Pooling Layer

CNN에 남은 마지막 퍼질 폴링입니다. Flatten 만을 이용한 모델의 코드는 이렇게 생겼습니다.

```
X = tf.keras.layers.Input(shape=[28, 28])
H = tf.keras.layers.Flatten()(X)
H = tf.keras.layers.Dense(84, activation='swish')(H)
Y = tf.keras.layers.Dense(10, activation='softmax')(H)
model = tf.keras.models.Model(X, Y)
model.compile(loss='scategorical_crossentropy', metrics='accuracy')
```

84 * (784 + 1) = 65940
10 * (84 + 1) = 850

```
model.summary()
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 28, 28)]          0
_________________________________________________________________
flatten (Flatten)            (None, 784)               0
_________________________________________________________________
dense (Dense)                (None, 84)                65940
_________________________________________________________________
dense_1 (Dense)              (None, 10)                850
=================================================================
Total params: 66,790
Trainable params: 66,790
Non-trainable params: 0
_________________________________________________________________
```

서머리라는 도구를 이용하면 모델의 요약 설명을 볼수 있는데요 그림의 오른쪽 처럼요. 플래튼 레이어 에서는 (28, 28) 2차원 형태의 숫자들을 한줄로 펼쳐서 784개의 변수(파란색네모)를 가지는 데이터를 출력합니다. 첫번째 히든 레이어는 84개 컬럼(초록색네모)을 가지는 데이터를 추척하고요. 마지막 레이어에서는 10개 컬럼(빨간색 네모)을 가지는 출력을 만듭니다. 첫번째 히든 레이어에서 84개 컬럼(초록색 네모)에 출력을 만드는 것은 84개의 출력이 있다는 말인데요. 하나의 수식을 위해서는 784개의 가중치 그리고 1개의 바이어스가 필요하므로 이 히든레이어에 구성하는데 필요한 가중치의 수는 65,940개(초록색네모)입니다. 마지막 레이어는 10개의 수식이 있고 하나의 수식을 위해서 84개의 가중치 그리고 1개의 바이어스가 필요하므로 마지막 출력층을 구성하는데 필요한 가중치수는 850개(빨간색 네모) 입니다. 850개의 가중치가 필요합니다.

이번에는 컬렉션 레이어가 추가 된 모델의 코드를 확인하겠습니다.

```
X = tf.keras.layers.Input(shape=[28, 28, 1])
H = tf.keras.layers.Conv2D(3, kernel_size=5, activation='swish')(X)
H = tf.keras.layers.Conv2D(6, kernel_size=5, activation='swish')(H)
H = tf.keras.layers.Flatten()(H)
H = tf.keras.layers.Dense(84, activation='swish')(H)
Y = tf.keras.layers.Dense(10, activation='softmax')(H)
model = tf.keras.models.Model(X, Y)
model.compile(loss='scategorical_cross
```

```
model.summary()
```

84 * (2400 + 1) = 201,684

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 28, 28, 1)]       0
_________________________________________________________________
conv2d (Conv2D)              (None, 24, 24, 3)         78
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 20, 20, 6)         456
_________________________________________________________________
flatten (Flatten)            (None, 2400)              0
_________________________________________________________________
dense (Dense)                (None, 84)                201684
_________________________________________________________________
dense_1 (Dense)              (None, 10)                850
=================================================================
Total params: 203,068
Trainable params: 203,068
Non-trainable params: 0
_________________________________________________________________
```

써머리 도구를 이용하여 모델의 요약 설명을 보면(그림의 하단 오른쪽) 플래튼 레이어(파란색 네모)의 입력으로 (20, 20, 6)(빨간색 네모)에 형태로 입력을 받죠. (20, 20) 이미지가 6장 있다는 말입니다. 한장의 400개의 숫자 있을 것이고 거기에 6을 곱하여 2400개(가운데 파란색네모)의 컬럼에 데이터를 출력합니다. 다음 히든 레이어에서 84개 컬럼(초록색 네모)에 출력을 만들고 있는데요. 이는 84개의 수식이 있다는 말이고, 이 하나의 수식을 위해서는 2400개의 가중치 1개의 바이어스가 필요하므로(수식) 히든 레이를 구성하는 데에 필요한 가중치에는 201,684개입니다.(우측 초록색 네모) 역시 모델 요약에 잘표시가 되어 있습니다. 컨벨루션 레이어에 출력된 특징맵의 개수가 증가하면 플래튼 이후 사용할 데이터에 컬럼 수가 증가하고 이는 곧 컴퓨터가 찾아야 하는 가중치의 증가를 의미합니다.

그래서 이러한 것을 해결하기위해 도입된 것이 **Pooling**입니다. 플래튼레이어 이후에 사용되는 가중치수를 작게 유지하기 위해 입력으로 사용할 컬럼수를 조정하는 것이 목적입니다. 폴링 레이어가 어떻게 가중치수를 줄이게 되는 지 확인해 보겠습니다.

폴링 레이어를 사용한 모델의 코드는 이렇습니다.(그림의 상단)

컨벌루션 레이어 이후에 맥스폴링 레이어를 추가해(빨간색 네모) 주었습니다. summary 를 확인해볼까요.(그림 하단 우측) 플래튼 메뉴에서는 입력으로 (4, 4)이미지 6장을 사용한는데요. 한장의 16개의 숫자가 있을 것이고, 16에 6을 곱하여 총 96개 컬럼(파란색 네모)에 데이터를 출력합니다. 다음 히든 레이어 에서는 84개의 컬럼에 출력을 만들고 있는데요. 그것은 84개의 수식이 있다는 얘기고 하나의 수식을 위해서는 16개의 가중치 그리고 1개의 바이어스가 필요하므로 (상단의 수식) 히든 레이어를 구성하는데에 필요한 가중치수 수는 8,148개입니다.(오른쪽 초록색 네모) 컨벨루션 레이어로 출력된 특징맵 이미지의 사이즈가 맥스폴링 레이어를 거치면서 절반으로 줄어드는 것을 확인할 수 있습니다. 맥스폴링 레이오를 통해 모델을 만들면 그냥 플래튼만 이용하여 만든 모델 보다도 가중치수가 적어질 수도 있다는 것이 놀라운 부분입니다.

아래는 폴링에 대한 설명입니다.

Pooling Layer란 Convolution Layer의 연산 결과로 출력된 Feature map의 사이즈를 감소시키면서도 특징을 보존하기 위한 과정이다.

일반적으로 Convolution Layer은 합성곱 연산 결과로 나온 Feature map의 non-linearity 성질을 주기 위해 활성화 함수 연산을 수행하는 것이 일반적입니다.

그 다음 Pooling Layer가 연산됩니다.

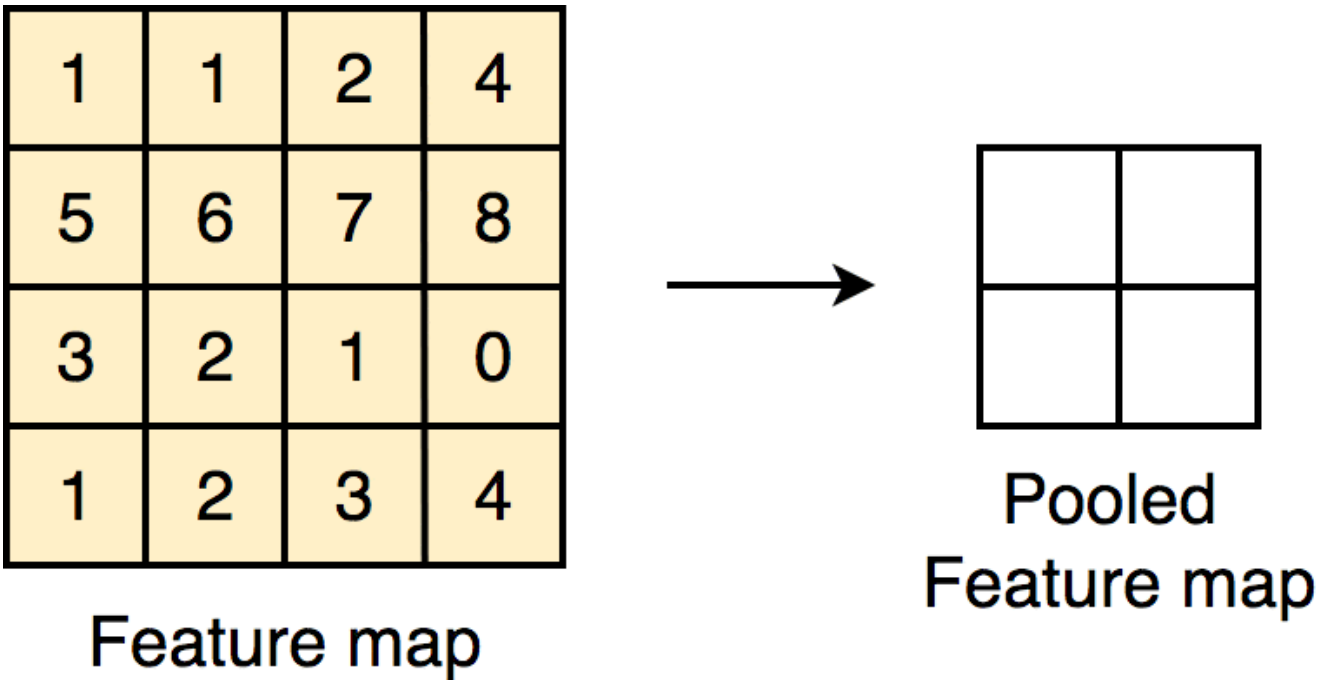
목적

1. 합성곱의 연산 결과로 나온 Feature map 크기 줄여서 신경망의 파라미터와 연산량을 줄이기 위해서
2. 신경망에 미세한 변화가 있어도 민감하지 않도록 하기 위해서 (Average Pooling)
3. local의 잡음을 제거하고 중요한 특징을 추출하기 위해서

원리

Max Pooling

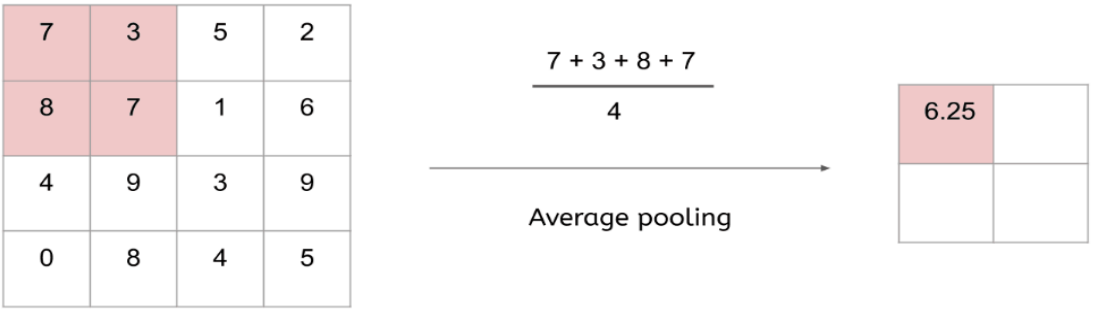
- Pooling Filter 내에서 가장 중요한 Feature를 추출하는 방식입니다.



- 위와 같은 방식으로 local에서 가장 큰 Feature를 추출하면서 Feature map의 크기를 줄입니다.
- 가장 큰 Feature를 추출하면서도 추출된 값보다 작은 덜 중요한 값들(잡음)을 제거하게 됩니다.

Average Pooling

- Pooling Filter 내의 Feature들의 평균을 구하는 방식입니다.



- 위와 같은 방식으로 locald에서 평균을 구하여 새로운 Feature map을 만듭니다.
- 평균을 구해 low-frequency 나 high-frequency 들을 감소시키는 역할을 합니다.

```python
# -*- coding: utf-8 -*-
import tensorflow as tf
import pandas as pd

(독립, 종속), _ = tf.keras.datasets.mnist.load_data()
독립 = 독립.reshape(60000, 28, 28, 1)
종속 = pd.get_dummies(종속)
print(독립.shape, 종속.shape)

X = tf.keras.layers.Input(shape=[28, 28, 1])
H = tf.keras.layers.Conv2D(3, kernel_size=5, activation='swish')(X)
H = tf.keras.layers.Conv2D(6, kernel_size=5, activation='swish')(H)
H = tf.keras.layers.Flatten()(H)
H = tf.keras.layers.Dense(84, activation='swish')(H)
Y = tf.keras.layers.Dense(10, activation='softmax')(H)
model = tf.keras.models.Model(X, Y)
model.compile(loss='categorical_crossentropy', metrics=['accuracy'])

model.summary()

84 * (2400 + 1) + 10 * (84 + 1) + 78 + 456

X = tf.keras.layers.Input(shape=[28, 28, 1])

H = tf.keras.layers.Conv2D(3, kernel_size=5, activation='swish')(X)
H = tf.keras.layers.MaxPool2D()(H)

H = tf.keras.layers.Conv2D(6, kernel_size=5, activation='swish')(H)
H = tf.keras.layers.MaxPool2D()(H)

H = tf.keras.layers.Flatten()(H)
H = tf.keras.layers.Dense(84, activation='swish')(H)
Y = tf.keras.layers.Dense(10, activation='softmax')(H)
model = tf.keras.models.Model(X, Y)
model.compile(loss='categorical_crossentropy', metrics=['accuracy'])

model.summary()

model.fit(독립, 종속, epochs=10)
```