blue.



What is image data

- PIL(Python Imaging Library)
  - provides general image handling and lots of useful basic image operations

```
from PIL import Image
from numpy import asarray
```

```
import pickle
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

- Load image

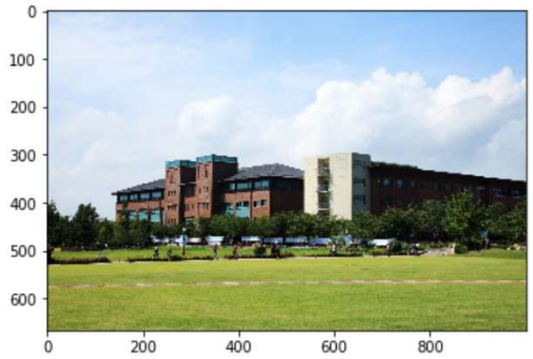
```
In [69]: img = Image.open('./data/school.jpg')

         # asarray() class is used to convert PIL images into NumPy arrays
         numpydata = asarray(img)

         #  shape
         print(numpydata.shape)
         print(type(numpydata))
         plt.imshow(numpydata)

         (666, 1000, 3)
         <class 'numpy.ndarray'>

Out[69]: <matplotlib.image.AxesImage at 0x12e49cd10>
```



What is image data

- RGB(Red-Green-Blue)
  - example

```
In [82]: plt.figure(figsize=(20,5))

         print('shape:', numpydata.shape)
         print('type:', type(numpydata))

         plt.subplot(141)
         plt.imshow(numpydata[300:600, 300:600, :])
         plt.axis("off")

         plt.subplot(142)
         plt.imshow(numpydata[300:600, 300:600, 0])
         plt.axis("off")

         plt.subplot(143)
         plt.imshow(numpydata[300:600, 300:600, 1])
         plt.axis("off")

         plt.subplot(144)
         plt.imshow(numpydata[300:600, 300:600, 2])
         plt.axis("off")

         shape: (666, 1000, 3)
         type: <class 'numpy.ndarray'>

Out[82]: (-0.5, 299.5, 299.5, -0.5)
```

Note that
subplot(1,4,1) = subplot(141)

Image processing

- Plot image *plt.imshow(np.ndarray)*
- Change color

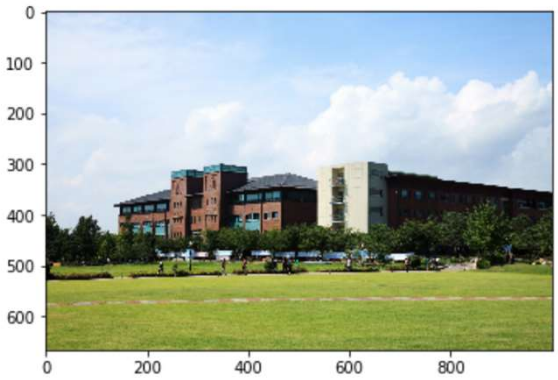
```
In [84]: img = Image.open('./data/school.jpg').convert('L')

         # asarray() class is used to convert PIL images into NumPy arrays
         numpydata = asarray(img)

         # <class 'numpy.ndarray'>
         print(type(numpydata))

         #  shape
         print(numpydata.shape)
         plt.imshow(numpydata, cmap='gray')
```

```
<class 'numpy.ndarray'>
(666, 1000)
```



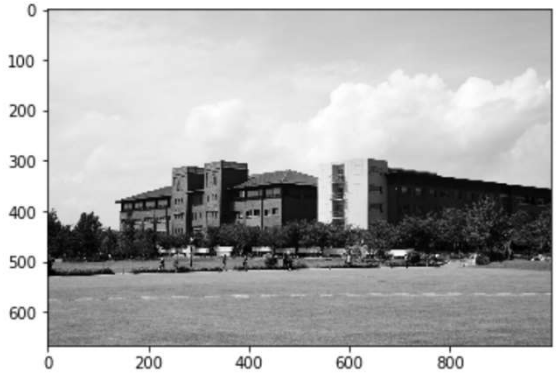


Image.convert('L'): convert into 256 level – gray image
cmap: colormap to use, try others 'Blues' or 'autumn'

Image processing

- Resize
  - Call resize() with a tuple giving the new size

```
In [97]: print(img.size)
         img2 = img.resize((300, 200))
         print(img2.size)
         img2
```

```
(1000, 666)
(300, 200)
```

Out[97]:

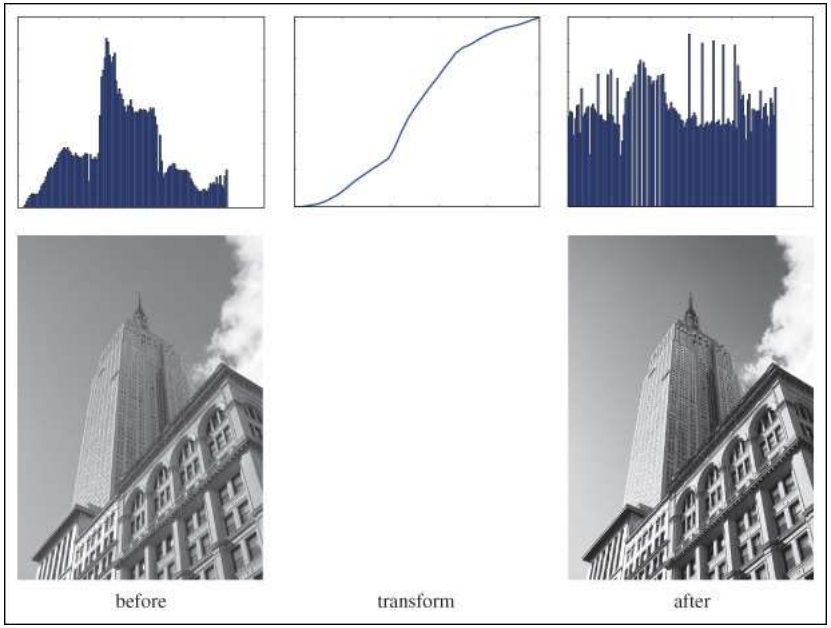
- Rotate image
  - Use counterclockwise angles and rotate()

