CNN(Convolution Neural Network) - 합성곱 신경망

학습 내용

- CNN의 기본 이해
- CNN을 실습을 통해 알아보기

목차

01 합성망 신경망 알아보기
02 MNIST 데이터 셋 - CNN 모델 구축
03 모델 학습 및 평가
04 모델 구축 및 학습, 평가

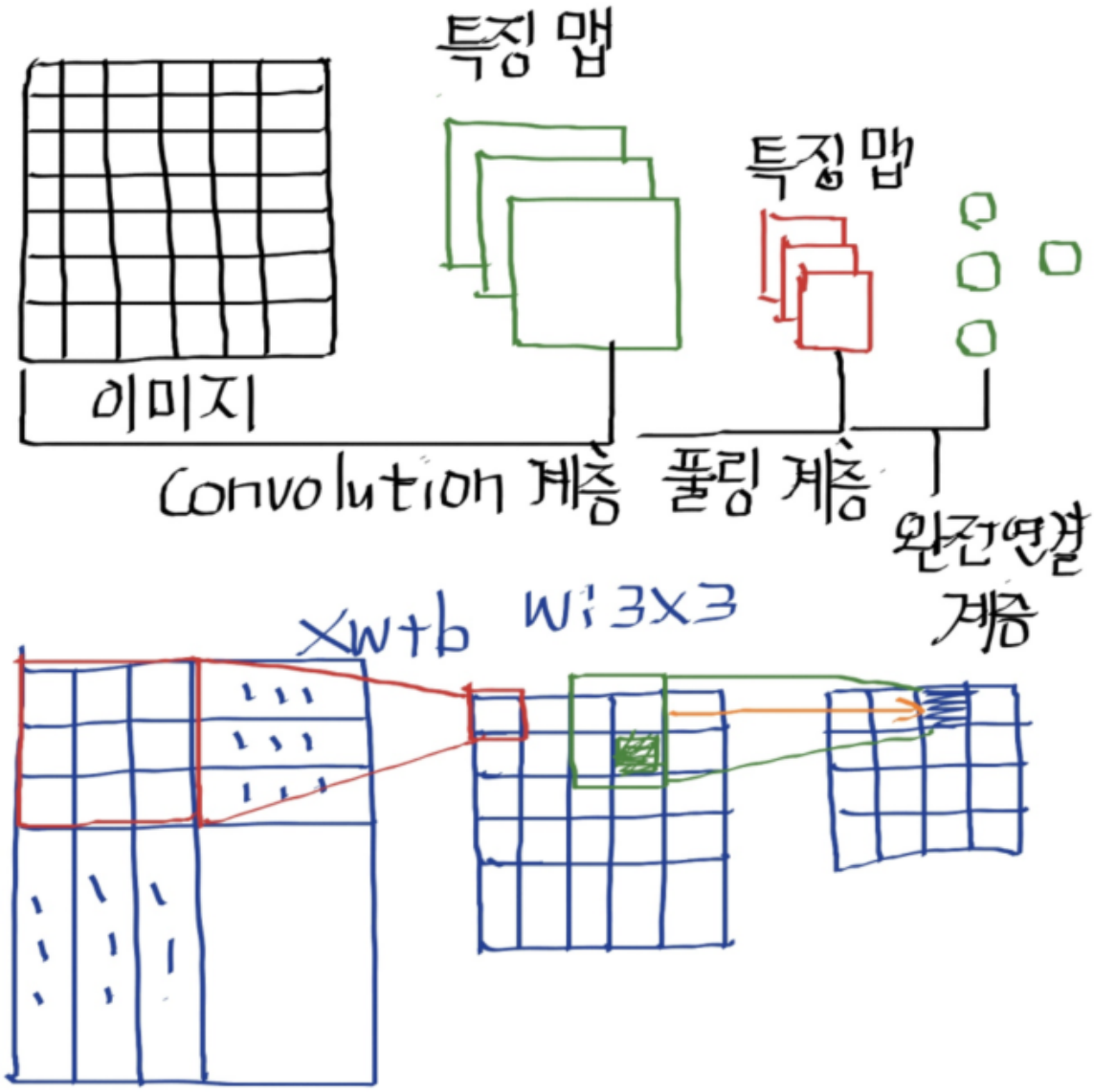
In [1]:

```
from IPython.display import display, Image
import os, warnings

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

In [2]:

```
display(Image(filename="img/cnn.png"))
```



01 합성망 신경망 알아보기

In [4]:

```
import tensorflow as tf
from tensorflow.keras import models
from tensorflow.keras import layers

print(tf.__version__)
```

2.9.0

```
# 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)

```
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
)
```

In [5]:

```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3),
                        activation='relu', input_shape=(28, 28, 1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
```

컨브넷 구조 알아보기

In [6]:

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 32)        320

 max_pooling2d (MaxPooling2D  (None, 13, 13, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 11, 11, 64)        18496

 max_pooling2d_1 (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.
- (3,3,64)를 완전 연결 네트워크에 펼쳐 연결한다.

완전 연결층(FCL) 추가

In [7]:

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

In [8]:

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 32)        320

 max_pooling2d (MaxPooling2D  (None, 13, 13, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 11, 11, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 5, 5, 64)         0
 2D)

 flatten (Flatten)           (None, 1600)              0

 dense (Dense)               (None, 64)                102464

 dense_1 (Dense)             (None, 10)                650

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

02 MNIST 데이터 셋 - CNN 모델 구축

- MNIST 데이터 셋 준비

In [9]:

```
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 (https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz)
11490434/11490434 [==============================] - 2s 0us/step
```

In [10]:

```
# 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 [11]:

```
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)

비용함수, 최적화 함수 구성

- 비용함수와 최적화 함수 지정
- 비용함수 : categorical_crossentropy
- 최적화 함수 : rmsprop

In [12]:

```
%%time

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

```
Epoch 1/5
938/938 [==============================] - 24s 25ms/step - loss: 0.1731 - accuracy:
0.9472 - val_loss: 0.0584 - val_accuracy: 0.9813
Epoch 2/5
938/938 [==============================] - 24s 25ms/step - loss: 0.0521 - accuracy:
0.9841 - val_loss: 0.0370 - val_accuracy: 0.9882
Epoch 3/5
938/938 [==============================] - 23s 25ms/step - loss: 0.0359 - accuracy:
0.9889 - val_loss: 0.0336 - val_accuracy: 0.9886
Epoch 4/5
938/938 [==============================] - 23s 25ms/step - loss: 0.0269 - accuracy:
0.9916 - val_loss: 0.0300 - val_accuracy: 0.9907
Epoch 5/5
938/938 [==============================] - 23s 24ms/step - loss: 0.0214 - accuracy:
0.9933 - val_loss: 0.0250 - val_accuracy: 0.9913
CPU times: total: 13min 43s
Wall time: 1min 56s
```

Out[12]:

```
<keras.callbacks.History at 0x248c6d570a0>
```

In [13]:

```
test_loss, test_acc = model.evaluate(test_images, test_labels)
print(test_acc)
```

```
313/313 [==============================] - 1s 4ms/step - loss: 0.0250 - accuracy: 0.
9913
0.9912999868392944
```

실습 01

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