Max Pooling 구현

최석재 *lingua@naver.com*

Pooling

Pooling

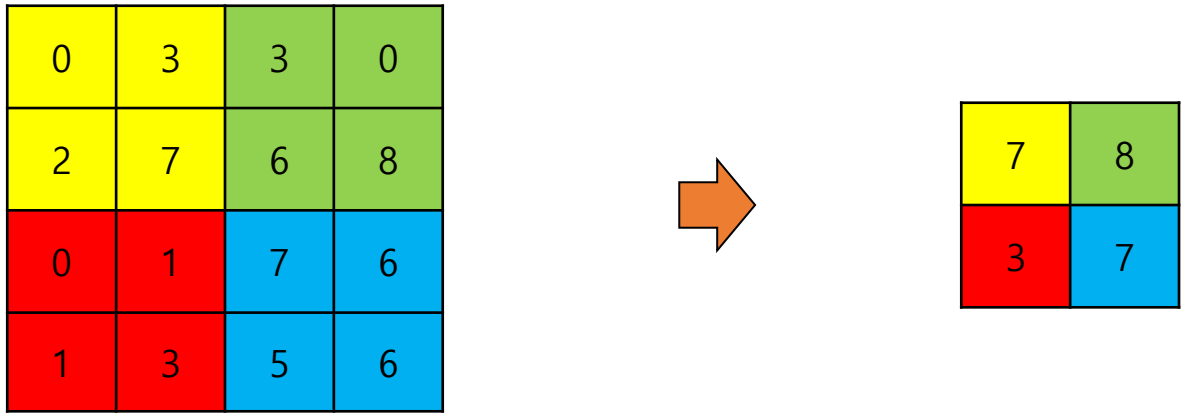
- 풀링 기법은 주로 CNN에서 사용되는 기법으로서
- 입력 이미지의 결과로 나오는 특성 맵(feature map)의 크기를 줄이면서
- 동시에 중요한 정보는 유지하는 효과를 갖는다

- 풀링을 사용하는 이유는 네트워크 파라미터 수와 계산량을 감소시킴으로써
- 훈련 데이터에 과적합되는 현상을 방지하고,
- 이미지 내 작은 변동이 미치는 영향을 줄임으로써, 변형이나 이동에 잘 대응하게 한다

- 주요 기법으로는
- 평균값을 사용하는 방법을 평균 풀링(average pooling)과
- 최댓값을 사용하는 방법을 맥스 풀링(max pooling)이 있으며,
- 맥스 풀링 방법이 더 선호된다

