

## CNN(Convolution Neural Network) 추가 실습

### 학습 내용

- CNN의 추가 실습 해보기
  - adam 0.9919
  - rmsprop -> adam으로 변경 [] - [0.9876]
  - adam, 마지막 pooling 없애기 [0.9876] - [0.9901]
  - sigmoid, 마지막 pooling 없애기 [0.9901]- [0.1134]
  - leakyRelu 적용, 마지막 pooling 없애기 [0.9901]- [0.990]
  - Adamax, relu [L123전부 사용]- [0.9929]

In [15]:

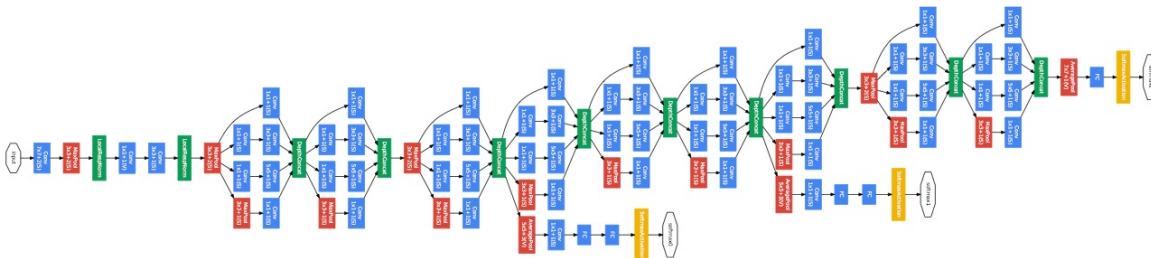
```
from IPython.display import display, Image
```

### 구글의 인셉션 모델

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

In [16]:

```
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 [17]:

```
# 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 [18]:

```
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 [7]:

```
# 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)))
```

In [10]:

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

In [11]:

```
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 28, 28, 32)	320
max_pooling2d (MaxPooling2D)	(None, 14, 14, 32)	0
conv2d_1 (Conv2D)	(None, 14, 14, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 7, 7, 64)	0
conv2d_2 (Conv2D)	(None, 7, 7, 64)	36928
max_pooling2d_2 (MaxPooling2D)	(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 [12]:



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

Epoch 1/5

938/938 [=====] - 159s 169ms/step - loss: 0.1847 - accurac

y: 0.9413

Epoch 2/5

938/938 [=====] - 145s 154ms/step - loss: 0.0471 - accurac

y: 0.9855

Epoch 3/5

938/938 [=====] - 169s 180ms/step - loss: 0.0310 - accurac

y: 0.9901 - loss: 0.0310 - accuracy:

Epoch 4/5

938/938 [=====] - 170s 181ms/step - loss: 0.0233 - accurac

y: 0.9926

Epoch 5/5

938/938 [=====] - 158s 168ms/step - loss: 0.0189 - accurac

y: 0.9941

Out[12]:

&lt;tensorflow.python.keras.callbacks.History at 0x1d3a8155a90&gt;

## 결과 확인

In [19]:



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

313/313 [=====] - 15s 49ms/step - loss: 0.0255 - accuracy:

0.9919

0.9919000267982483

In [19]:



```
from keras.layers.advanced_activations import LeakyReLU
```

In [20]:



```
# 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  
### Convolution  
model.add(layers.Conv2D(32, (3,3),  
                        padding='same', # 기본값 : valid  
                        input_shape = (28,28,1) ))  
model.add(LeakyReLU(0.2))  
### Pooling - 다운 샘플링  
model.add(layers.MaxPooling2D( (2,2) )) # 2x2  
  
### L2 계층  
### Convolution 3x3, 64필터수, padding (같은 크기의 특징맵)  
model.add(layers.Conv2D(64, (3,3),  
                        padding='same' ) )  
model.add(LeakyReLU(0.2))  
### Pooling - 다운 샘플링  
model.add(layers.MaxPooling2D( (2,2) ))  
  
### L3 계층  
### Convolution 3x3, 64필터수, padding (같은 크기의 특징맵)  
model.add(layers.Conv2D(64, (3,3),  
                        padding='same' ) )  
model.add(LeakyReLU(0.2))  
  
### Pooling - 다운 샘플링  
# model.add(layers.MaxPooling2D( (2,2) ))  
  
model.add(layers.Flatten())  
model.add(layers.Dense(64, activation='relu'))  
model.add(layers.Dense(10, activation='softmax'))
```

In [21]:



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

Epoch 1/5

938/938 [=====] - 173s 185ms/step - loss: 0.1463 - accuracy: 0.9547

Epoch 2/5

938/938 [=====] - 143s 153ms/step - loss: 0.0452 - accuracy: 0.9864

Epoch 3/5

938/938 [=====] - 125s 134ms/step - loss: 0.0311 - accuracy: 0.9901

Epoch 4/5

938/938 [=====] - 106s 113ms/step - loss: 0.0248 - accuracy: 0.9924 - loss: 0.0

Epoch 5/5

938/938 [=====] - 91s 97ms/step - loss: 0.0215 - accuracy: 0.9934

Out[21]:

<tensorflow.python.keras.callbacks.History at 0x1d3a99c4e80>

In [22]:



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

313/313 [=====] - 7s 22ms/step - loss: 0.0318 - accuracy: 0.9896  
0.9896000027656555

## REF

- 컨볼루션 참고 :

- <http://bitly.kr/jereFi6> (<http://bitly.kr/jereFi6>)
- <http://bitly.kr/rX6s5iw> (<http://bitly.kr/rX6s5iw>)
- <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://mlbl.com/includes/research/index.html> (<https://mlbl.com/includes/research/index.html>) : Convolution 이해, 다양한 논문
- 국내(<http://bitly.kr/wzCehHv> (<http://bitly.kr/wzCehHv>)) : 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 [ ]:

