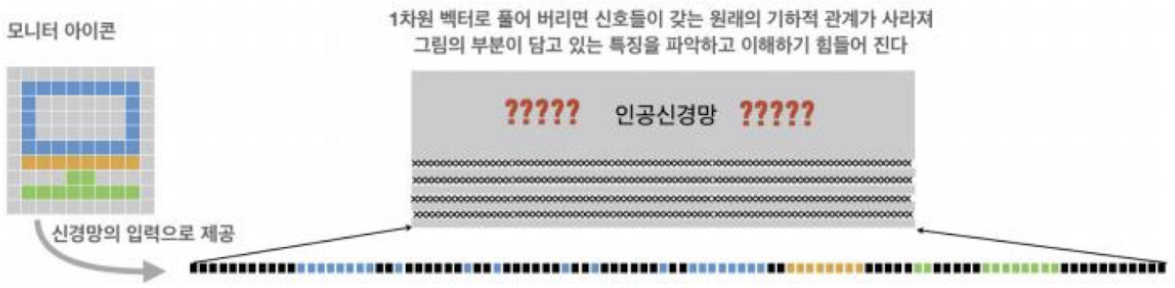
CNN(Convolutional Neural Network) 정리

CNN(Convolutional Neural Network)

- 딥러닝(Deep Learning)의 주요 모형
- CNN은 영상 및 이미지 인식에 특화된 다층 신경망
- 이미지, 혹은 자연의 영상의 경우 공간적으로 강한 지역적 상관관계를 갖고 있음
- 위와 같은 특성을 이용하여 신경망을 효율적으로 구현할 수 있고, 상당한 양의 가중치 매개변수를 감소시키는 특징을 가짐
- CNN의 주요 목적은 **테두리, 선, 색 등 이미지의 특성을 감지**하는 것
- 합성곱 계층과 풀링 계층을 쌓아 나가는 형태로 인공 신경망 구성

Convolution이라는 연산 수행을 통해 학습

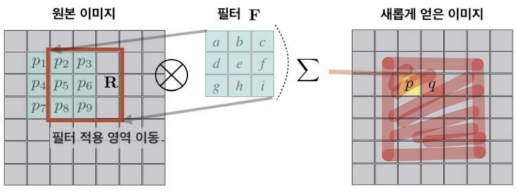
- 다른 영상 분류 알고리즘에 비해 상대적을 입력에 대한 전처리가 거의 필요하지 않음
- 신경망은 이미지를 잘 다루기 위해 특징을 추출하여 입력으로 만드는 단계가 필요하지만, CNN은 학습 과정에 입력을 특징으로 추출하는 방법도 함께 학습하므로 문제와 관련된 지식을 바탕으로 **특징을 추출해 내는 과정이 필요하지 않음**



- 기존 신경망은 이미지의 픽셀을 1차원 벡터로 펼쳐서 제공함. 이런 형태에서는 특징의 찾기가 쉽지 않다.
- 이미지 내의 특정한 영역에 대해 원래의 이미지가 가지는 기하적 관계를 유지한 채로 살펴보는 것이 필요 → 합성곱(convolution) - 특정한 영역을 하나의 특징으로 변환하는 역할

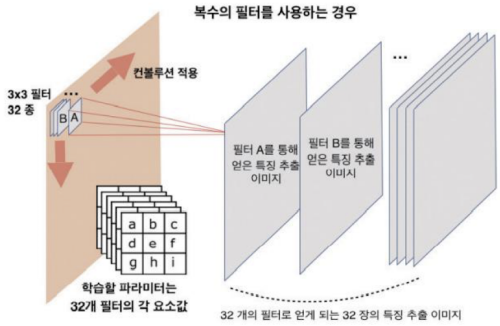
필터(filte)=커널(kenel)

: 3X3 노란색 박스 - 특정 영역의 가중치 영역(행렬) - 특정 영역과 필터의 합성곱을 통해 하나의 특징을 얻음



- 평균 필터 : 하나의 픽셀 값을 변경할 때 주위의 값을 고려하여 평균을 취하는 방법
- 가우스 필터 : 모든 픽셀에 동일한 중요성을 부여하는 것이 아니라 중심 픽셀에는 더 높은 중요도를 부여하고, 중심에서 멀어질수록 중요성을 낮게함.