```
In [100]: img3 = img2.rotate(45)
          img3
```

Out[100]:

Image processing [optional]

- Histogram Equalization
  - a very useful example of a gray-level transform
  - flatten the gray-level histogram of an image so that all intensities are as equally common as possible
  - normalize image intensity before other processing and increase image contrast



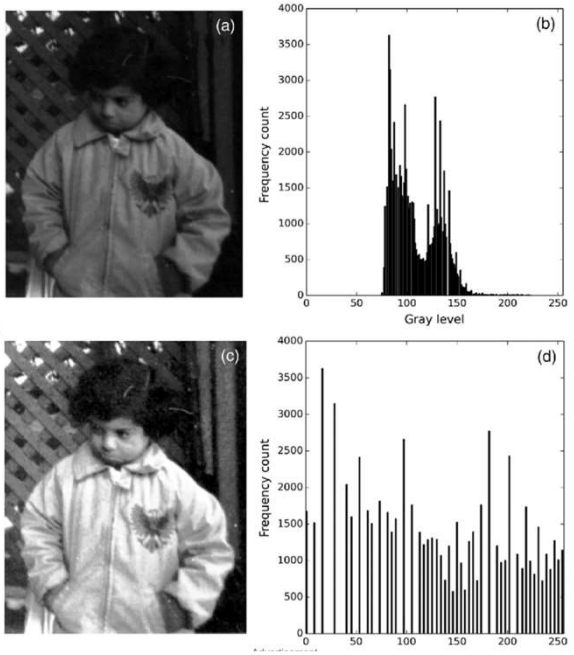
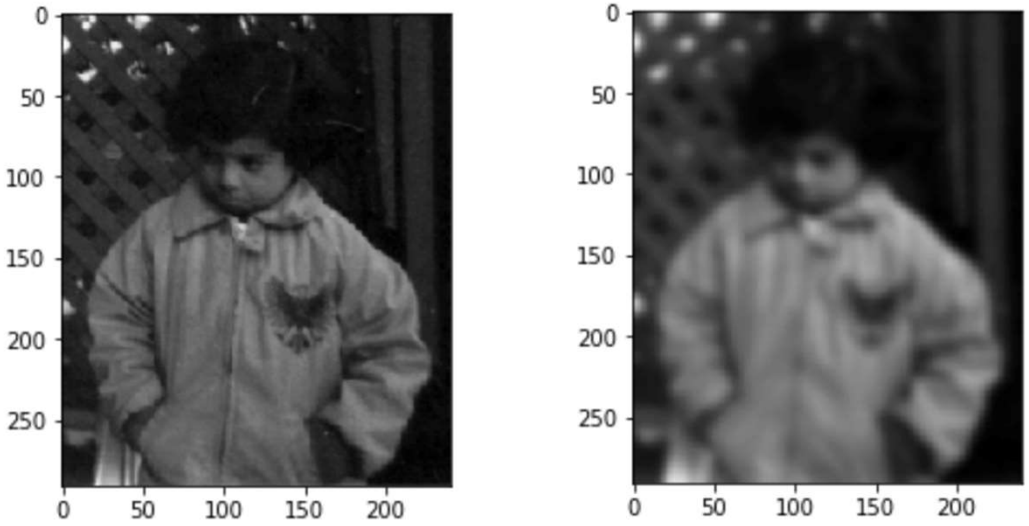


Image processing [optional]

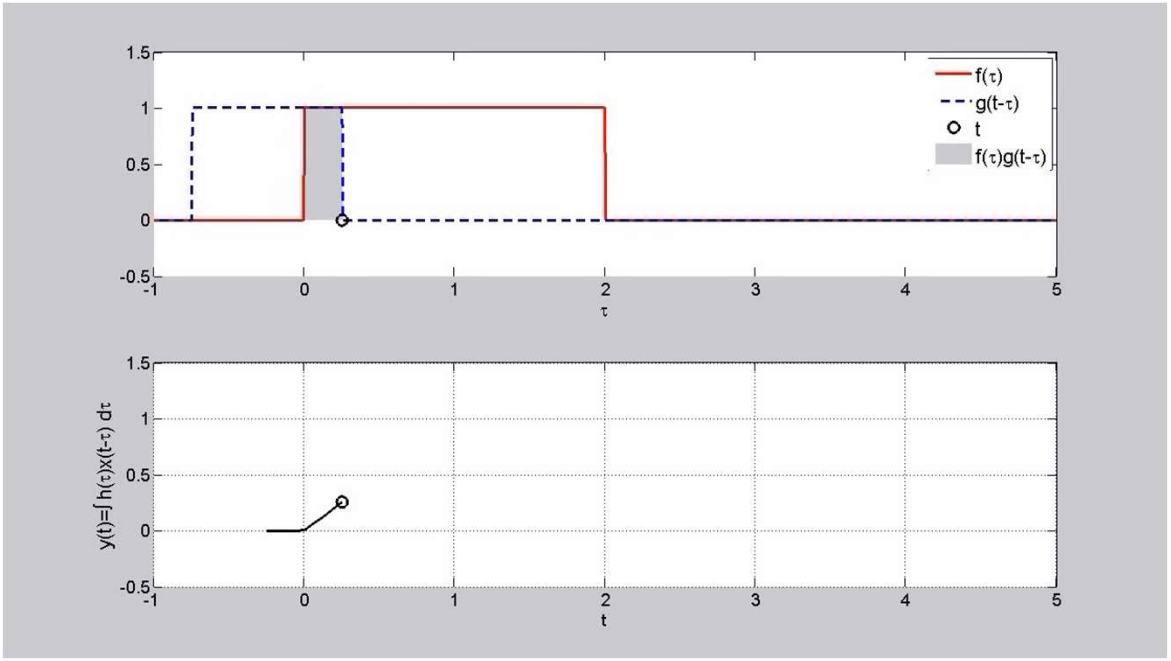
- Blurring images
  - Gaussian blurring of images
  - the (grayscale) image I is convolved with a Gaussian kernel
  - $I_\sigma = I * G_\sigma$, $G_\sigma = \frac{1}{2\pi\sigma} e^{-(x^2+y^2)/2\sigma^2}$.

