CNN(Convolution Neural Network) - 합성곱 신경망

학습 내용

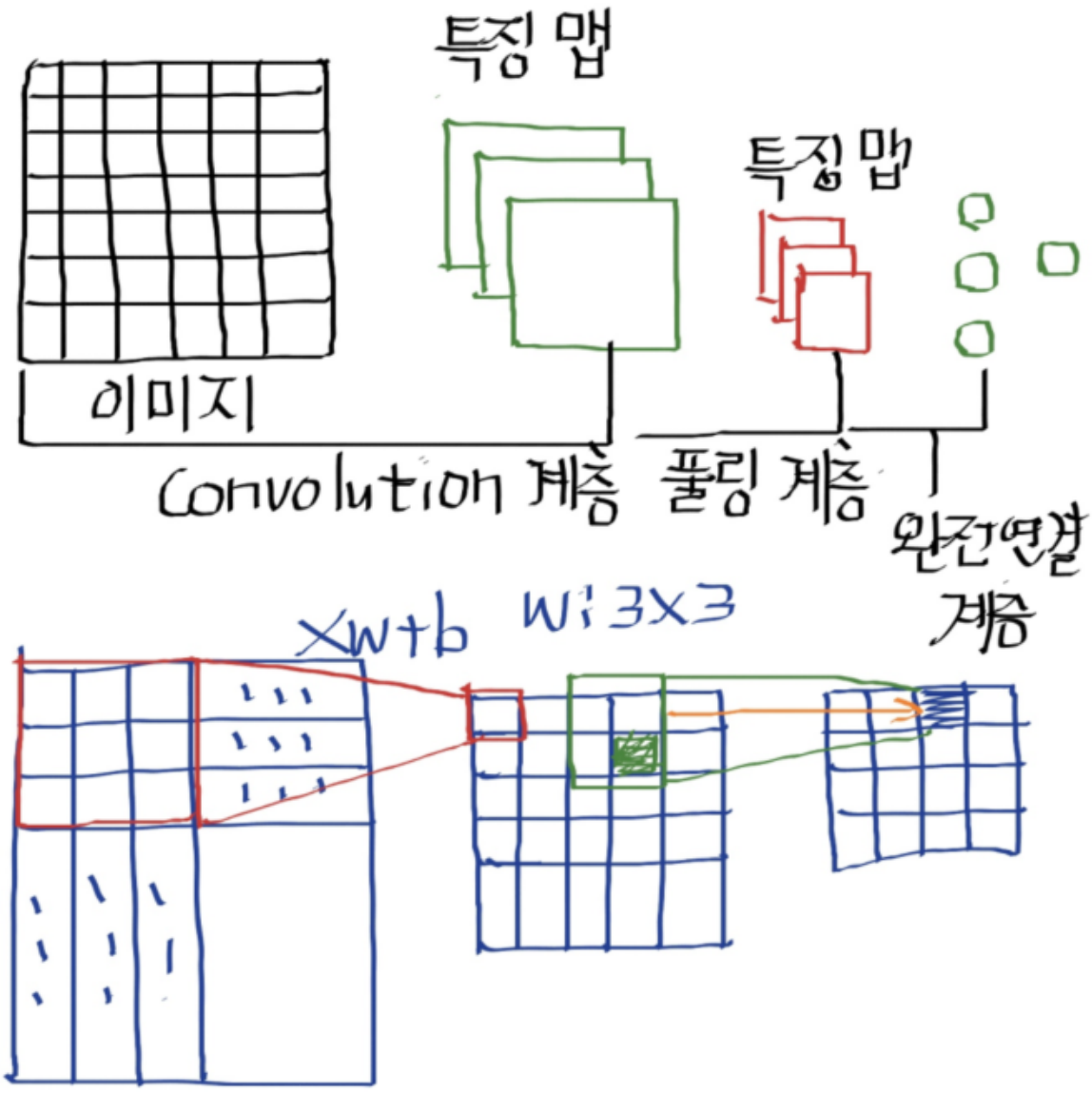
- CNN의 기본 이해
- CNN을 실습을 통해 알아보기

목차

In [19]:
```
from IPython.display import display, Image
import os, warnings

warnings.filterwarnings(action='ignore')
```

In [20]:
```
display(Image(filename="img/cnn.png"))
```



01 합성망 신경망 알아보기

목차로 이동하기

In [21]:
```
import tensorflow as tf
from tensorflow.keras import models
from tensorflow.keras import layers
import matplotlib.pyplot as plt

print(tf.__version__)
```

```
2.12.0
```

```
# tf 2.5.x 버전 local에서 error 발생할 경우 있음.
from keras import layers
from keras import models

[해결]
from tensorflow.keras import models
from tensorflow.keras import layers
```

- Conv :3x3 필터, 32개의 필터개수, 입력 이미지 (28, 28, 1)

- Maxpooling (2,2)
- Conv :3x3 필터, 64개의 필터개수
- Maxpooling (2,2)

Conv2D와 MaxPool2D

```
tf.keras.layers.Conv2D(
    filters, kernel_size, strides=(1, 1), padding='valid',
    data_format=None, dilation_rate=(1, 1), groups=1,
activation=None,
    use_bias=True, kernel_initializer='glorot_uniform',
    bias_initializer='zeros', kernel_regularizer=None,
    bias_regularizer=None, activity_regularizer=None,
kernel_constraint=None,
    bias_constraint=None, **kwargs
)


tf.keras.layers.MaxPool2D(
    pool_size=(2, 2), strides=None, padding='valid',
data_format=None,
    **kwargs
)
```

02 MNIST 데이터 셋 준비

목차로 이동하기

- MNIST 데이터 셋 준비

In [22]:
```
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical

(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
```

모델 구축

In [24]:
```
model = models.Sequential()

model.add(layers.Conv2D(filters=32, kernel_size=(3, 3),
                        activation='relu', input_shape=(28, 28, 1)))
model.add(layers.MaxPooling2D(pool_size=(2, 2)))

model.add(layers.Conv2D(filters=64, kernel_size=(3, 3),
                        activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
```

CNN 구조 알아보기

In [25]:
```
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_4 (Conv2D)           (None, 26, 26, 32)        320

 max_pooling2d_4 (MaxPooling  (None, 13, 13, 32)       0
 2D)

 conv2d_5 (Conv2D)           (None, 11, 11, 64)        18496

 max_pooling2d_5 (MaxPooling  (None, 5, 5, 64)         0
 2D)

=================================================================
Total params: 18,816
Trainable params: 18,816
Non-trainable params: 0
_________________________________________________________________
```

- (height, width, channels)크기의 3D텐서
- 높이와 넓이 차원은 네트워크가 깊어질수록 작아지는 경향이 있다.
- 채널의 수는 Conv2D층에 전달된 첫번째 매개변수에 의해 조절된다.
- (5,5,64)를 최종 이미지를 FCN(완전 연결 네트워크)의 1차원 벡터로 변경 후, 이를 연결한다.

완전 연결층(FCL) 추가

In [26]:
```
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))
```

In [27]:
```
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_4 (Conv2D)           (None, 26, 26, 32)        320

 max_pooling2d_4 (MaxPooling  (None, 13, 13, 32)       0
 2D)

 conv2d_5 (Conv2D)           (None, 11, 11, 64)        18496

 max_pooling2d_5 (MaxPooling  (None, 5, 5, 64)         0
 2D)

 flatten_1 (Flatten)         (None, 1600)              0

 dense_2 (Dense)             (None, 64)                102464

 dense_3 (Dense)             (None, 10)                650

=================================================================
Total params: 121,930
Trainable params: 121,930
Non-trainable params: 0
_________________________________________________________________
```

데이터 전처리

- 이미지 Reshape
- 정규화 0~1사이의 값으로 변경

In [28]:
```
# 입력층은 이미지 그대로, 입력층의 값의 범위 정규화
train_images = train_images.reshape((60000, 28, 28, 1))
train_images = train_images.astype('float32') / 255

test_images = test_images.reshape((10000, 28, 28, 1))
test_images = test_images.astype('float32') / 255

# 출력층 데이터-원핫 인코딩
train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)
```

In [29]:
```
print("입력층 데이터(X) : ",train_images.shape, test_images.shape )
print("출력층 데이터(y) : ",train_labels.shape, test_labels.shape )
```

```
입력층 데이터(X) :  (60000, 28, 28, 1) (10000, 28, 28, 1)
출력층 데이터(y) :  (60000, 10) (10000, 10)
```

03 모델 학습 및 평가(CNN모델)

목차로 이동하기

- MNIST 데이터 셋 준비

비용함수, 최적화 함수 구성

- 비용함수와 최적화 함수 지정
- 비용함수 : categorical_crossentropy
- 최적화 함수 : rmsprop
  - 변수(feature)마다 적절한 학습률을 적용하여 효율적인 학습 진행
  - AdaGrad보다 학습을 오래 할 수 있다.

In [31]:
```
%%time

model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
hist = model.fit(train_images, train_labels,
                 validation_data=(test_images, test_labels),
                 epochs=5, batch_size=64)
```

```
Epoch 1/5
938/938 [==============================] - 67s 70ms/step - loss: 0.1638 - accuracy:
0.9499 - val_loss: 0.0569 - val_accuracy: 0.9812
Epoch 2/5
938/938 [==============================] - 57s 61ms/step - loss: 0.0506 - accuracy:
0.9844 - val_loss: 0.0384 - val_accuracy: 0.9873
Epoch 3/5
938/938 [==============================] - 58s 62ms/step - loss: 0.0340 - accuracy:
0.9896 - val_loss: 0.0510 - val_accuracy: 0.9827
Epoch 4/5
938/938 [==============================] - 58s 62ms/step - loss: 0.0256 - accuracy:
0.9918 - val_loss: 0.0298 - val_accuracy: 0.9897
Epoch 5/5
938/938 [==============================] - 77s 82ms/step - loss: 0.0197 - accuracy:
0.9940 - val_loss: 0.0275 - val_accuracy: 0.9913
Wall time: 5min 17s
```

In [32]:
```
test_loss, test_acc = model.evaluate(test_images, test_labels)
print(test_acc)
```

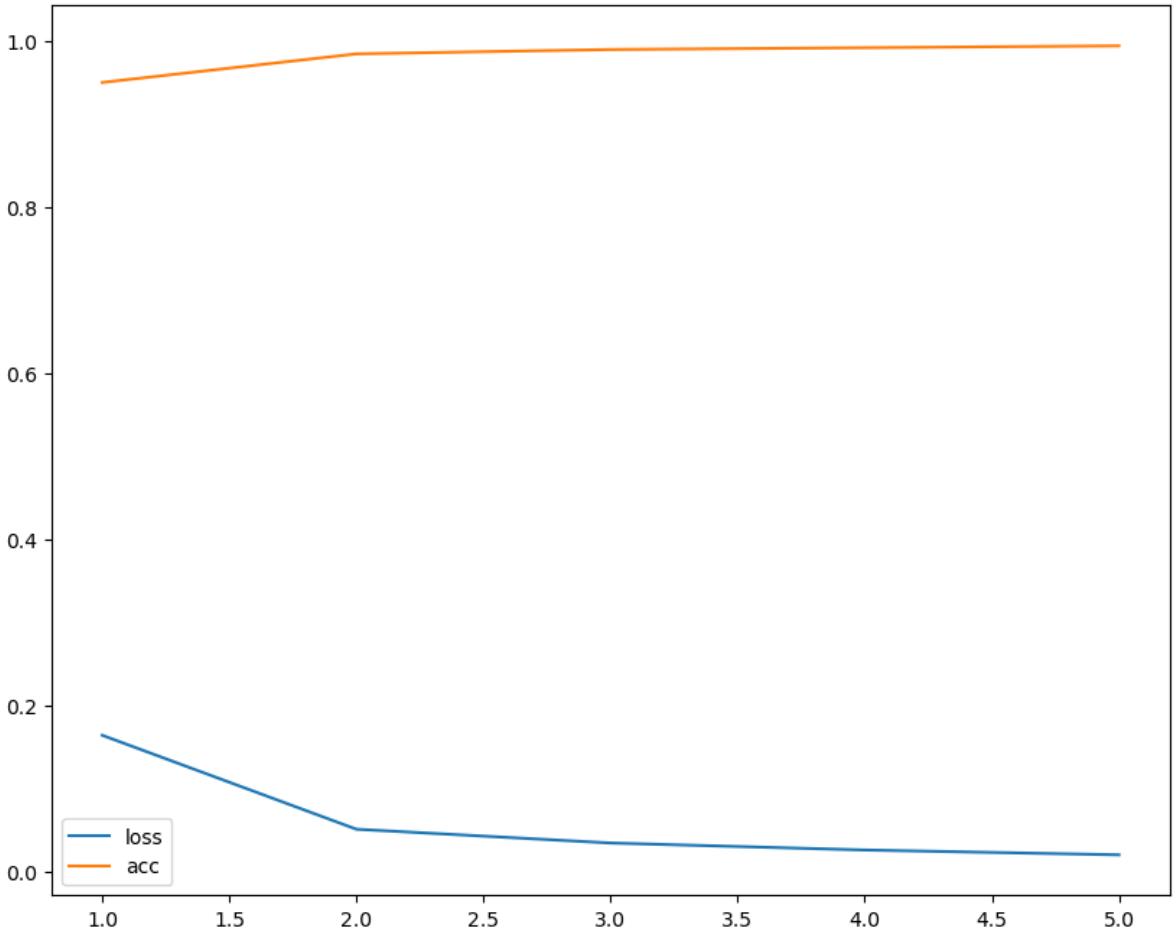
```
313/313 [==============================] - 9s 27ms/step - loss: 0.0275 - accuracy:
0.9913
0.9912999868392944
```

In [38]:
```
hist.history['loss']
```

Out[38]:
```
[0.16375844180583954,
 0.05059141293168068,
 0.03399966284632683,
 0.02555452473461628,
 0.019749119877815247]
```

In [39]:
```
plt.figure(figsize=(10,8),facecolor='white')
x_lim = range(1,6)
plt.plot(x_lim,  hist.history['loss'])
plt.plot(x_lim,  hist.history['accuracy'])
plt.legend(['loss','acc'])
```

Out[39]:
```
<matplotlib.legend.Legend at 0x20fd697e340>
```



04 모델 저장 및 불러오기

목차로 이동하기

In [43]:
```
import os

path = os.path.join(os.getcwd(), "dl_model")
savefile = os.path.join(path, "my_model_mnist.h5"  )

model.save(savefile)

os.listdir(path)
```

Out[43]:
```
['my_lastmodel.h5', 'my_model_fashion.h5', 'my_model_mnist.h5']
```

In [46]:
```
# 모델을 불러온다.
load_model = tf.keras.models.load_model(savefile)
load_model.evaluate(test_images,  test_labels, verbose=2)
```

Out[46]:
```
313/313 - 7s - loss: 0.0275 - accuracy: 0.9913 - 7s/epoch - 23ms/step
[0.027460671961307526, 0.9912999868392944]
```

실습 01

- 10epochs를 돌려보기
- conv를 하나 삭제 해보기
- conv를 하나 추가 해보기
- GPU로 돌려보기(Google Colab 이용)