  —> 이 중요도를 가중치(weight)라고 부름.



이렇게 필터를 통해 출력된 값을 특성맵(feature map)이라 한다.

feature map : 합성곱(필터를 이용한 연산) 계산을 통해 얻은 출력

합성곱을 적용하면 이미지가 작아짐

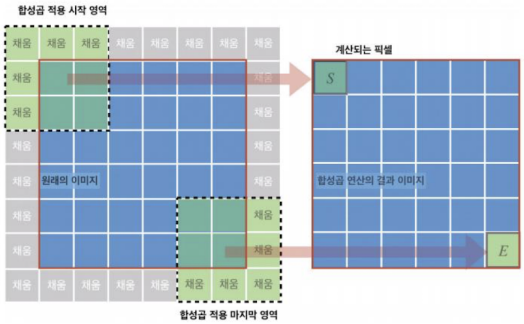
! —>

- 합성곱을 통해 이미지의 크기가 작아지는 것이 이미지 전체의 특성을 요약한 것이 아님
- **이미지의 가장자리 정보들이 소실되고 중심부의 정보만 남기** 때문에 정보의 추상화 과정X
- 정보를 요약하지 않으면 입력 단계의 작은 잡음이 가진 영향력도 계층을 거치며 사라지지 않고 계속해서 전파됨
- 합성곱에 의해 이미지가 작아지는 것이 아니고, 정보를 요약하는 계층을 통해 이미지를 적절한 크기로 줄이는 과정이 필요(pooling)

!! **padding(패딩)** : 입력 이미지의 주변에 값을 가상의 원소로 덧대어 채워주는 일

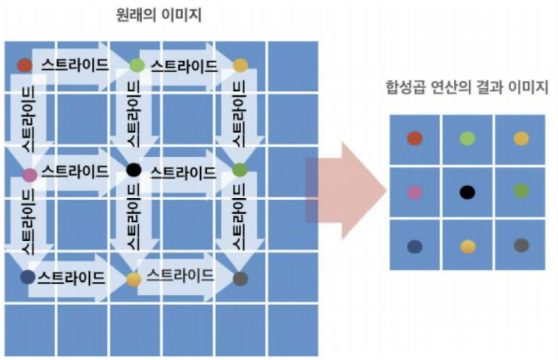
→ 원래의 이미지와 같은 크기의 이미지

- same padding(세임 패딩) : 0으로 패딩하는 것
- valid padding(밸리드 패딩) : 패딩을 사용하지 않은 합성곱 연산



모서리에 있는 중요한 정보가 특성맵으로 잘 전달되지 않고 가운데 있는 정보만 두드러지게 표현된다.

→ 중앙부와 모서리 픽셀이 합성곱에 참여하는 비율이 동일해짐.



윈도우(window)

: 필터가 적용되는 영역

스트라이드(stride)

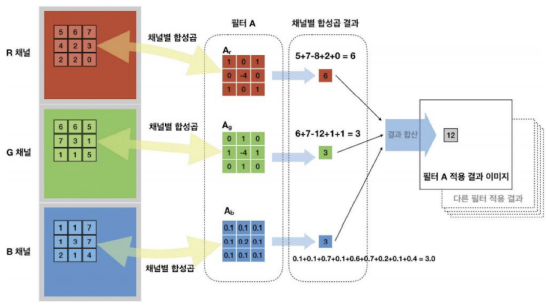
: 필터가 움직이는 보폭

컬러 이미지를 사용한 합성곱

컬러 이미지는 RGB(빨강, 초록, 파랑) 채널로 이루어진 다중 채널이다. 이는 3차원 배열이다.

