TF2.0 신경망 만들기

- CNN 신경망 이해

In [1]:

```
!pip install -q tensorflow-gpu==2.0.0-rc1
```

```
     |████████████████████████████████| 380.5MB 39kB/s
     |████████████████████████████████| 501kB 13.9MB/s
     |████████████████████████████████| 4.3MB 41.3MB/s
```

In [0]:

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

In [2]:

```
print(tf.__version__)
```

```
2.0.0-rc1
```

In [5]:

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

train_images = train_images.reshape((60000, 28, 28, 1))
test_images = test_images.reshape((10000, 28, 28, 1))

# 픽셀 값을 0~1 사이로 정규화합니다.
train_images, test_images = train_images / 255.0, test_images / 255.0
train_images.shape, test_images.shape
```

Out[5]:

```
((60000, 28, 28, 1), (10000, 28, 28, 1))
```

합성곱 층 만들기

- 3D 텐서 : (이미지 높이, 이미지 너비, 컬러채널)
- MNIST 데이터 셋은 흑백이미지 : (28,28,1)

In [0]:

```
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)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
```

In [7]:

```
### 모델의 구조 출력
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 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 3, 3, 64)          36928
=================================================================
Total params: 55,744
Trainable params: 55,744
Non-trainable params: 0
_________________________________________________________________
```

- Total params 는 Param #을 전부 더해준것.

마지막 Dense 층 추가(FC)

- Dense 층은 벡터(1D)를 입력으로 받는다. 현재 입력은 3D이므로 이를 1D로 펼치기 위해 Flatten()를 사용

In [0]:

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

모델의 구조 확인

In [9]:

```
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 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 3, 3, 64)          36928
_________________________________________________________________
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
_________________________________________________________________
```

모델의 컴파일과 훈련

In [11]:

```
%%time

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(train_images, train_labels, epochs=5)
```

```
Train on 60000 samples
Epoch 1/5
60000/60000 [==============================] - 63s 1ms/sample - loss: 0.0353 - accur
acy: 0.9892
Epoch 2/5
60000/60000 [==============================] - 63s 1ms/sample - loss: 0.0249 - accur
acy: 0.9921
Epoch 3/5
60000/60000 [==============================] - 63s 1ms/sample - loss: 0.0198 - accur
acy: 0.9938
Epoch 4/5
60000/60000 [==============================] - 63s 1ms/sample - loss: 0.0163 - accur
acy: 0.9947
Epoch 5/5
60000/60000 [==============================] - 64s 1ms/sample - loss: 0.0127 - accur
acy: 0.9959
CPU times: user 8min 44s, sys: 22.8 s, total: 9min 7s
Wall time: 5min 16s
```

모델 평가

In [0]:

```
test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)
```

In [0]:

```
print(test_acc)
```

In [0]:

```

```

REF

- CNN : https://www.tensorflow.org/tutorials/images/cnn (https://www.tensorflow.org/tutorials/images/cnn)

In [0]:

```

```