CNN(Convolution Neural Network)

학습 내용

- CNN의 기본 용어 이해
- CNN에 대해 이해해 보기
- CNN을 활용한 신경망 학습시켜 보기

기본 용어 이해

커널(kernel) or 필터(filter)

- 입력층의 원도우를 은닉층의 뉴런하나로 압축할 때 사용된다.
- 기본 신경망을 이용한다면 모든 뉴런 연결(28 x 28 = 784)을 해야 한다. 784개의 가중치 필요
- 커널을 사용한다면 3 x 3의 가중치만 사용하면 된다. 가중치 사용이 줄어듬.
- 단, 복잡한 특징을 잡기에 부족하므로 보통 커널을 여러개 사용

채널(Channel)

- 하나의 색은 RGB의 세가지 색의 결합으로 이루어진다. 따라서 컬러이미지는 가로, 세로, 색의 3개의 채널로 구성된다.

스트라이드(Stride)

- 필터는 입력되는 데이터(이미지)를 위를 이동하면서 합성곱을 계산합니다. 이때 필터의 순회하는 간격을 의미합니다.

패딩(Padding)

- Convolution 레이어에서 Filter와 Stride로 인해 기존의 데이터가 줄어드는 것을 방지하는 방법. 입력데이터의 외각에 지정된 픽셀만큼 특정값으로 채운다. 보통 패딩 값으로 0으로 채워진다.

Pooling 레이어

- 컨볼류션층의 출력 데이터를 입력으로 받아, 출력 데이터(Activation Map)의 크기를 줄이거나 특정 데이터를 강조하기 위해 사용. Max Pooling과 Average Pooling,Min Pooling등이 있음.
- 학습 파라미터 없음.
- Pooling 레이어를 통해 행렬 크기 감소
- Pooling 레이어를 통해 채널 수 변경 없음.

- 이미지 인식 분야에서는 매우 강력한 성능을 발휘하고 있다.
- 1998년 얀 레쿤(YannLecun)교수가 소개한 이래로 사용되고 있는 신경망이다.
- CNN은 기본적으로 컨볼루션 계층과 풀링(pooling layer)로 구성된다.
- 평면의 경우, 행렬에서 지정한 영역의 값을 하나로 압축시킨다.
- 하나의 값으로 압축할 때,
  - 컨볼루션은 가중치와 편향을 적용하고,
  - 풀링 계층은 단순한 값들중의 하나를 가져오는 방식을 취한다.

In [1]:

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

CNN의 이해

컨볼루션 계층(convolution layer)

- 2D 컨볼루션의 경우, 2차원 평면 행렬에서 지정한 영역의 값을 하나로 압축.

풀링 계층(pooling layer)

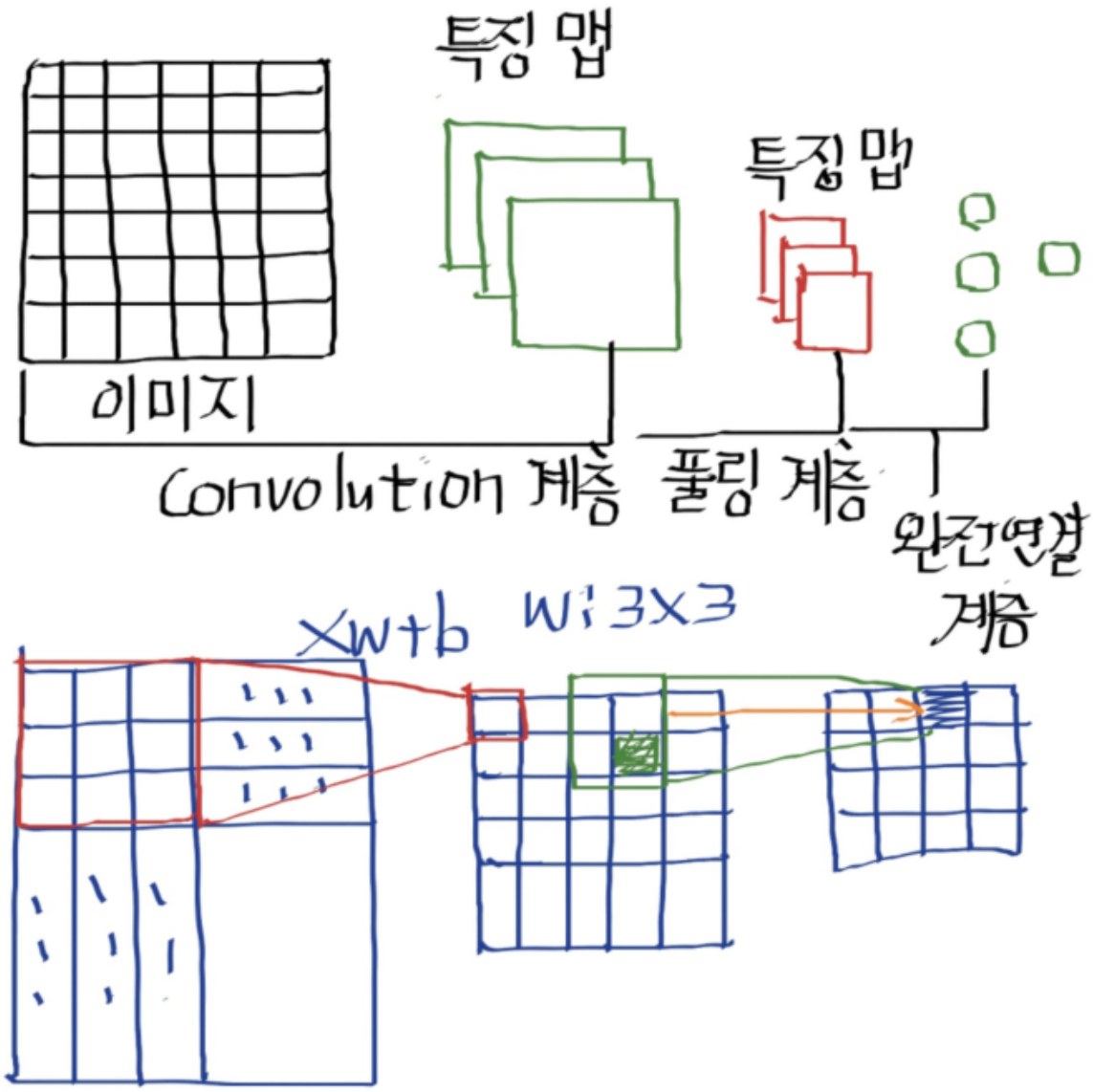
- 값들 중에 하나를 선택해서 가져온다. (최대값, 최소값, 평균값)

