CNN

작성일시	@2025년 2월 19일 오후 10:42
유형	MS AI 1차 프로젝트
자료	cnn_test_annotated.ipynb
복습	☐
ID	TSK-28

CNN(Convolutional Neural Nets)

https://wikidocs.net/227547

1. 개요:

CNN(Convolutional Neural Nets)은 이미지 및 영상 인식을 위한 딥러닝의 기본 모델이다

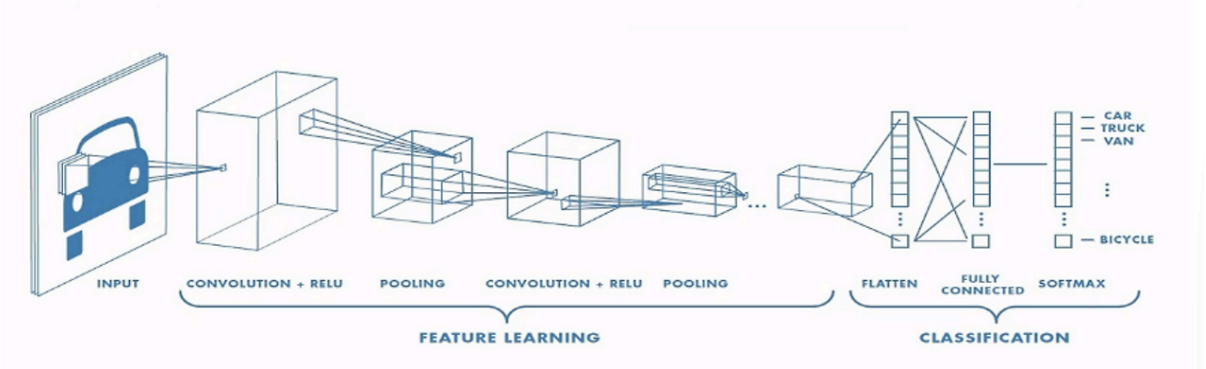
MLP(Multi layer perceptron)의 개선형

- **MLP의 신경망 구조**의 경우 **완전연결층(Fully-Connected Layer)을 구성**하기 때문에 **은닉층을 추가할수록** 학습해야 할 파라미터(weights)의 수가 많아지고, **학습 복잡도가 높아**진다
- **Convolution 연산을 활용하는 CNN**의 경우 MLP와 달리 인접 층의 노드들이 **Fully Connected를 이루지 않는다.**