```
im = array(Image.open('./data/girl.jpg').convert('L'))
im2 = filters.gaussian_filter(im, 3)
```



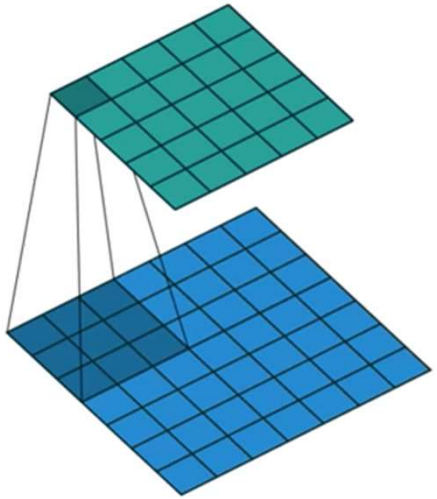
INTRODUCTION TO CNN

- CNN: Convolution Neural Network
- What is convolution?

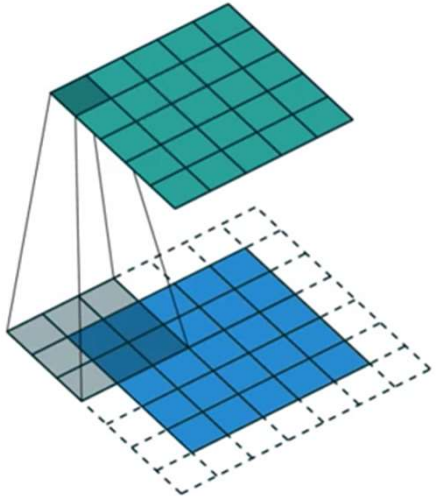


https://youtu.be/C1N55M1VD2o

❖ 2D Convolution (+ padding, strider, output dimensional shape)

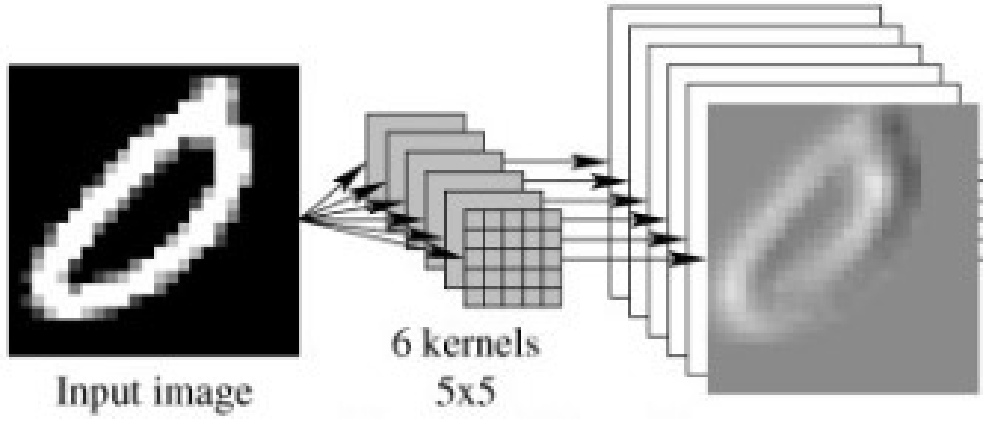


Strider = 1, Without padding
https://youtu.be/kMA1jL1E-M4



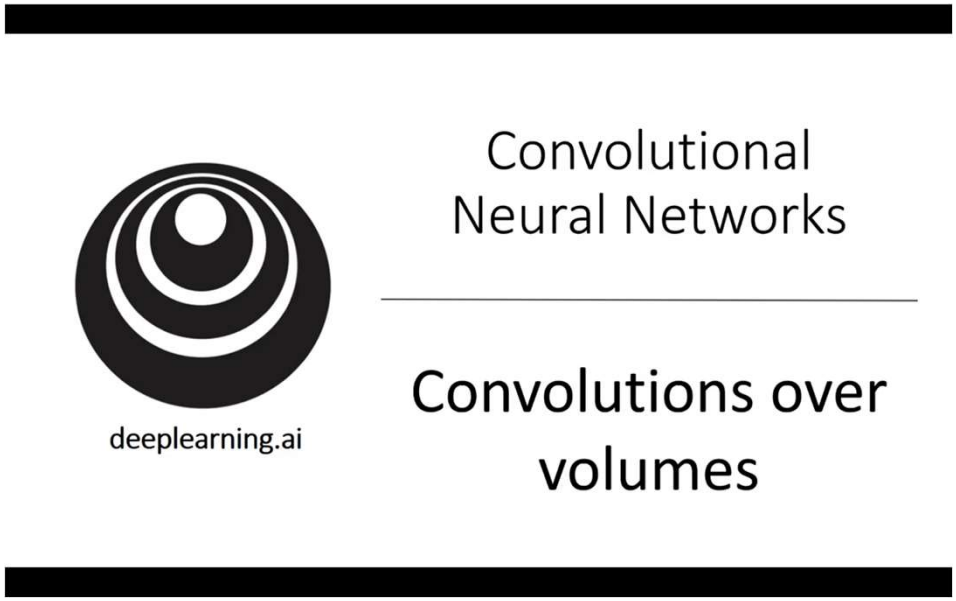
Strider = 1, With padding