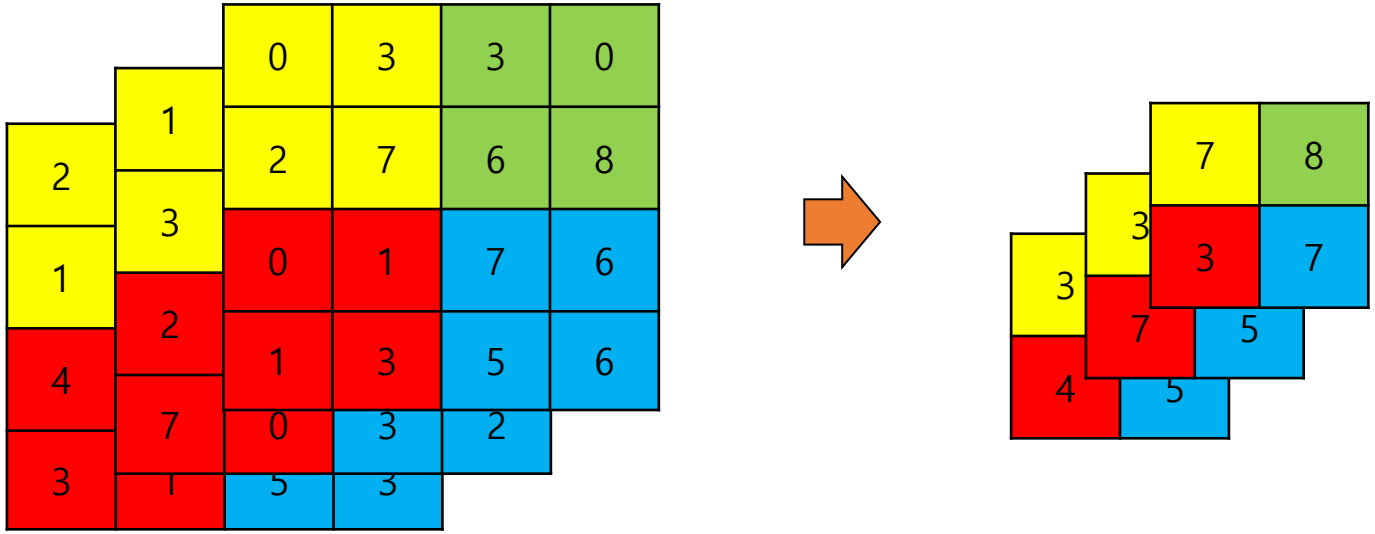
풀링의 특징 – 매개변수 없음

- 풀링의 특징을 살펴보면,
- 우선 합성곱 연산에서는 학습시켜야 하는 필터 가중치가 있는 반면,
- 풀링은 학습해야 할 별도의 매개변수 없이 단순히 최댓값이나 평균을 취하는 방법이다
- 즉, 학습할 가중치가 없다는 점이 특징이다



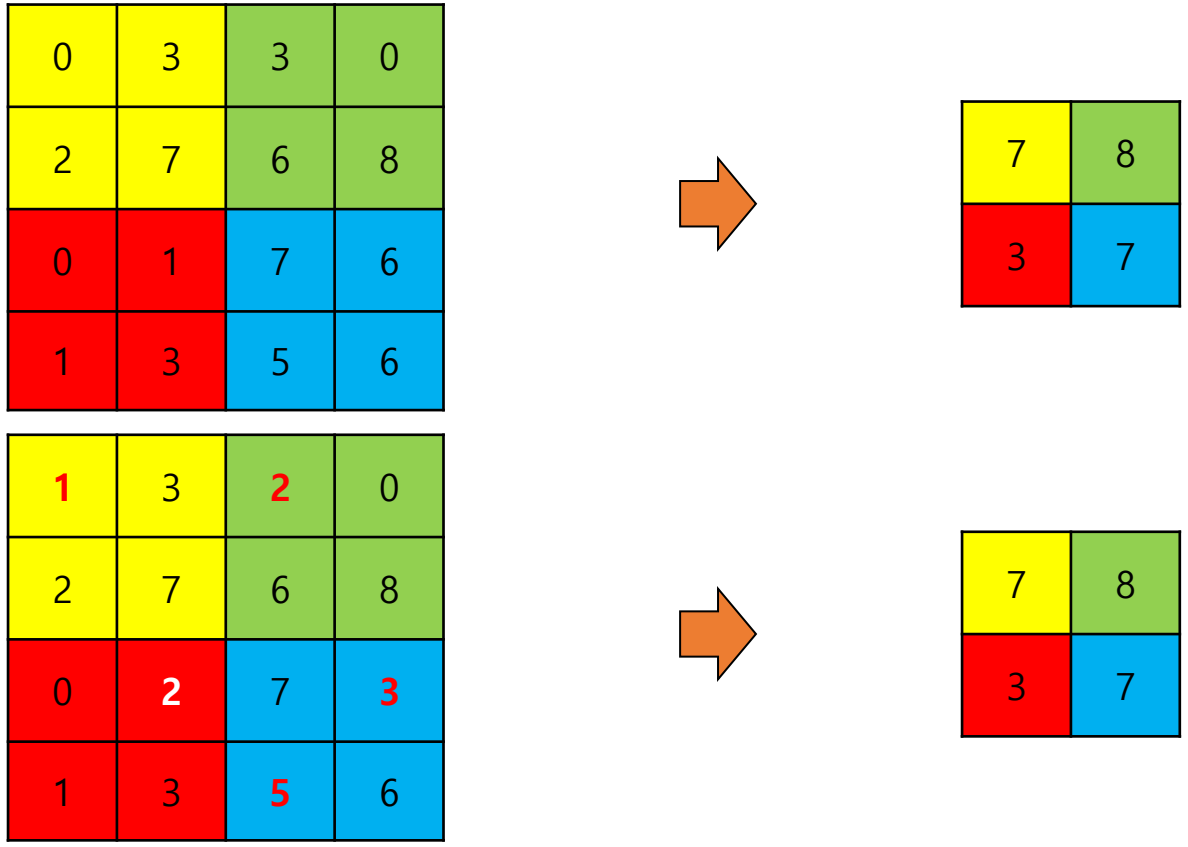
풀링의 특징 – 채널 수 고정

- 채널 수가 변하지 않는다
- 합성곱 연산에서는 하나의 필터가 모든 채널에 걸쳐 적용되므로 출력 채널은 필터의 수이다
  - 필터도 입력 데이터와 동일한 채널 수를 가져야 한다
  - 각 필터는 입력 이미지의 모든 채널에 대해 합성곱을 수행하고, 그 결과를 합산하여 하나의 출력 채널을 만든다
  - 따라서 최종 출력 결과의 채널 수는 필터의 수와 같다
- 반면, 풀링 연산은 채널마다 독립적으로 이루어지므로 채널 수가 고정된다
- 즉, 풀링 연산은 입력 데이터의 채널 수 그대로 출력 데이터로 내보낸다



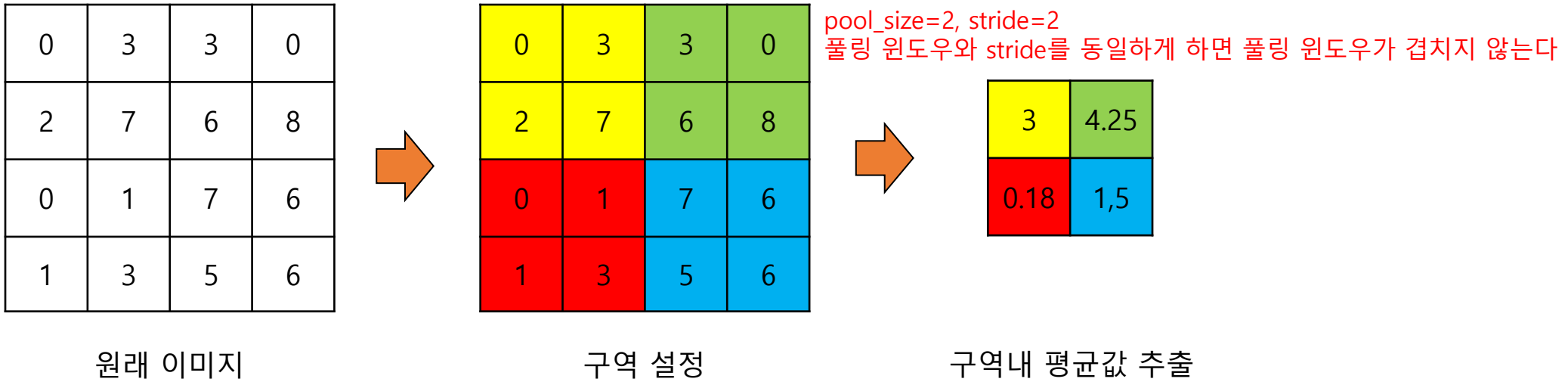
풀링의 특징 – 강건성

- 입력 데이터가 조금 변해도 풀링의 결과는 잘 변하지 않는다
- 특히 맥스 풀링의 경우에는 최댓값을 취하기 때문에 작은 노이즈의 영향을 받지 않는다



Average Pooling

- 평균 풀링은 주어진 영역의 평균을 취하는 기법
- 평균을 취하므로 영역 내의 모든 픽셀 정보를 반영하며,
- 극단적인 값들을 평활화하고, 더 부드러운 특성을 제공하는 특징이 있다
- 그 결과로 노이즈에 강하고, 안정적인 특성을 추출할 수 있다



Average Pooling 구현

- import numpy as np

- def pool2d_mean(img, pool_size=2, stride=2):
    height, width = img.shape
    out_height = int((height - pool_size) / stride + 1)
    out_width = int((width - pool_size) / stride + 1)
    img_ = np.zeros((out_height, out_width), dtype=np.float32)

    for y in range(out_height):
        for x in range(out_width):
            img_[y, x] = np.mean(img[y*stride:y*stride+pool_size, x*stride:x*stride+pool_size])

    return img_

일반적으로 취하는 값. 크기는 절반으로 줄어든다
원본 이미지의 높이와 너비
출력 이미지의 높이
출력 이미지의 너비
소수점 유지 위해 float로 설정
풀링 윈도우도 필터와 같은 개념으로 볼 수 있다
원본 이미지 사이즈 – 풀링 사이즈 + 1을 해야 한다
풀링 윈도우가 가능한 모든 위치에서 진행

y*stride : 시작 위치. stride 거리만큼 이동하여 다음 시작 위치를 결정
+pool_size : 풀링 윈도우의 폭만큼에서
np.mean : 평균값을 구함

시각화

- import cv2
- from google.colab.patches import cv2_imshow

- img = cv2.imread("mountain.jpg", cv2.IMREAD_GRAYSCALE)
- output = pool2d_mean(img, pool_size=2, stride=2)
- cv2_imshow(img)
- cv2_imshow(output)
- cv2.waitKey(0)
- cv2.destroyAllWindows()

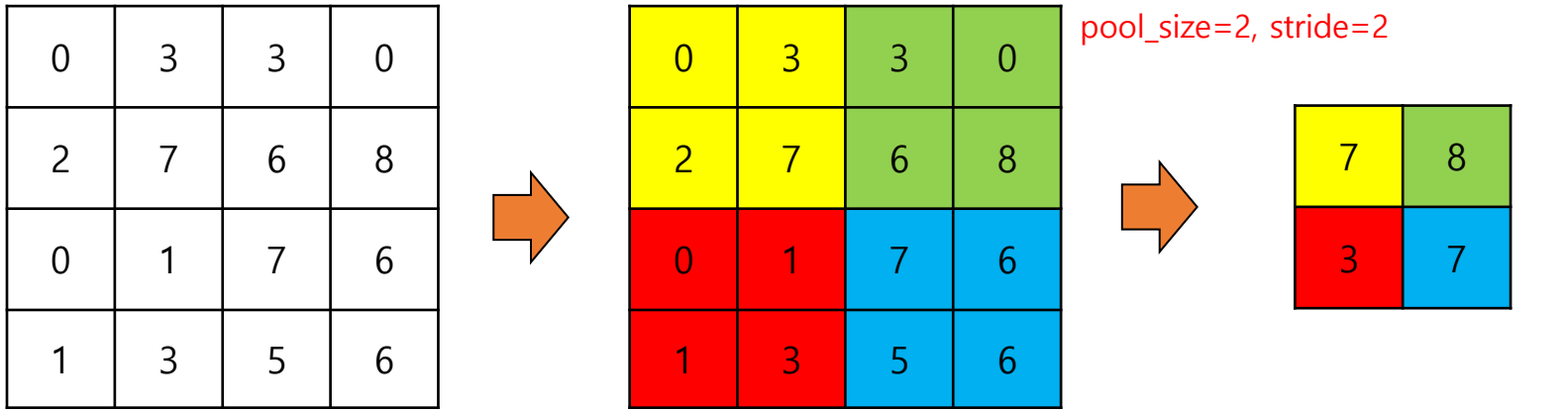
크기 확인

- print(img.shape)
- print(output.shape)

```
(700, 1024)
(350, 512)
```

Max Pooling

- 맥스 풀링은 이미지의 모든 값을 사용하지 않고 일정 구역 내에서 최대 값만을 사용한다
- 각 윈도우 내에서 가장 두드러진 특성이 강조되므로, 특히 경계 파악에 효과적이다
- 선택된 최대값은 윈도우 내에서 일정하게 유지될 가능성이 높으므로
- 입력 데이터의 작은 변화나 위치 이동에 덜 민감하다
- 또한 노이즈와 같은 불필요한 변동은 보통 낮은 값이므로 효과적으로 필터링한다



이미지에 존재하는 객체의 특징을 더 잘 포착하기 때문에 맥스 풀링이 더 선호된다

Max Pooling 구현

- import numpy as np

- def pool2d_max(img, pool_size=2, stride=2):
    height, width = img.shape
    out_height = int((height - pool_size) / stride + 1)
    out_width = int((width - pool_size) / stride + 1)
    img_ = np.zeros((out_height, out_width), dtype=np.uint8)

    for y in range(out_height):
        for x in range(out_width):
            img_[y, x] = np.max(img[y*stride:y*stride+pool_size, x*stride:x*stride+pool_size])

    return img_

일반적으로 취하는 값
원본 이미지의 높이와 너비
출력 이미지의 높이
출력 이미지의 너비
결과 이미지 초기화. 원본의 데이터 타입 유지

풀링 윈도우는 필터와 같은 개념이므로,
원본 이미지 사이즈 – 풀링 사이즈 + 1을 해야 한다

풀링 윈도우가 가능한 모든 위치에서 진행

y*stride : 시작 위치. stride 거리만큼 이동하여 다음 시작 위치를 결정
+pool_size : 풀링 윈도우의 폭만큼에서 가장 큰 값을 구함(np.max)

시각화

- import cv2
- from google.colab.patches import cv2_imshow

- img = cv2.imread("mountain.jpg", cv2.IMREAD_GRAYSCALE)
- output = pool2d_max(img, pool_size=2, stride=2)
- cv2_imshow(img)
- cv2_imshow(output)
- cv2.waitKey(0)
- cv2.destroyAllWindows()

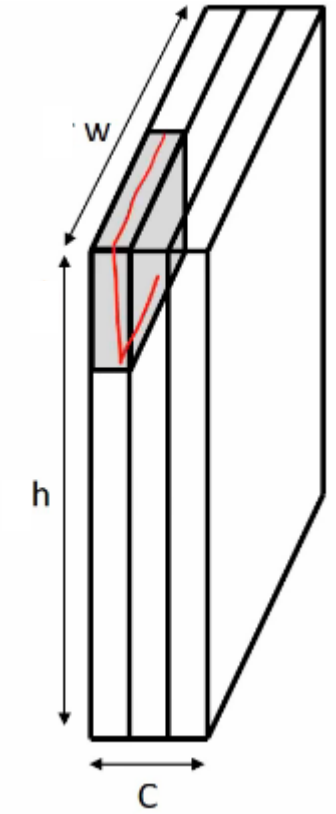
크기 확인

- print(img.shape)
- print(output.shape)

```
(700, 1024)
(350, 512)
```

멀티 채널 풀링

- 여기서는 2차원 이미지 데이터의 풀링만 알아보았지만,
- 색상별 채널이 있는 3차원 데이터에서도 동일한 방법으로 적용된다
- 다만 채널별로 풀링이 이루어질 뿐이다



Keras에서의 Pooling

- Keras에서는 다음과 같이 구현할 수 있다
- 풀링 윈도우가 겹치지 않도록 stride는 pool_size와 갖도록 기본 설정되어 있다
- pool_size는 stride로서 아래와 같이 2라고 하면 크기가 반으로 줄어든다

MaxPooling

- model.add(layers.MaxPooling2D(**pool_size=2**))

AveragePooing

- model.add(layers.AveragePooling2D(**pool_size=2**))

pool_size=2 를 선택하면 2 x 2 윈도우와 stride=2를 사용하게 된다 (즉 2 == (2, 2))

풀링 기법 비교

- 평균 풀링과 맥스 풀링 결과를 비교해본다