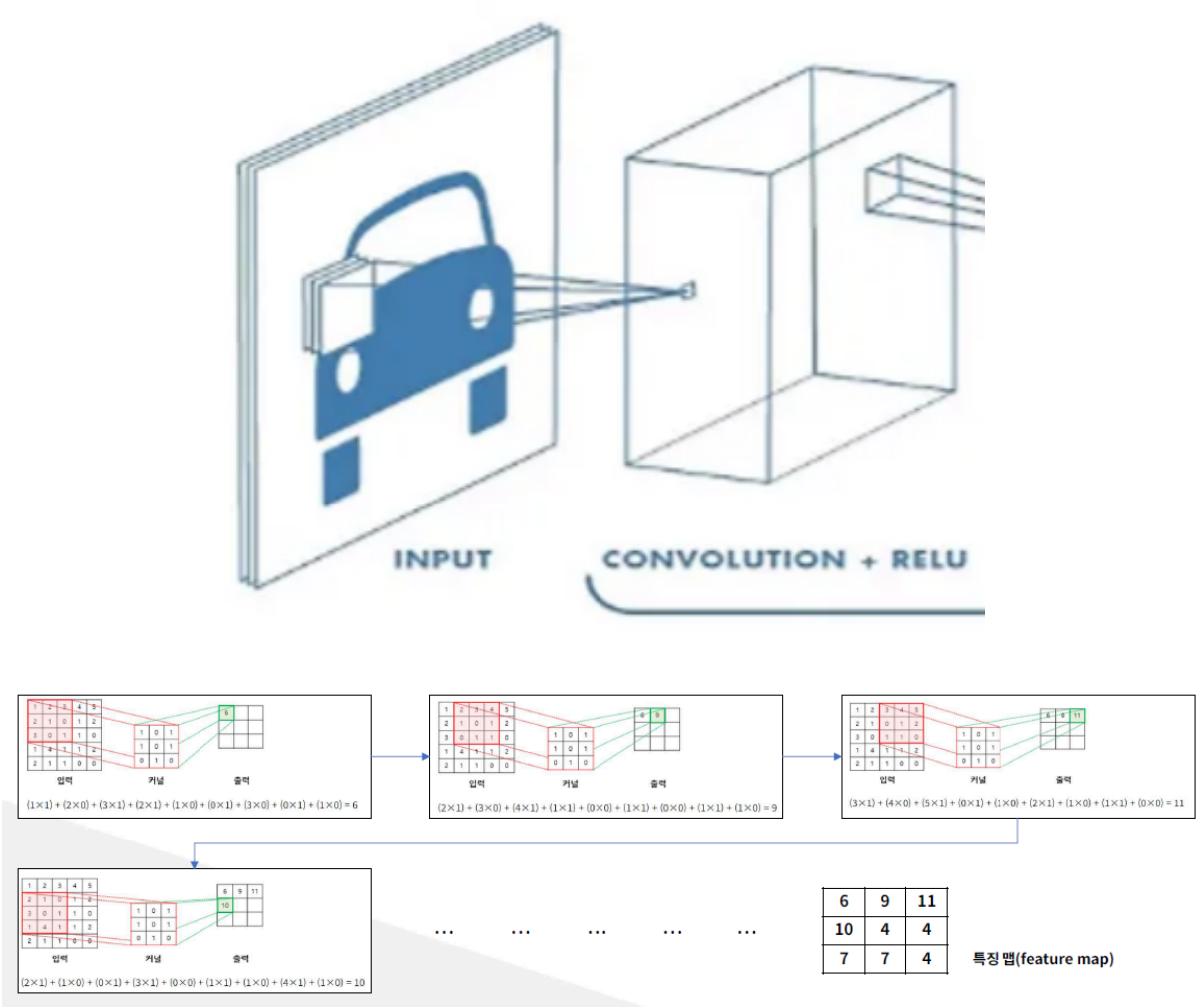
→ 따라서 **신경망의 복잡도가 낮아지고, 효율적인 학습이 가능**하게 된다

2. CNN layer의 구조

- **convolutional layer** : convolution 연산을 수행하여 신경망의 복잡도를 낮추는 층
- **pulling layer** : pulling 연산을 수행하여 이미지의 특징을 추출하는 층
- **Fully Connected Layer** : MLP와 같은 구조의 최종 출력층



3. convolution 연산



Convolutional Layer는 **Convolution**처리와 **Activation function**으로 구성되어 있다

입력 데이터는 **kernel(혹은 filter)**라고 불리는 행렬과 **convolution 연산(*)**을 수행한다.

Convolution 연산을 통해 **이미지 차원의 축소**되고(Padding을 하지 않을 경우) 결과값에 **Activation function**(예를 들면 **ReLU function**)을 적용하여 **특징 맵(featrue map)**이 만들어지게 된다.

3.1 Activation Function(활성화 함수)

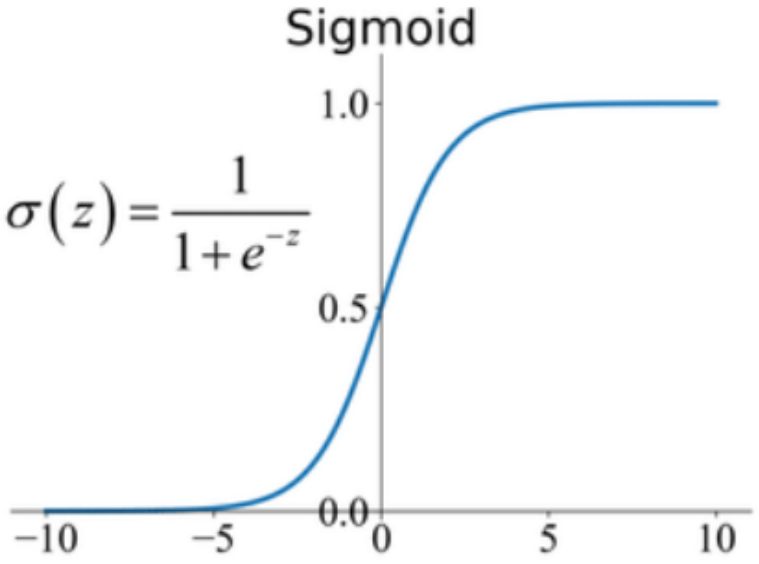
https://happy-obok.tistory.com/55https://heeya-stupidbutstudying.tistory.com/entry/ML-활성화-함수Activation-Function

