CNN(Convolution Neural Network) - 합성곱 신경망

학습 내용

- CNN의 기본 이해
- CNN을 실습을 통해 알아보기

목차

사전 설치

- 만약 tensorflow가 설치되어 있지 않다면 다음과 같이 설치가 가능
  `pip install tensorflow`

01 합성망 신경망 알아보기

목차로 이동하기

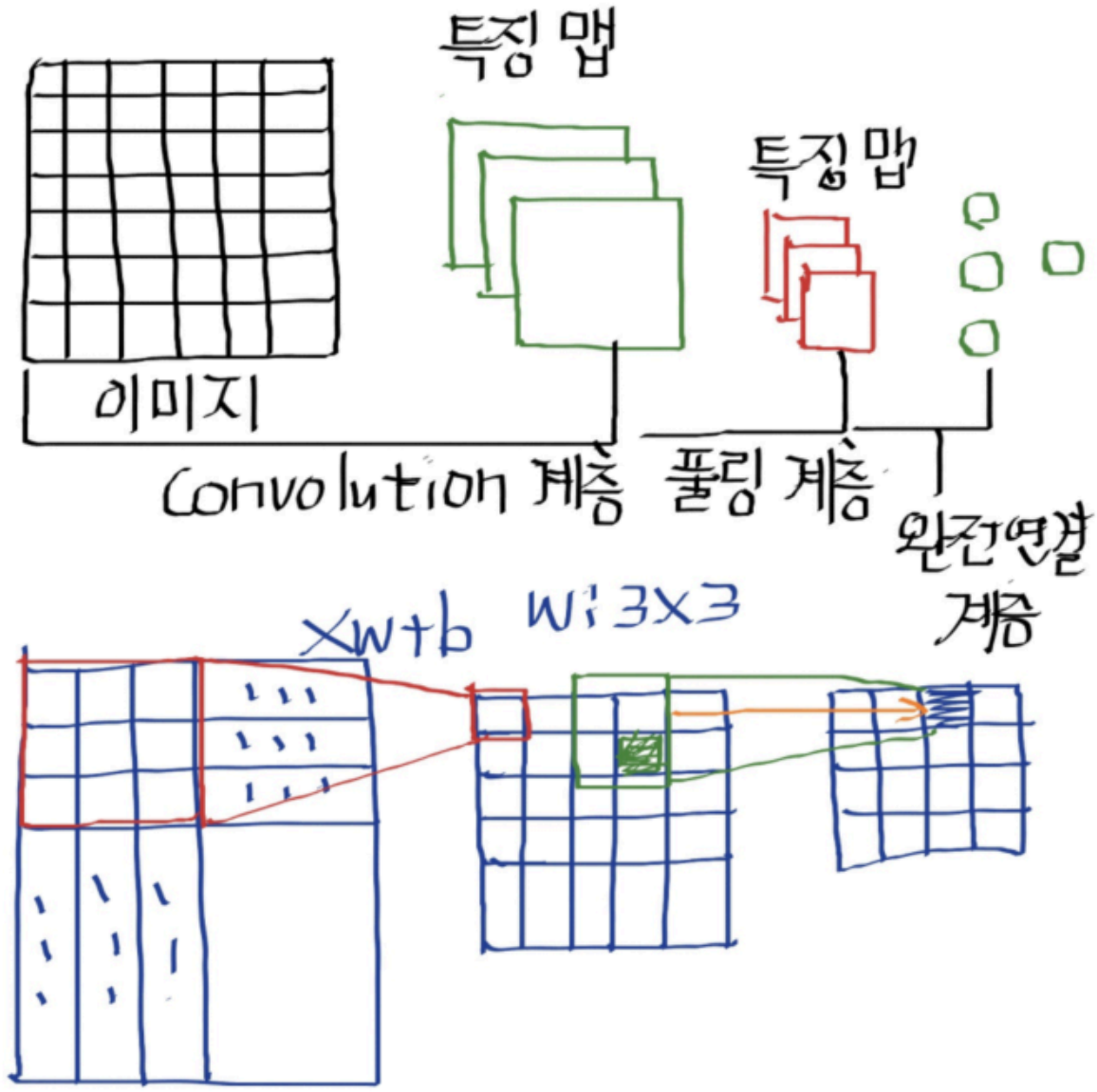
CNN은 무엇인가요?