https://youtu.be/Egh6iWVb_Ac

❖ **Convolutions with multiple filters(or kernels, or windows)**



https://www.sciencedirect.com/science/article/pii/S2212764X17300614

❖ **Convolutions over volumes(multiple input channels)**



https://youtu.be/KTB_OFoAQcc

❖ An example of CNN structure

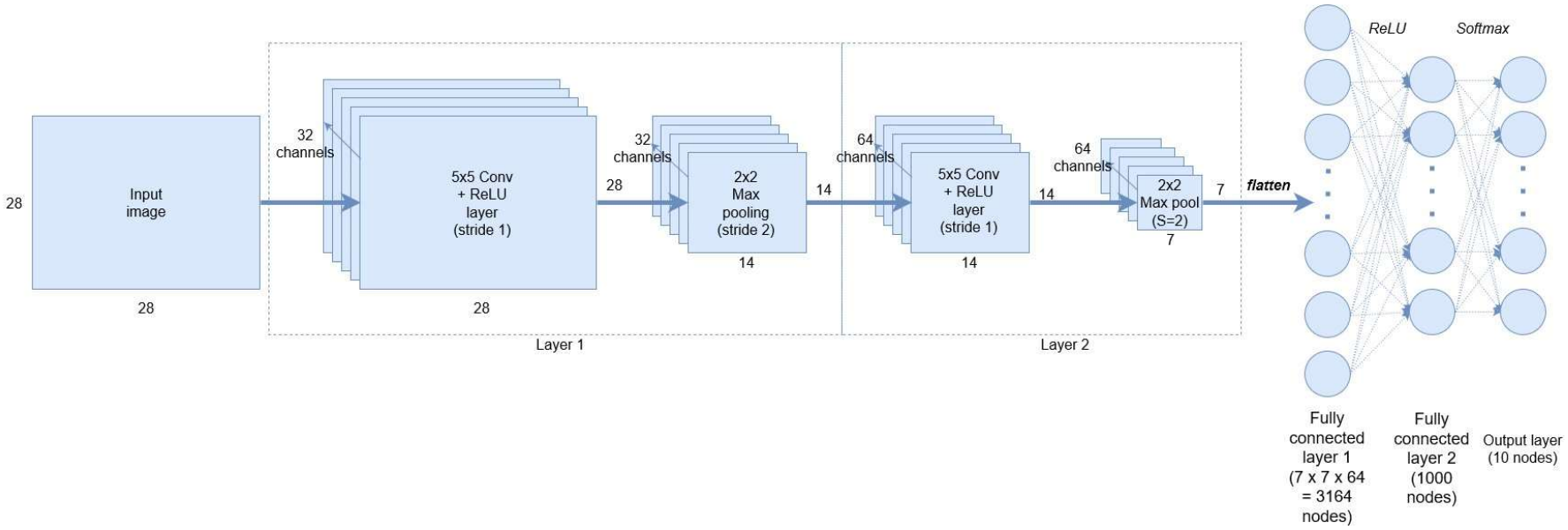
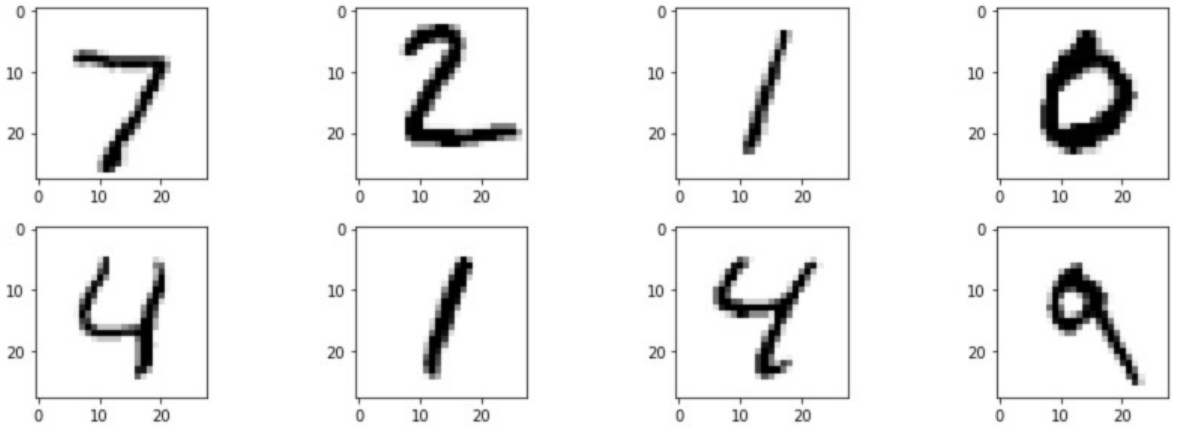


Image from Adventures in machine learning

MNIST DATASET

❖ **Hand-writing images**

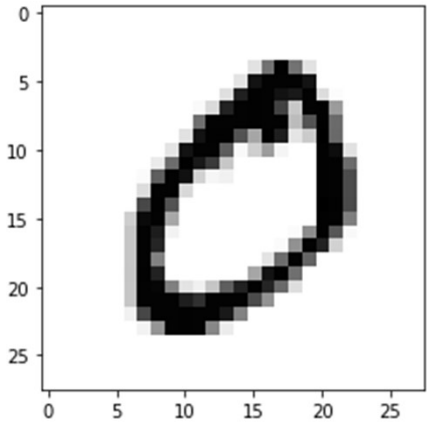


MNIST Dataset