선형 함수의 문제는 층을 아무리 깊게해도 은닉층이 없는 네트워크로도 똑같은 기능을 할 수 있기에 신경망에서 선형 함수를 이용하면 신경망의 층을 깊게하는 의미가 없어진다

→

이처럼 **선형인 활성화 함수**를 이용하면 여러층으로 구성하는 이점을 살릴 수 없기 때문에 층을 쌓기 위해서는 **비선형 함수인 활성화 함수를 사용**해야 하고, **활성화 함수**로는 **비선형 함수를 사용**해야 합니다.

3.1.1 시그모이드(sigmoid) 함수



시그모이드(sigmoid)란 'S자 모양'이라는 뜻입니다. 식에서 e 마이너스 x제곱에서 e는 자연 상수로 2.7182... 의 값을 갖는 실수입니다. 실수 값을 입력 받아 **0~1 사이의 값**으로 압축합니다.

단점 1) 기울기 소멸 문제 (Vanishing Gradient Problem)가 발생

역전파 중에 이전의 기울기와 현재 기울기를 곱하면서 점점 기울기가 사라지게 된다.

그렇게 되면 신경망의 학습 능력이 제한되는 포화(Saturation)가 발생

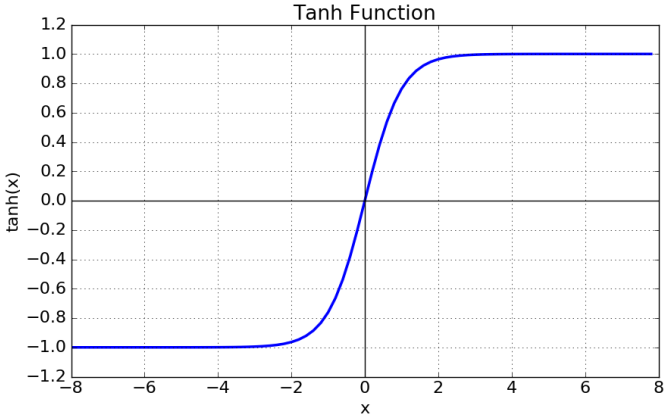
단점 2) 시그모이드 함숫값은 0이 중심 (zero-centered)이 아닙니다.

sigmoid 함수는 항상 양수이기에 **sigmoid**를 한번 거친 이후론 input 값은 **항상 양수**가 된다. 그렇게 되면 **역전파(backpropagation)**을 할 때 문제가 생긴다.

2차원 평면에서 살펴보면 부호가 모두 같은 지점은 1,3 사분면 뿐이다. 따라서 지그재그의 형태로 학습이 될 수 밖에 없고 이는 학습을 오래 걸리게 한다.

단점 3) exp 연산 때문에 자원과 시간이 많이 소모됩니다.

3.1.2. Tanh (Hyperbolic Tangent function)

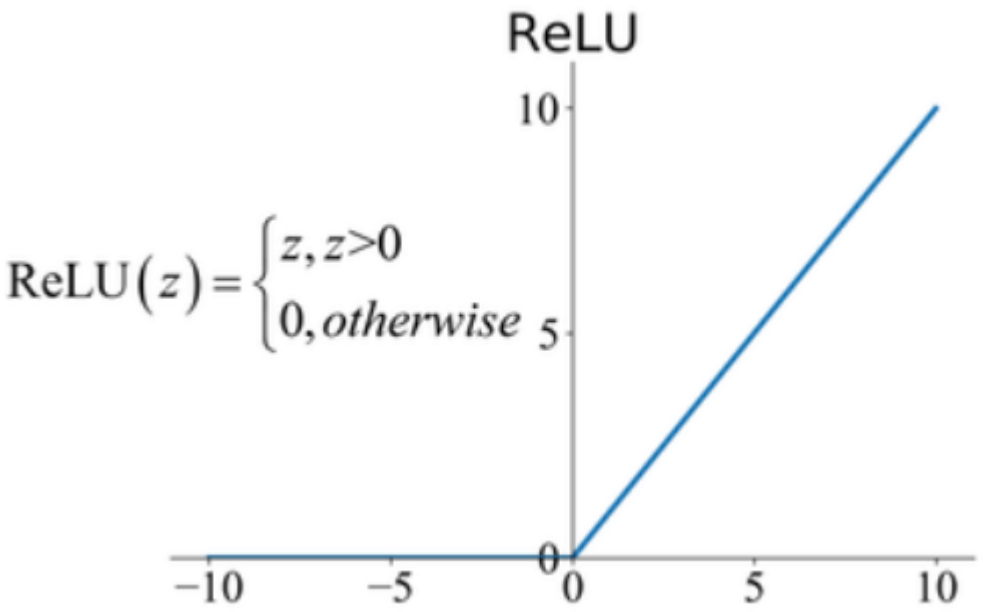


시그모이드와 비슷하지만 실수 값을 입력받아 **-1~1 사이의 값**으로 압축합니다.

시그모이드를 두 배 해주고 -1 한 값과 같습니다.

tanh는 결괏값이 -1~1 사이로 zero-centered 하여 지그재그가 덜하여 시그모이드에 비해 최적화를 잘합니다. 하지만 **시그모이드와 같이 기울기 소실 문제**를 가지고 있습니다.

3.1.3. ReLU(Rectified Linear Unit) 함수



가장 많이 사용하는 활성화 함수입니다.

입력이 0을 넘으면 그 입력을 그대로 출력하고, 0 이하이면 입력을 출력하는 함수입니다.

양수 부분에서는 **포화(saturate)가 발생하지 않습니다.exp 연산이 없어 빠릅니다**.

하지만 0 이 중심(zero-centered)이 아니어서 위의 **지그재그 문제**가 있습니다.

3. 2. convolution 연산에서 조정 가능한 파라미터

필터 크기(Kernel Size) : convolution 연산에서 사용되는 필터의 크기를 정한다.

(필터 크기가 커지면 특징추출이 증가하지만 연산량이 늘어난다.)

스트라이드(stride) : 필터가 입력 데이터에서 convolution 연산을 수행하는 간격.

(스트라이드가 커질수록 convolution연산을 수행횟수가 줄어들지만 이미지의 정보 손실이 발생하게 된다.

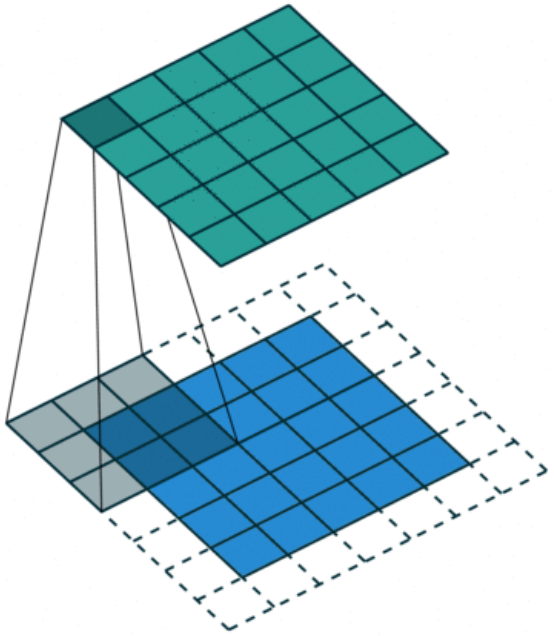
패딩(Padding) : 출력 크기 조절을 위해 사용하는 기법.

convolution연산을 수행하면서 작아진 **특징 맵(feature map)**의 크기를 입력 맵과 동일하게 유지하기 위해 크기를 늘리고 주변을 '0'과 같은 무의미한 데이터로 채워 넣는다.

패딩을 수행할 경우 이미지의 중요한 특징이 가장자리에서 손실 되지 않고 유지될 수 있다.

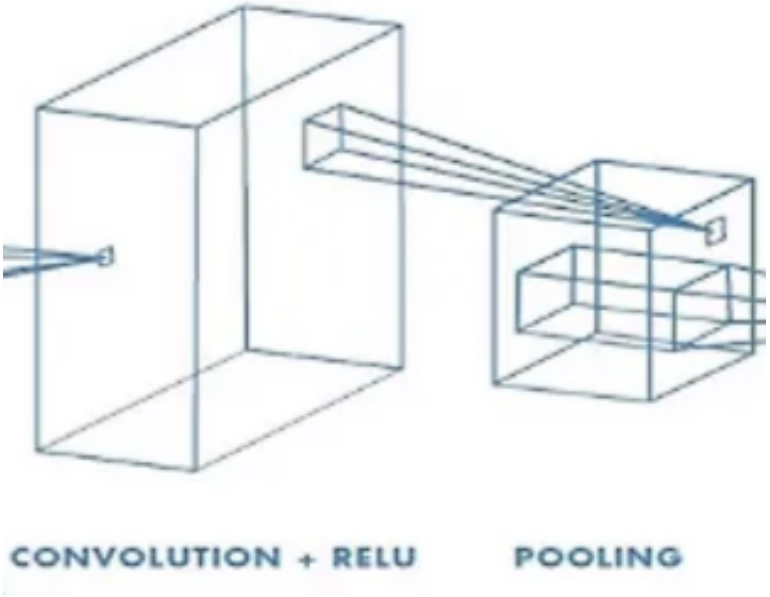
스트라이드와 함께 사용하면 모델의 크기를 유지하면서 **연산량을 줄일** 수 있다.

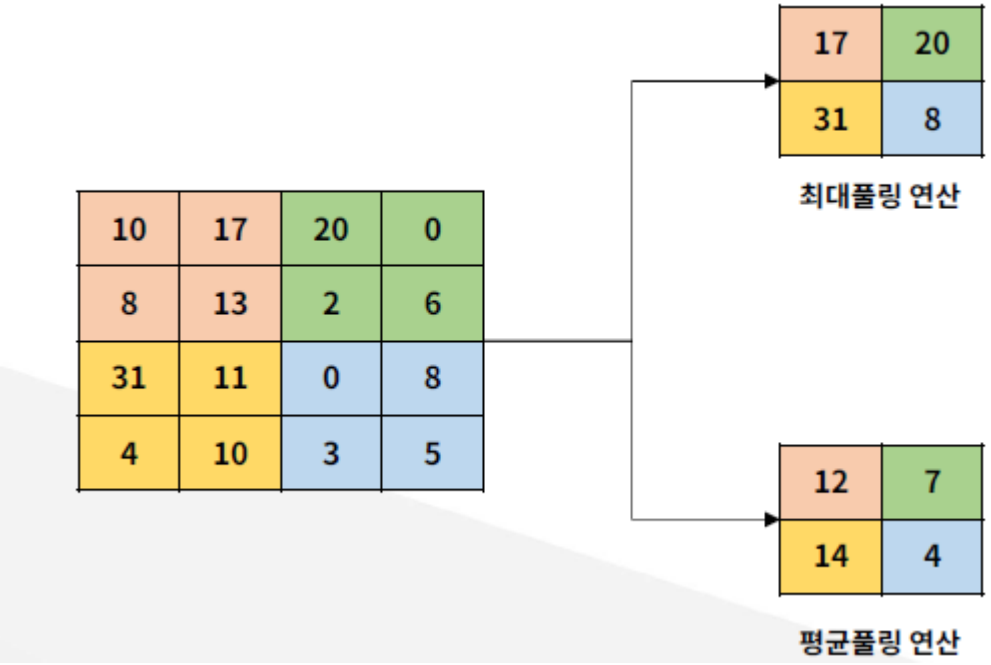
패딩 처리한 데이터는 **학습 시 가중치 업데이트가 더 안정적**으로 진행된다고 알려져 있다.



https://images.app.goo.gl/2sHrEZEr1TKsRNte8

4. Pooling Layer





- **pulling** 연산은 input 이미지를 정해진 경계 만큼 격자 형태로 분할한 뒤, 해당 분할된 영역 안에서 가장 큰 값 **Max Pooling (최대 풀링)** 하나만을 선택하거나, 분할 영역 안의 요소들의 평균값**Average Pooling (평균 풀링)**을 취한다. (일반적으로는 **Max Pooling (최대 풀링)**을 많이 사용한다.)

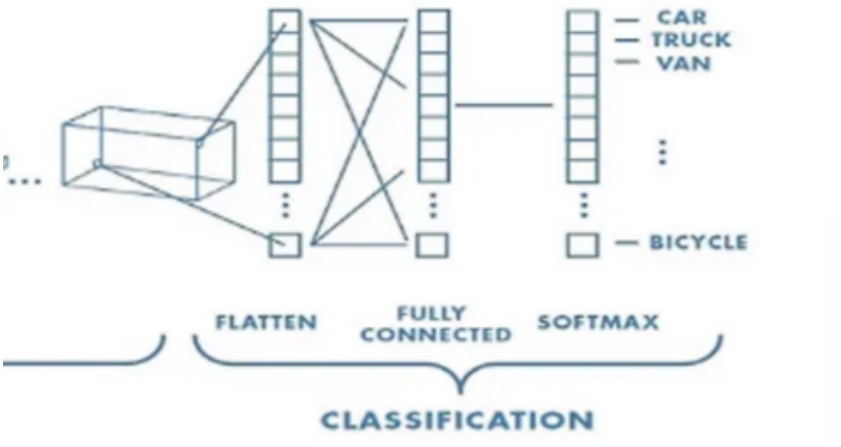
📌**Max Pooling**→ 강한 특징(엣지, 텍스처) 강조 →**객체 인식에 유리**

📌**Average Pooling**→ 전체적인 정보 유지 →**이미지 평탄화, 블러링에 유리**

4.1 Pooling 연산의 역할

- **특징 다운샘플링(Feature Downsampling)** : 모델 **데이터의 크기를 감소**시키면서도 모델의 **중요한 특징을 유지**한다
- **변이에 대한 강건함(Intensivity) 유지** : 풀링 연산이 진행되는 **지역 안에서 가장 두드러지는 특징만 추출**함으로서 입력 이미지의 작은 **변화나 왜곡**에 대해서도 **모델이 안정성을 유지**하도록 만든다
- **과적합(Overfitting) 방지** : 특징 개수를 감소 시키면서 모델의 파라미터를 줄이고, 과적합의 위험을 감소시킨다
- 추상화 수준 증가 : 이미지의 추상화 수준을 높여 더 **고차원의 특징을 추출**할 수 있게 한다

5. Fully Connected Layer(Perceptron)



Convolution과 Pulling 연산을 반복하여 차원이 축소된 이미지는
flatten 연산을 통해 1차원 벡터로 변환되고, 이 **벡터값이 Fully Connected layer를 통과**하면서 출력을 발생시킨다.

6. 다중 채널 이미지(RGB, RGBA, CMYK...)

설명한 CNN 구조는 입력 데이터가 **한 층의 격자 그리드로 표현**될 수 있는 이미지에 적용된다. (예: 흑백 이미지)

색상이나 명암, 투명도 등을 가진 더 복잡한 형태의 이미지일 경우 **입력 레이어가 여러 장**으로 들어오게 된다.

예컨대 A라는 RGB 이미지의 경우, R(적), G(녹), B(청)으로 구분될 수 있는 **세 개의 layer로 구분될 수 있다**.

32×32의 이미지일 경우 전체 input은 32×32×3이 된다.

(※ MLP에서는 이러한 이미지가 flatten 연산 후 concat(그대로 이어 붙임)되어 32×32×3의 1차원 벡터로 신경망에 입력된다.)

채널별 특징 학습 : 다중 채널일 경우 각 색상 채널에서 특징이 학습된다. 각 채널 간의 관계와 채널 간의 패턴을 모두 학습해야 한다.

데이터 전처리 시 모든 입력 이미지가 동일한 크기의 채널 구성(동일한 이미지 형식)으로 전처리 되어야 한다.

[참고]

다중 이미지 채널의 형태들

- RGB : 적, 녹, 청의 3개의 채널로 구성됨
- RGBA : RGB에 알파 채널(투명도)이 추가된 4개의 채널로 구성됨
- CMYK : 인쇄 산업에서 주로 사용되는 색상 모델로 Cyan(시안), Magenta(마젠타), Yellow(노랑), Key(검정)의 4개의 채널
- HSV : 색상(Hue), 채도(Saturation), 명도(Value)의 3개의 채널을 사용함
- 적외선, 열화상 : RGB 이외의 채널을 사용함
- 멀티스택트럼 이미지 : 4개 이상의 채널을 가짐