- CNN은 Convolutional Neural Network의 약자
- 딥러닝 분야에서 주로 사용되는 인공신경망 모델의 한 종류
- CNN은 특히 컴퓨터 비전과 이미지 인식 분야에서 뛰어난 성능을 보이고 있음.
- CNN의 주요 구조는 컨볼루션 계층과 풀링 계층으로 이루어져 있음.
  - 컨볼루션 계층에서는 이미지에 필터를 적용하여 특징을 추출.
  - 풀링 계층에서는 이미지의 크기를 줄여 계산 효율성을 높임.
- CNN은 이미지, 비디오 등의 시각 데이터 처리에 탁월한 성능을 보이며, 컴퓨터 비전 분야에서 널리 활용되고 있음.

In [5]:
```
from IPython.display import display, Image
import os, warnings

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

In [6]:
```
display(Image(filename="img/cnn.png"))
```



In [7]:
```
import tensorflow as tf
from tensorflow.keras import models
from tensorflow.keras import layers
import matplotlib.pyplot as plt

print(tf.__version__)
```

```
2.16.1
```

우리가 구성할 모델

- Conv :3x3 필터, 32개의 필터개수, 입력 이미지 (28, 28, 1)
- Maxpooling (2,2)
- Conv :3x3 필터, 64개의 필터개수
- Maxpooling (2,2)

사용할 함수

Conv2D()와 MaxPool2D() 클래스의 상세 형식

```
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 [8]:
```
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical

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

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mn
ist.npz
11490434/11490434 ——————————————————————————— 5s 0us/step
```

03 모델 만들기

목차로 이동하기

In [9]:
```
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)))
```

완전 연결층(FCL) 추가

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

CNN 구조 알아보기

In [11]:
```
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	
conv2d (Conv2D)	(None, 26, 26, 32)	
max_pooling2d (MaxPooling2D)	(None, 13, 13, 32)	
conv2d_1 (Conv2D)	(None, 11, 11, 64)	
max_pooling2d_1 (MaxPooling2D)	(None, 5, 5, 64)	
flatten (Flatten)	(None, 1600)	
dense (Dense)	(None, 64)	
dense_1 (Dense)	(None, 10)	

Total params: 121,930 (476.29 KB)

Trainable params: 121,930 (476.29 KB)

Non-trainable params: 0 (0.00 B)

- (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.

04. 모델에 맞춰 데이터 전처리

목차로 이동하기

데이터 전처리

- 이미지 Reshape
- 정규화 0~1사이의 값으로 변경

In [12]:
```
# 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 [13]:
```
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)
```

05 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 [14]:
```
%%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 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13s 13ms/step - accuracy: 0.8778 -
loss: 0.3841 - val_accuracy: 0.9823 - val_loss: 0.0554
Epoch 2/5
938/938 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13s 13ms/step - accuracy: 0.9828 -
loss: 0.0566 - val_accuracy: 0.9886 - val_loss: 0.0366
Epoch 3/5
938/938 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13s 14ms/step - accuracy: 0.9896 -
loss: 0.0345 - val_accuracy: 0.9903 - val_loss: 0.0298
Epoch 4/5
938/938 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13s 14ms/step - accuracy: 0.9915 -
loss: 0.0251 - val_accuracy: 0.9898 - val_loss: 0.0295
Epoch 5/5
938/938 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12s 13ms/step - accuracy: 0.9941 -
loss: 0.0185 - val_accuracy: 0.9925 - val_loss: 0.0258
CPU times: total: 1min 11s
Wall time: 1min 3s
```

In [15]:
```
test_loss, test_acc = model.evaluate(test_images, test_labels)
print(test_acc)
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9901 - l
oss: 0.0324
0.9925000071525574
```