```
import tensorflow as tf
from tensorflow import keras
import numpy as np

(x_train, y_train), (x_test, y_test) = keras.datasets.mnist.load_data()
#x_train.shape
# type(x_train[0,0,0])

x_train = x_train.astype('float32') / 255.
n = 1
plt.imshow(x_train[n], cmap='Greys', interpolation='nearest')
plt.show()
```



MNIST Dataset (Real Implementation Code)

```
x_train = x_train.reshape(x_train.shape[0], 28, 28, 1)
#x_train.shape
x_test = x_test.reshape(x_test.shape[0], 28, 28, 1)

input_shape = (28, 28, 1)
#input_shape
```

MNIST Dataset (Real Implementation Code)

```
y_train[0:10]
```

```
array([7, 2, 1, 0, 4, 1, 4, 9, 5, 9], dtype=uint8)
```

```
num_classes = 10
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
y_train[0:10]
```

```
array([[ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  1.,  0.,  0.],
       [ 0.,  0.,  1.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
       [ 0.,  1.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
       [ 1.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  1.,  0.,  0.,  0.,  0.,  0.],
       [ 0.,  1.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  1.,  0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  1.],
       [ 0.,  0.,  0.,  0.,  0.,  1.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  1.]], dtype=float32)
```

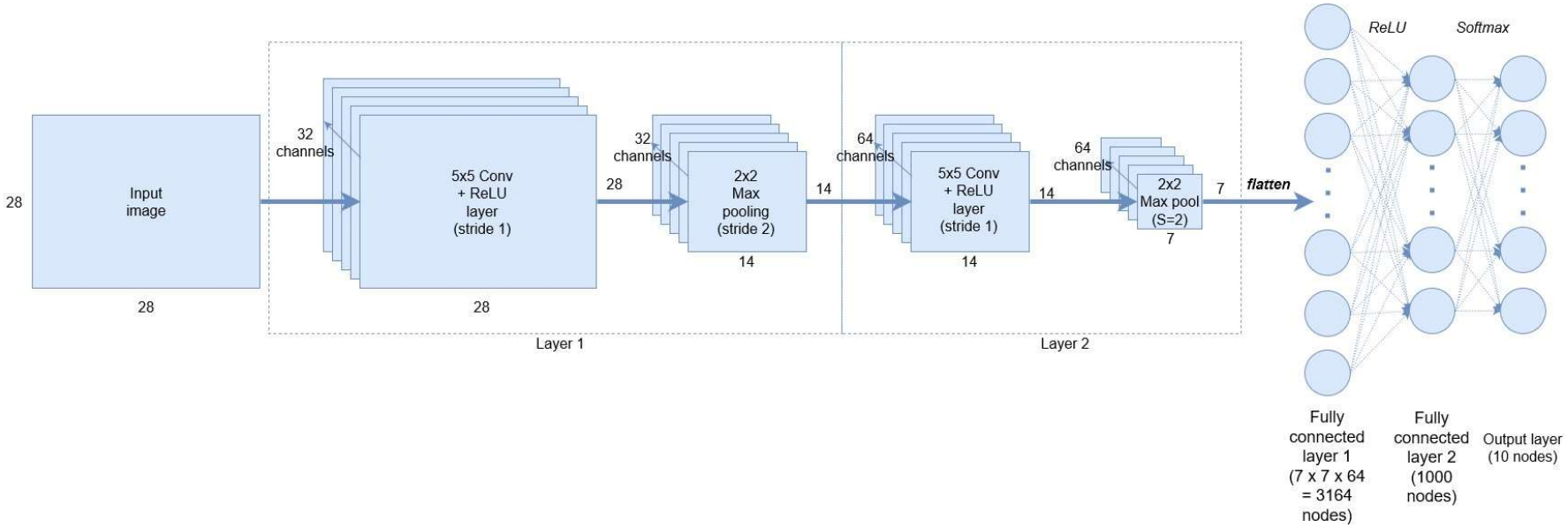
MNIST Dataset (Real Implementation Code)

```
import sys
import tensorflow as tf
import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers.convolutional import Conv2D, MaxPooling2D
import numpy as np
np.random.seed(7)
```

Code from http://pinkwink.kr/1121

CNN IMPLEMENTATION

- **Structure**



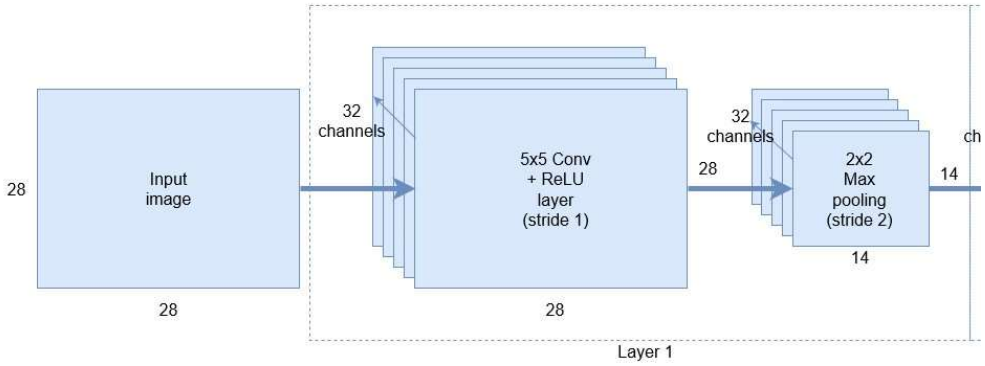
CNN Implementation (Create a model)

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(5, 5), strides=(1, 1), padding='same',
                 activation='relu',
                 input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Conv2D(64, (2, 2), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(1000, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
```

Code from http://pinkwink.kr/1121

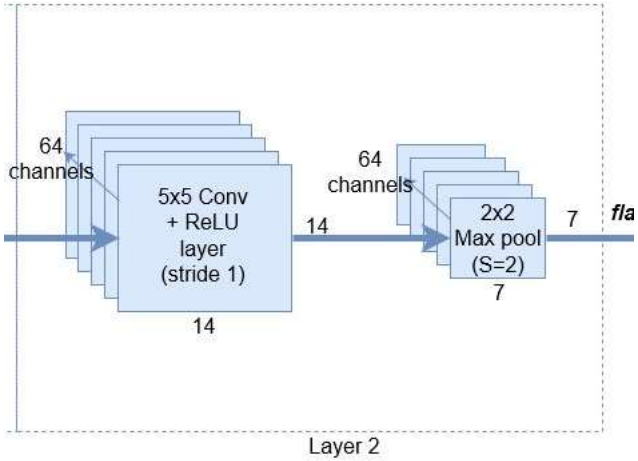
CNN Implementation (Create a model)

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(5, 5), strides=(1, 1), padding='same',
                 activation='relu',
                 input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
```



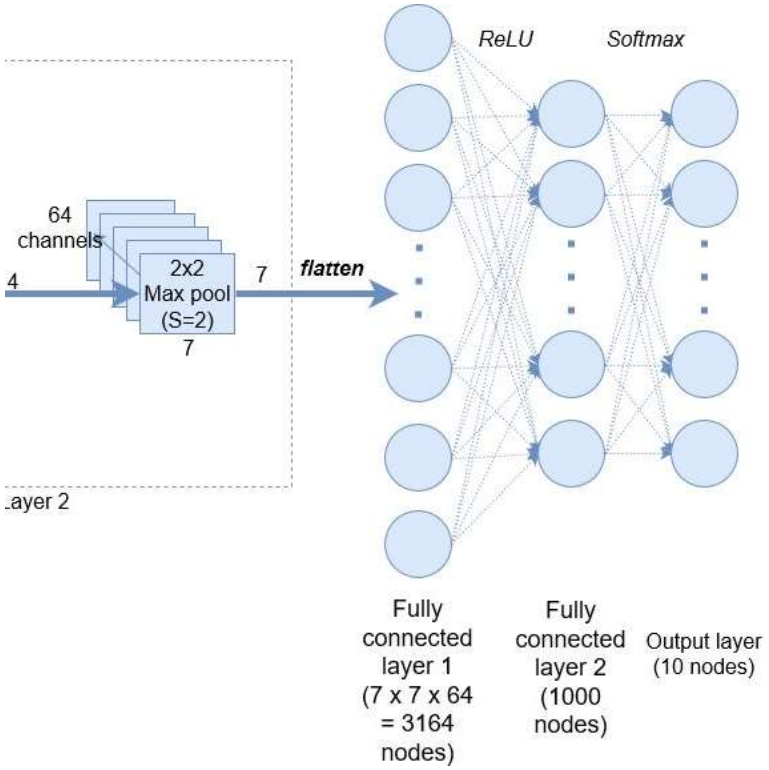
CNN IMPLEMENTATION (CREATE A MODEL)

```
model.add(Conv2D(64, (2, 2), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
```



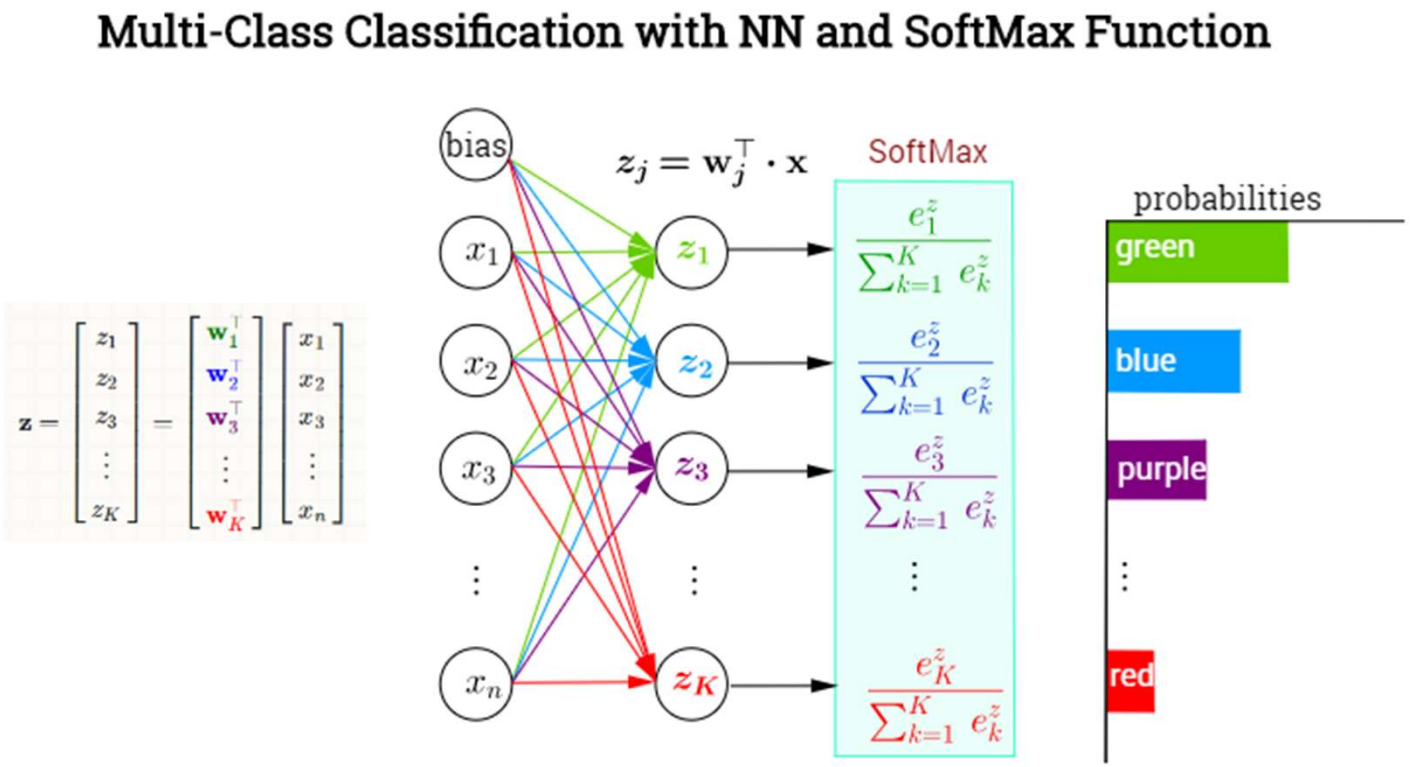
CNN IMPLEMENTATION (CREATE A MODEL)

```
model.add(Flatten())
model.add(Dense(1000, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
```



CNN Implementation (Create a model)

- **Softmax**

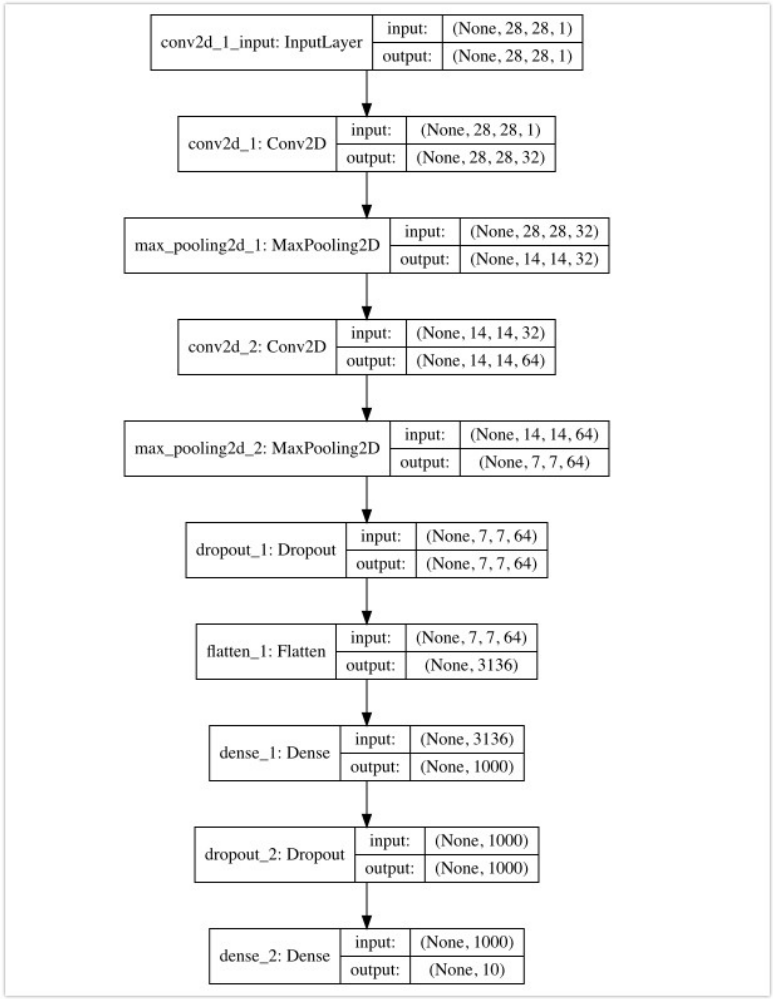


https://stats.stackexchange.com/questions/265905/derivative-of-softmax-with-respect-to-weights

CNN IMPLEMENTATION

```
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 28, 28, 32)        832
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 14, 14, 64)        8256
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 7, 7, 64)          0
_________________________________________________________________
dropout_1 (Dropout)          (None, 7, 7, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 3136)              0
_________________________________________________________________
dense_1 (Dense)              (None, 1000)              3137000
_________________________________________________________________
dropout_2 (Dropout)          (None, 1000)              0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                10010
=================================================================
Total params: 3,156,098
Trainable params: 3,156,098
Non-trainable params: 0
_________________________________________________________________
```



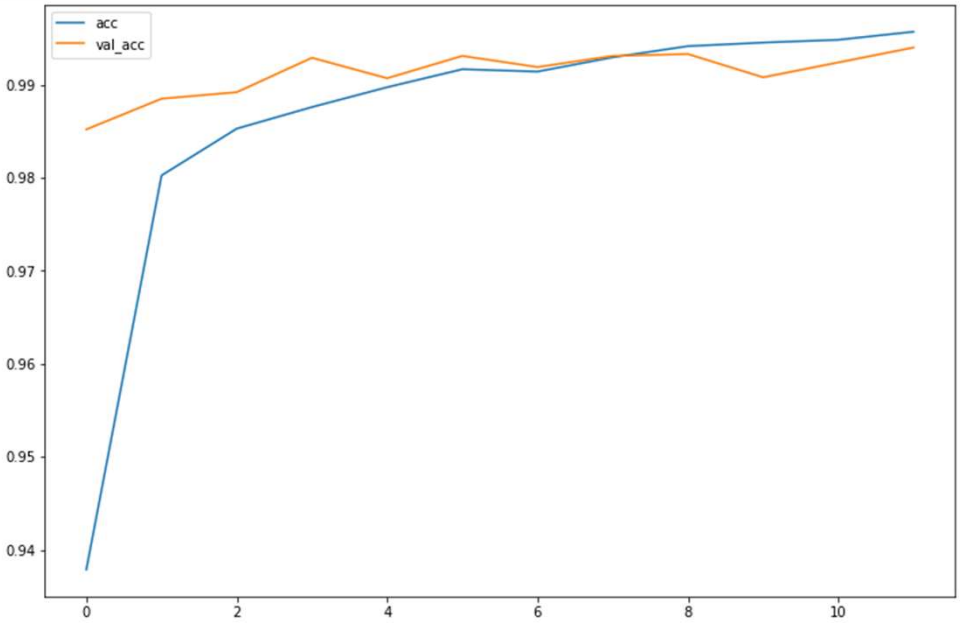
CNN Implementation (Compile and Train)

```
batch_size = 128
epochs = 12

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
hist = model.fit(x_train, y_train,
                 batch_size=batch_size,
                 epochs=epochs,
                 verbose=1,
                 validation_data=(x_test, y_test))
```

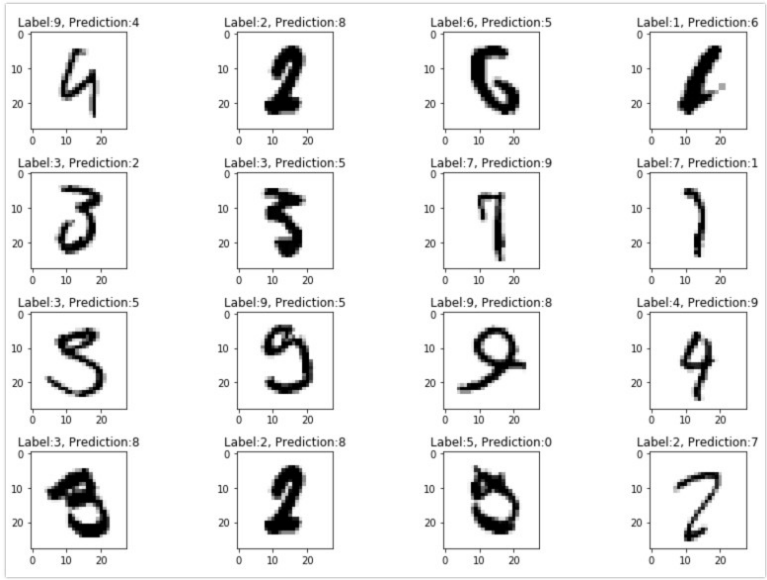
CNN IMPLEMENTATION (SHOW ACCURACY GRAPH)

```
plt.figure(figsize=(12,8))
#plt.plot(hist.history['loss'])
#plt.plot(hist.history['val_loss'])
plt.plot(hist.history['accuracy'])
plt.plot(hist.history['val_accuracy'])
#plt.legend(['loss','val_loss', 'acc','val_acc'])
plt.legend(['acc','val_acc'])
plt.show()
```



CNN IMPLEMENTATION (SHOW WRONG RESULTS)

```
import random
predicted_result = model.predict(x_test)
predicted_labels = np.argmax(predicted_result, axis=1)
test_labels = np.argmax(y_test, axis=1)
wrong_result = []
for n in range(0, len(test_labels)):
    if predicted_labels[n] != test_labels[n]:
        wrong_result.append(n)
samples = random.choices(population=wrong_result, k=16)
count = 0
nrows = ncols = 4
plt.figure(figsize=(12,8))
for n in samples:
    count += 1
    plt.subplot(nrows, ncols, count)
    plt.imshow(x_test[n].reshape(28, 28), cmap='Greys', interpolation='nearest')
    tmp = "Label:" + str(test_labels[n]) + ", Prediction:" + str(predicted_labels[n])
    plt.title(tmp)
plt.tight_layout()
plt.show()
```



HOMEWORK

❖ Follow the instructions below.

- ✓ p.30의 내용을 수업 시간 중에 검사 받거나 결과를 보고서에 첨부한다. (10점)
- ✓ How would you test your own hand writing data? (10점)
  - Implement a code that predicts your data.
    - » 본인의 손글씨로 숫자 0부터 9까지 10개의 흑백 이미지를 만든다.
      - 최대한 배경은 흰색, 글자는 검정색
      - 28x28 사이즈로 미리 만들어둬도 좋다.
    - » 이미지를 Colab에 불러온다.
    - » 28x28 이미지로 변환한다.
    - » Numpy array, (28,28,1) shape으로 변환한다.
    - » 본인이 학습한 모델에 테스트해본다.
  - 자신의 손글씨가 모두 잘 인식되었다면,
    - » 의도적으로 애매한 모양의 숫자를 적어서 데이터를 만들고 위 과정을 반복한다.
    - » 어떤 상황에서 이미지가 잘 인식되지 않는지 여러 방법으로 테스트해본다.
  - 자신의 손글씨 중에 잘 인식되지 않은 것이 있다면,
    - » 왜 잘 인식되지 않았는지 분석해본다.
    - » 의도적으로 애매한 모양의 숫자를 적어서 데이터를 만들고 위 과정을 반복한다.
    - » 어떤 상황에서 이미지가 잘 인식되지 않는지 여러 방법으로 테스트해본다.
  - 이미지를 불러와 테스트하는 내용의 코드 및 위 인식 결과에 대한 분석을 작성한다.
    - » 보고서로 HDLMS에 제출


IFU
Thank you!