

CNN(Convolution Neural Network) 추가 실습

학습 내용

- CNN의 파라미터를 변경해가며 추가 실습 (최적화 함수를 leaky relu를 사용)

In [1]:

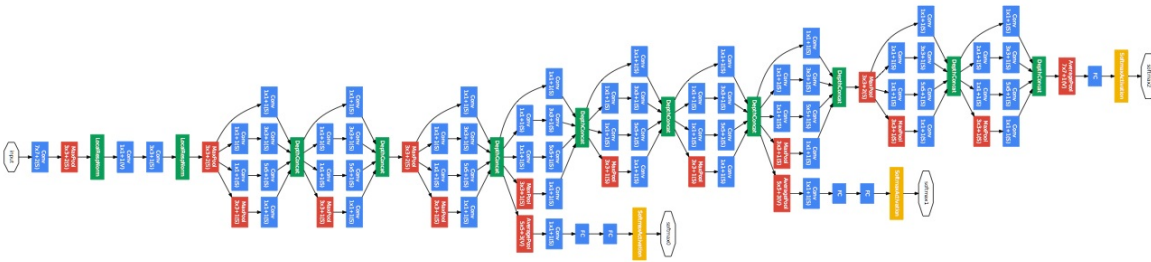
```
from IPython.display import display, Image
```

구글의 인셉션 모델

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

In [2]:

```
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 [4]:

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

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

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

In [5]:



```
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 [6]:



```
# 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 [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, 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 [9]:



```
%%time
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
model.fit(train_images, train_labels, epochs=3, batch_size=64)
```

Epoch 1/3
938/938 [=====] - 71s 75ms/step - loss: 0.1864 - accuracy: 0.9429
Epoch 2/3
938/938 [=====] - 65s 69ms/step - loss: 0.0494 - accuracy: 0.9844
Epoch 3/3
938/938 [=====] - 63s 67ms/step - loss: 0.0351 - accuracy: 0.9890
Wall time: 3min 19s

Out[9]:

<tensorflow.python.keras.callbacks.History at 0x24701610850>

결과 확인

In [10]:



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

```
313/313 [=====] - 4s 11ms/step - loss: 0.0302 - accuracy:
0.9893 0s - loss:
0.989300012588501
```

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

In [13]:



```
from tensorflow.keras.layers import LeakyReLU
```

- `tf.nn.leaky_relu(features, alpha=0.2, name=None)`
 - `alpha` : $x < 0$ 일때의 활성화 함수의 기울기

In [18]:



```
# 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(alpha=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(alpha=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(alpha=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 [19]:



```
model.summary()
```

Model: "sequential_3"

Layer (type)	Output Shape	Param #
conv2d_7 (Conv2D)	(None, 28, 28, 32)	320
leaky_re_lu_3 (LeakyReLU)	(None, 28, 28, 32)	0
max_pooling2d_5 (MaxPooling2D)	(None, 14, 14, 32)	0
conv2d_8 (Conv2D)	(None, 14, 14, 64)	18496
leaky_re_lu_4 (LeakyReLU)	(None, 14, 14, 64)	0
max_pooling2d_6 (MaxPooling2D)	(None, 7, 7, 64)	0
conv2d_9 (Conv2D)	(None, 7, 7, 64)	36928
leaky_re_lu_5 (LeakyReLU)	(None, 7, 7, 64)	0
flatten_2 (Flatten)	(None, 3136)	0
dense_4 (Dense)	(None, 64)	200768
dense_5 (Dense)	(None, 10)	650
Total params: 257,162		
Trainable params: 257,162		
Non-trainable params: 0		

In [16]:



```
%%time

model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
model.fit(train_images, train_labels, epochs=3, batch_size=64)
```

Epoch 1/3
938/938 [=====] - 68s 72ms/step - loss: 0.1502 - accuracy: 0.9538
Epoch 2/3
938/938 [=====] - 70s 75ms/step - loss: 0.0438 - accuracy: 0.9866
Epoch 3/3
938/938 [=====] - 68s 72ms/step - loss: 0.0316 - accuracy: 0.9899
Wall time: 3min 26s

Out[16]:

<tensorflow.python.keras.callbacks.History at 0x247017f4760>

In [17]:



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

```
313/313 [=====] - 4s 13ms/step - loss: 0.0329 - accuracy:
0.9894
0.9894000291824341
```

다양한 실습 확인

- 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]

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
 - <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/>)

history

- 2021.08.12 tf ver 2.5