In [16]:
```
hist.history['loss']
```

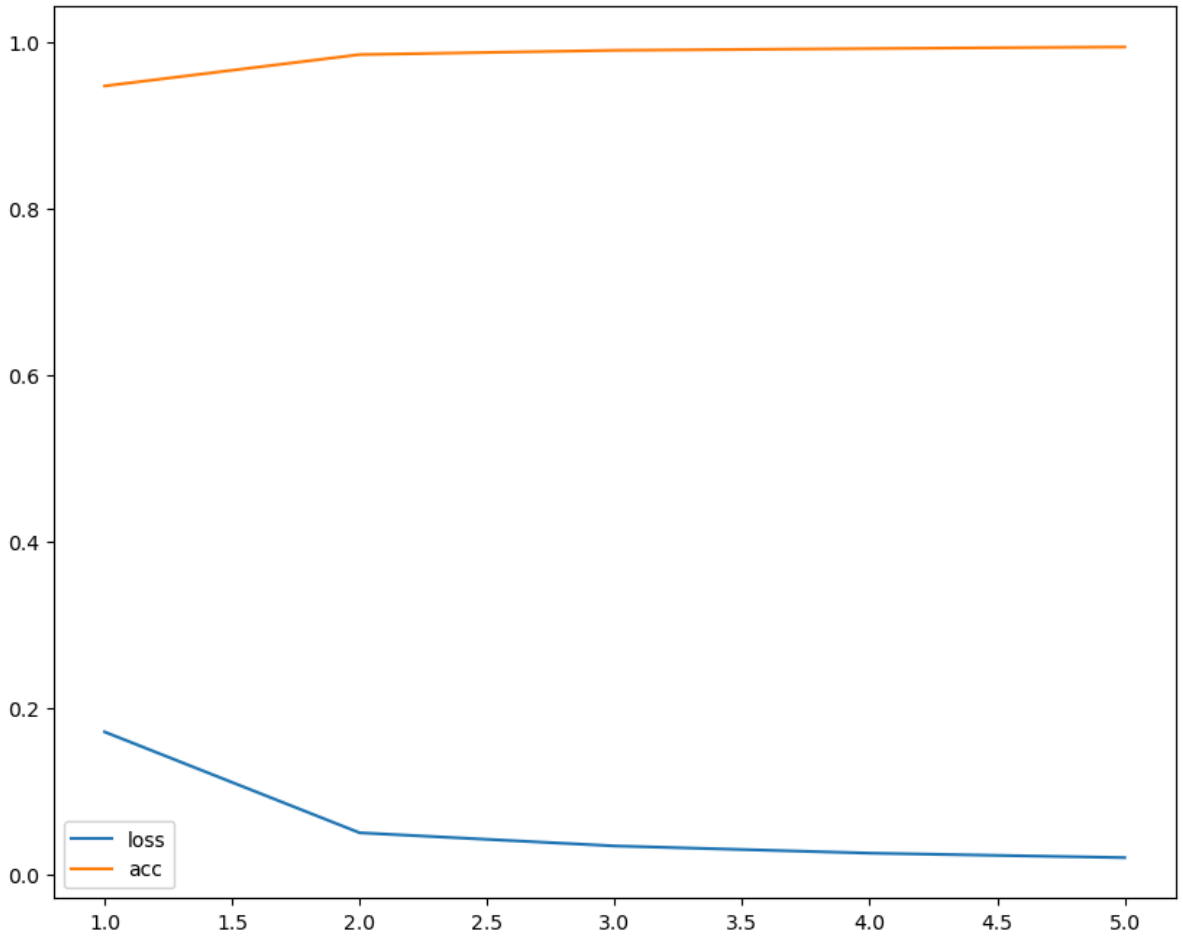
Out[16]:
```
[0.17156606912612915,
 0.050500720739364624,
 0.03471164405345917,
 0.026060106232762337,
 0.020667018368840218]
```

06. 모델 결과 시각화

목차로 이동하기

In [18]:
```
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[18]:
```
<matplotlib.legend.Legend at 0x1b081d6f450>
```



07 모델 저장 및 불러오기

목차로 이동하기

In [19]:
```
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)
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.
```

Out[19]:
```
['my_model_mnist.h5']
```

In [20]:
```
# 모델을 불러온다.
load_model = tf.keras.models.load_model(savefile)
load_model.evaluate(test_images,  test_labels, verbose=2)
```

```
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be buil
t. `model.compile_metrics` will be empty until you train or evaluate the model.
313/313 - 1s - 3ms/step - accuracy: 0.9925 - loss: 0.0258
```

Out[20]:
```
[0.025776531547307968, 0.9925000071525574]
```

실습 01

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