→ 합성곱 연산을 위해서는 필터 또한 3차원 배열이다. 그러나 출력 값은 1차원 배열로 이루어진 값 1개이다.

→ 배열이 늘어난 만큼 연산량도 늘어난다. 그래서 대부분 이미지 학습 과정에서 grayscaled을 진행한다.



Pooling

: 이미지의 일정한 영역 내의 픽셀들이 가진 값을 하나로 축소하는 연산 즉, 합성곱 층에서 만든 특성 맵의 가로세로 크기를 줄이는 역할 수행

- 정보를 요약하는 역할
- 합성곱과 달리 윈도우를 중첩시켜 이동시키지 않는 것이 일반적
- Max Pooling : 풀링 적용 영역 내에서 가장 큰 값을 결과로 선택하는 것
- Average pooling : 평균을 구해 결과로 삼는 것
- 다수의 픽셀 정보를 통합하여 하나의 픽셀을 생성하기 때문에 정보를 요약할 수 있으며 **이 과정에서 입력의 변화에 대해 덜 민감한 신호 전달을 달성**

Pooling을 사용하면 이미지의 크기가 줄어들지만 합성곱 연산을 패딩 없이 적용했을 때 이미지가 줄어드는 것과는 다름

- 합성곱 연산에 의한 축소 : 이미지의 주변부 정보를 잃는 일
- 풀링에 의한 축소 : 다수의 픽셀 정보를 통합하여 하나로 만드는 것으로 원래 신호에 존재하는 noise(잡음)요소를 제거

Pooling은 최대값 추출이나 평균값 추출과 같이 미리 정해진 기능을 수행하므로 학습 단계에서 파라미터를 최적화할 필요가 없는 계층

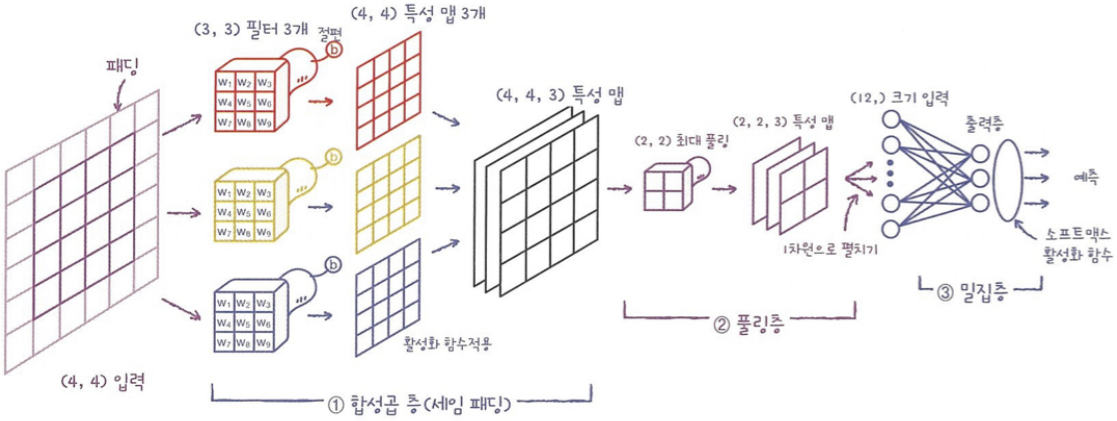
- 원래 이미지가 가지고 있던 채널을 그대로 유지하면서 공간만 줄이는 일을 수행
- Pooling에는 가중치가 없기 때문

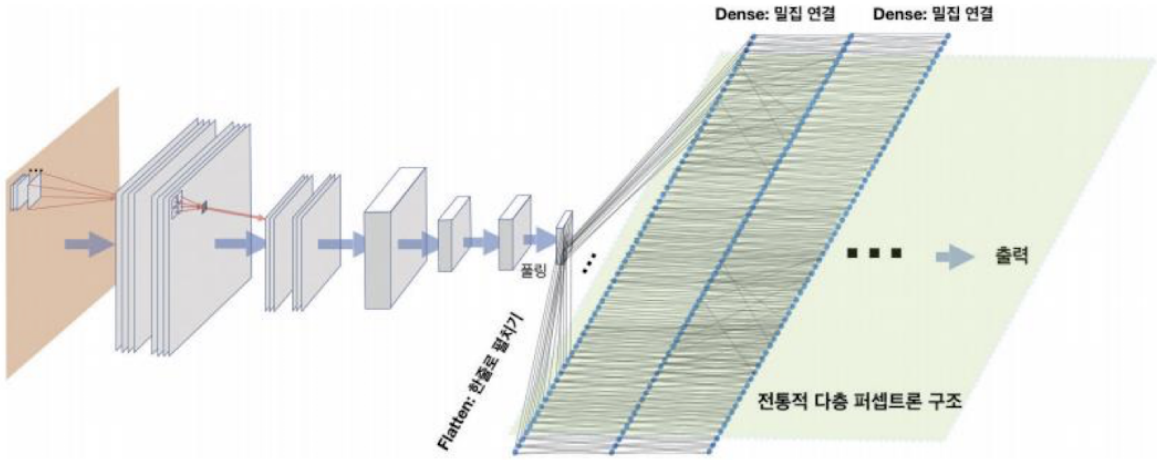
픽셀의 값을 일부 변경하여도 최대값 풀링의 결과는 잘 바뀌지 않음

→ 이동 불변성(translation invariance), 이동 불변성을 가진 모델 : 강건한(robust)모델이라함.

→ 합성곱 계층, 풀링 계층을 통해, 모델 파라미터 개수를 효율적으로 줄여주어 전체 모델 복잡도가 감소하는 효과

합성곱 신경망 모델의 구성





- 합성곱 신경망의 신호 연결 : 패딩을 고려한 합성곱과 풀링이 순차적으로 이루어지게 하는 것
- 합성곱 필터의 가중치 : 파라미터로 사용되는 것
- 몇 차례의 합성곱을 한 뒤에 풀링을 할 수도 있고, 풀링 없이 합성곱으로만 연결 가능
- 풀링만으로 연결될 수는 없음. → 풀링은 정보를 줄이기만 할 뿐 학습 가능한 파라미터가 없기 때문.
- 이미지를 처리하기 위해서는 위 과정을 통해 얻은 최종 이미지를 다층 퍼셉트론(MLP) 형태의 전통적인 신경망에 연결
- 위 과정을 위해서는 합성곱 신경망의 앞 부분에서 얻은 이미지 정보들을 1차원 벡터로 만드는 **flatten(평탄화)** 과정 필요

```
model = keras.models.Sequential( [
    keras.layers.Conv2D(input_shape = (64, 64, 3),
                        kernel_size = (3,3), filters = 32),
    keras.layers.MaxPooling2D((2, 2), strides=2),
    keras.layers.Conv2D(kernel_size = (3,3), padding='same' filters = 64),
    ... ])
```

1. Sequential 클래스의 객체를 만든다.
2. 첫번째 합성곱 층인 Conv2D를 추가함. cf. add() 메서드를 사용해 층을 하나씩 차례대로 추가.

ex) keras.layers.Conv2D(32, kenel_size=3, activation='relu', padding='same', input_shape=(28,28,1))

3. pooling 층 추가 → 여기까지 첫번째 합성곱-풀링 층 생성
4. 두번째 합성곱-풀링 층 생성
5. Flatten
6. Dense 은닉층 생성
7. 드롭아웃 → 과대적합 막음
8. Dense 출력층 생성 - activation = softmax

→ summary() 메서드로 모델 구조 출력

—> 이를 ex) model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics='accuracy')

ModelCheckpoint 콜백과 EarlyStopping 콜백을 함께 사용해 조기 종료 기법을 구현함.