스트라이드(stride)

REF : https://towardsdatascience.com/types-of-convolutions-in-deep-learning-717013397f4d (https://towardsdatascience.com/types-of-convolutions-in-deep-learning-717013397f4d)

In [2]:

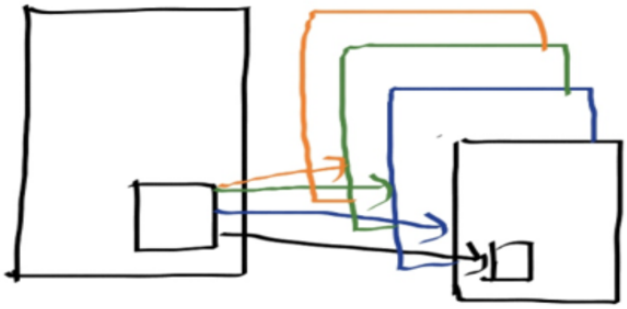
```
display(Image(filename="img/cnn.png"))
```



In [3]:

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

▶ **복잡한 특징을 가진 이미지는 분석하기에 커널이 부족할 수 있으므로 보통 커널을 여러 개 사용**

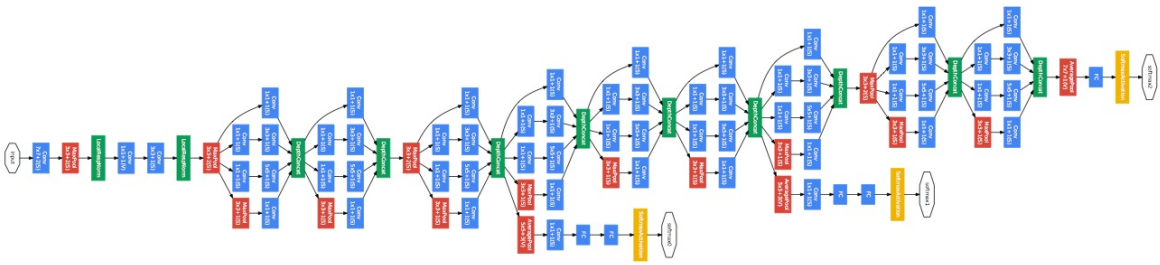


구글의 인셉션 모델

- 2014년 간신히 VGG 모델을 제치고 ImageNet에서 1등을 차지한 모델

In [4]:

```
display(Image(filename="img/google-inception.jpg"))
```



구글 인셉션 모델의 계층

- 작은 컨볼루션 계층을 매우 많이 연결.
- 구성이 꽤 복잡하다. 구현이 조금 까다로움.
  - 파란색-컨볼루션
  - 빨간색-풀링계층

01 데이터 가져오기

In [10]:

```
# 이미지 처리 분야에서 가장 유명한 신경망 모델인 CNN 을 이용
from keras import layers
from keras import models

from keras.datasets import mnist
from keras.utils import to_categorical

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

In [11]:

```
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

train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)
```

02 신경망 모델 구성

- (28, 28, 1) 가로 세로 : 28 픽셀, RGB 단색
- (28, 28, 3) 가로 세로 : 28 픽셀, RGB 3색
- MNIST는 회색조의 이미지로 색상이 한개

In [12]:

```
# 기존 모델에서는 입력 값을 28x28 하나의 차원으로 구성하였으나,
# CNN 모델을 사용하기 위해 2차원 평면과 특성치의 형태를 갖는 구조로 만듭니다.
# None는 한번의 학습당 사용할 입력데이터의 개수,
# 마지막 차원 1은 특징의 개수. MNIST는 회색조의 이미지로 색상이 한개

model = models.Sequential()

### L1 계층
model.add(layers.Conv2D(32, (3, 3),
                        activation='relu',
                        padding="same",  #  기본값 : valid
                        input_shape=(28, 28, 1)))
model.add(layers.MaxPooling2D((2, 2)))

### L2 계층
model.add(layers.Conv2D(64, (3, 3),
                        activation='relu',
                        padding="same") )
model.add(layers.MaxPooling2D((2, 2)))

### L3 계층
model.add(layers.Conv2D(64, (3, 3),
                        activation='relu',
                       padding="same"))
model.add(layers.MaxPooling2D((2, 2)))
```

참고할만한 내용

- url : https://mlnotebook.github.io/post/CNN1/ (https://mlnotebook.github.io/post/CNN1/)

```
model.add(layers.Conv2D(32, (3, 3),
                        activation='relu',
                        padding="same",
                        input_shape=(28, 28, 1)))
model.add(layers.MaxPooling2D((2, 2)))
```

L1 계층

컨볼루션

- [3 3]: 커널 크기
- 32: 필터 or 커널 갯수
- input_shape=(28, 28, 1) : 입력 이미지

풀링

- MaxPooling2D((2, 2))
- (?, 28, 28, 32) -> (?, 14, 14, 32)
- 32 (3 3) + 32(b) * 1 = 320(파라미터 개수)

L2 계층

model.add(layers.Conv2D(64, (3, 3), activation='relu', padding="valid")) model.add(layers.MaxPooling2D((2, 2)))

컨볼루션

- [3 3]: 커널 크기
- 64: 필터 or 커널 갯수
- (?, 14, 14, 32) : 입력

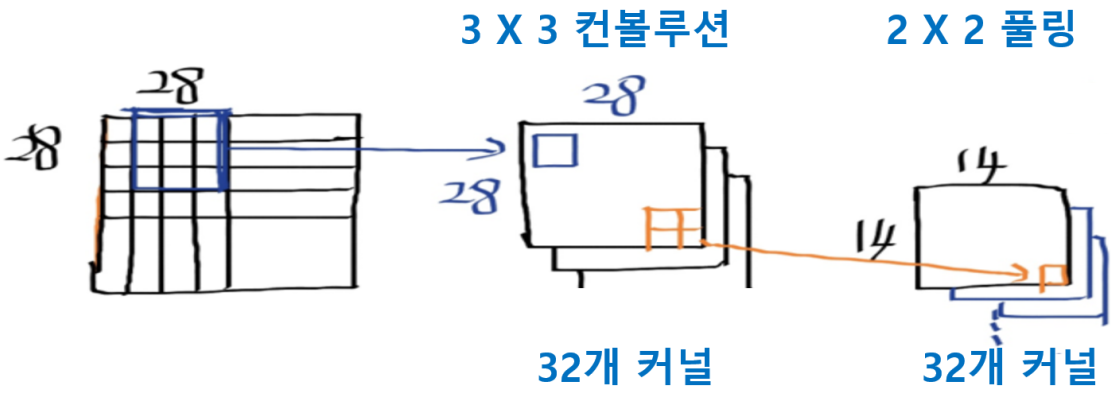
풀링

- MaxPooling2D((2, 2))
- (?, 14, 14, 64) -> (?, 7, 7, 64)
- (32채널) 64(필터수) (3 3) + 64(b) 1 = 18496(파라미터 개수)

In [13]:

```
display(Image(filename="img/L1_Cnn.png"))
```

▶ L1 계층 구성



- 원도우의 크기 3 X 3 의 32개 커널을 사용

Pooling

- 2 X 2로 하는 풀링 계층
- MaxPooling2D(pool_size=(2, 2),strides=(1, 1), padding='valid')

strides : 풀링 창이 이동하는 거리

- 기타 : AveragePooling, GlobalMaxPooling, GlobalAveragePooling

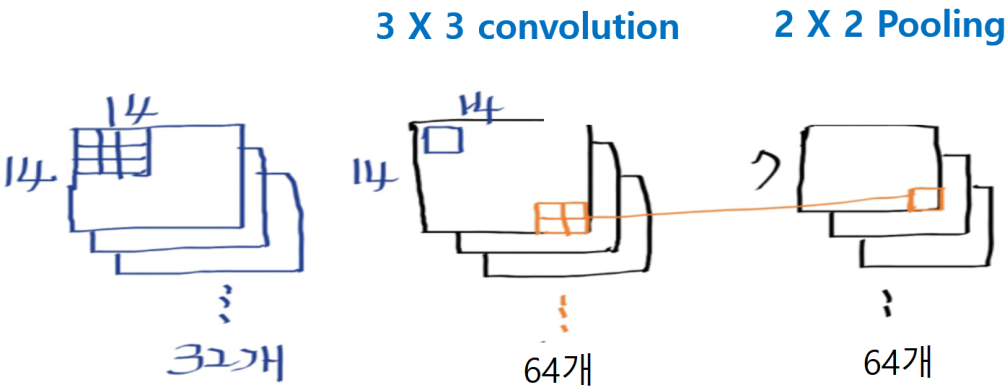
두번째 계층 구성

- 두 번째 컨볼루션 계층의 커널인 W2의 변수의 구성은 [3,3,32,64]이다.
- 32는 앞서 구성된 첫 번째 컨볼루션 계층의 커널 개수이다.
- 즉 출력층의 개수이며, 첫 번째 컨볼루션 계층이 찾아낸 이미지의 특징 개수라고 할 수 있다.

In [14]:

```
display(Image(filename="img/L2_Cnn.png"))
```

▶ L2 계층 구성



세번째 계층 구성

- 앞의 풀링 계층의 크기가 3 X 3 X 64 이므로, 1차원으로 만듬.
- 인접한 계층의 모든 뉴런과 상호 연결된 계층을 완전 연결 계층(fully connect layer)라 한다
- 마지막 층의 뉴런수는 576개.

In [15]:

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

In [16]:

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 28, 28, 32)        320
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 14, 14, 64)        18496
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 7, 7, 64)          0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 7, 7, 64)          36928
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 3, 3, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 576)               0
_________________________________________________________________
dense (Dense)                (None, 64)                36928
_________________________________________________________________
dense_1 (Dense)              (None, 10)                650
=================================================================
Total params: 93,322
Trainable params: 93,322
Non-trainable params: 0
_________________________________________________________________
```

비용함수, 최적화 함수

In [17]:

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

```
Epoch 1/5
938/938 [==============================] - 38s 40ms/step - loss: 0.4170 - accuracy:
0.8669
Epoch 2/5
938/938 [==============================] - 37s 39ms/step - loss: 0.0530 - accuracy:
0.9840
Epoch 3/5
938/938 [==============================] - 37s 40ms/step - loss: 0.0305 - accuracy:
0.9902
Epoch 4/5
938/938 [==============================] - 44s 47ms/step - loss: 0.0219 - accuracy:
0.99270s - loss: 0
Epoch 5/5
938/938 [==============================] - 52s 55ms/step - loss: 0.0178 - accuracy:
0.9941
```

Out[17]:

```
<tensorflow.python.keras.callbacks.History at 0x2ba9e1f5be0>
```

결과 확인

In [18]:

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

```
313/313 [==============================] - 3s 9ms/step - loss: 0.0280 - accuracy: 0.
9919
0.9919000267982483
```

(실습과제)

1. AdamOptimizer를 RMSPropOptimizer로 변경해서 해보기

(추가과제)

2. RMSPropOptimizer와 기타 최적화 함수를 알아보기(적용해보기) 어떤 것이 성능이 가장 좋은가?

REF

- 컨볼루션 참고 :
  - https://hackernoon.com/visualizing-parts-of-convolutional-neural-networks-using-keras-and-cats-5cc01b214e59 (https://hackernoon.com/visualizing-parts-of-convolutional-neural-networks-using-keras-and-cats-5cc01b214e59)
  - https://mlblr.com/includes/research/index.html (https://mlblr.com/includes/research/index.html) : Convolution 이해, 다양한 논문

- 국내: CNN 용어 및 계산
  - http://taewan.kim/post/cnn/#1-cnn%EC%9D%98-%EC%A3%BC%EC%9A%94-%EC%9A%A9%EC%96%B4-%EC%A0%95%EB%A6%AC (http://taewan.kim/post/cnn/#1-cnn%EC%9D%98-%EC%A3%BC%EC%9A%94-%EC%9A%A9%EC%96%B4-%EC%A0%95%EB%A6%AC)
- 인셉션 관련 : https://sunghan-kim.github.io/ml/3min-dl-ch11/# (https://sunghan-kim.github.io/ml/3min-dl-ch11/)

In [ ]: