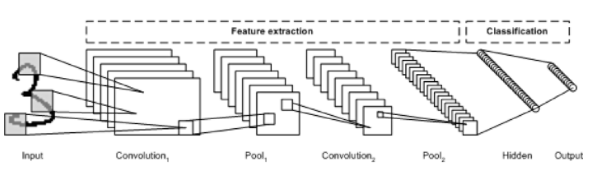
CNN 기본 원리[[1]](#footnote-1)

이미지의 공간 정보를 유지하면서 인접 픽셀과의 특징을 효과적으로 인식하고 강조하는 방식으로 이미지의 특징을 추출한다.

1차원 배열형태로 input을 압축시키지 않고, 이미지의 공간 정보를 유지한 상태로 학습이 가능한 모델, 합성곱(Convolution)을 이용하여 이미지 처리에 필요한 가중치의 수를 줄임으로써 연산량의 감소와 효과적인 이미지 처리를 목표로 함.

크게 2개의 단계로 나누어져 있는데, 아래 그림과 같이 Feature extraction와 Classification(Fully Connected Layer)로 나뉘어진다. Feature extraction단계에서는 Convolution Layer와 Pooling Layer가 반복되면서 이미지의 특징을 찾아내고, Classification단계에서는 발견한 특징점을 기반으로 이미지를 분류하는데 사용됨.



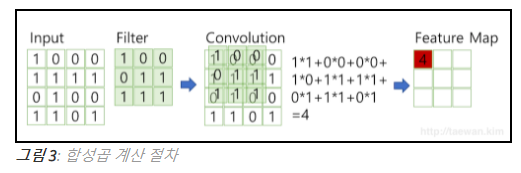
* 복수의 필터로 이미지의 특징 추출 및 학습
* 추출한 이미지의 특징을 모으고 강화하는 Pooling 레이어(선택적)

Convolution 채널에 유입되는 입력 데이터에는 한 개 이상의 필터가 적용됨. Convolution Layer에 n개의 필터가 적용된다면 출력 데이터는 n개의 채널을 갖게 됨.

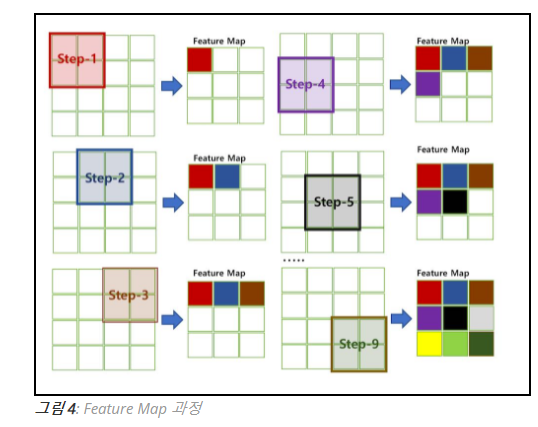
예시> 컬러 이미지는 각 픽셀마다 RGB값을 가지고 있기 때문에 2(가로 x 세로)+1(각 픽셀 당 RGB)=3차원 데이터이다. Red, Green, Blue Filter가 적용된다면 이 이미지는 3개의 채널을 가지는 것과 같다.

# 필터 & Stride

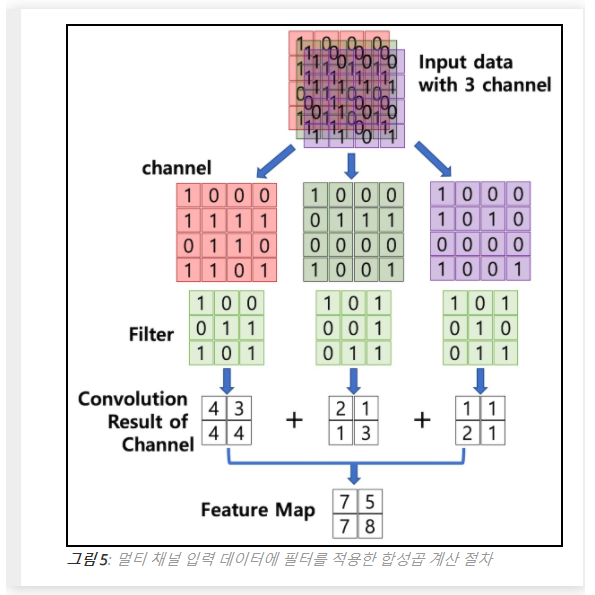
필터(=Kernel) : 이미지의 특징을 찾아내기 위한 공용 파라미터, 일반적으로 4x4, 3x3과 같은 정사각 행렬로 정의됨.



CNN에서 학습을 하는 것은 필터에 각 위치에 들어가는 숫자(!) = 필터 파라미터



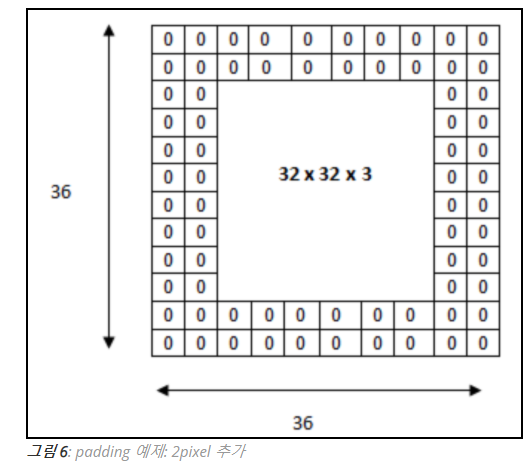
위 그림과 같이 Filter가 일정한 칸 수 씩 옆으로 움직이면서 합성곱 연산(Component-wise multiplication)을 수행하게 되는데, 이 때 ‘일정한 칸 수’가 Stride값이 된다.



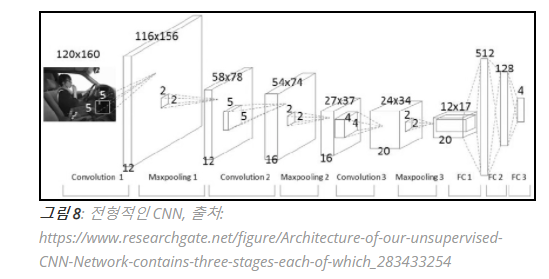
위의 그림에서는 하나의 필터를 적용했지만, 하나의 Convolution Layer에 크기가 같은 여러 개의 필터를 적용할 수 있다. 적용한 필터의 개수만큼 Feature Map의 채널이 늘어나게 된다. Feature Map에 활성 함수를 적용하면 Activation Map이 되고, 이것이 Convolution Layer의 최종 출력 결과이다.

Filter의 작용으로 보통 Feature Map의 크기는 입력데이터보다 작게 된다. 이때, Feature Map이 작아지는 것을 방지하기 위해 사용하는 방법이 Padding으로, 주어진 input의 외각에 지정된 픽셀 만큼 특정 값으로 채워 넣는 것을 의미한다. 일반적으로 0값을 집어넣는다고 한다.

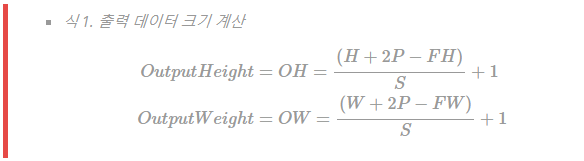
이렇게 하면 Feature Map의 사이즈를 조절할 수 있을 뿐만 아니라, 사진의 외각을 인식하는 효과도 얻을 수 있다고 한다. 실제로 그럴 거 같은 느낌이 있긴 있다.

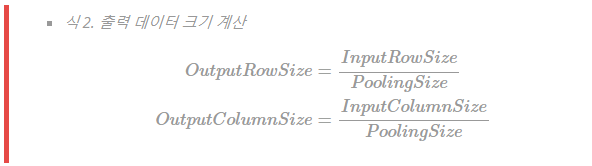


Pooling Layer는 Activation Map의 크기를 줄이거나 특정 데이터를 강조하는 용도로 사용된다. CNN에서는 주로 Max Pooling을 사용하며, 모든 원소가 한 번씩 처리되도록 Pooling크기와 Stride값을 같게 해주는 것이 일반적이다.



Convolution Layer와 Max Pooling Layer를 반복적으로 쌓은 특징 추출 부분과 Fully Connected Layer를 구성하고 마지막 출력층에 Softmax를 적용한 분류 부분으로 나뉜다.





CNN을 구성하면서 Filter, Stride, Padding, Pooling을 조절하여 특징 추출 부분의 입력과 출력 크기를 계산하고 맞추는 작업이 중요하다.

What is object detection?

1. 사진 속에 어떤 Object가 있는지 찾아낸다.
2. 그 Object가 어디에 위치해 있는지 찾아낸다.

Fast R-CNN, Retina-Net, Single-Shot MultiBox Detector(SSD)와 같은 프로그램이 있다. YOLO는 single algorithm run에 물체들을 인식할 수 있기 때문에 앞의 방법들보다 유리하다고 한다.

What is YOLO?[[2]](#footnote-2)

YOLO : You Only Look Once / real-time object detection system

실시간 처리에 알맞은 높은 속도와 정확성이 장점이라고 함.

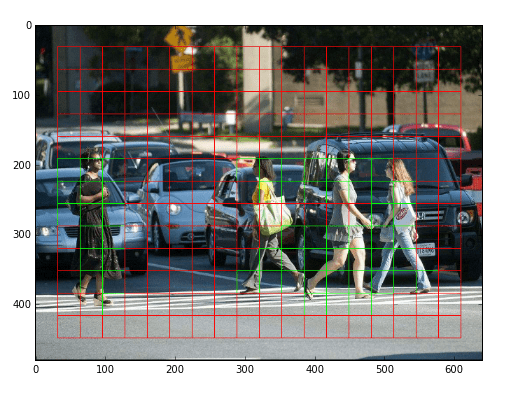
Solve regression problem & Provides the class probabilities of the detected images

YOLO algorithm은 CNN을 사용한다. 물체를 인식하기 위해 뉴럴 네트워크에 대해 단 한번의 forward propagation을 필요로 한다. CNN은 다양한 물체들의 후보와 각각에 대한 확률(class probabilities)을 계산하고 그 물체의 위치를 표시해주는 Bounding Box를 예측하는데 쓰인다.

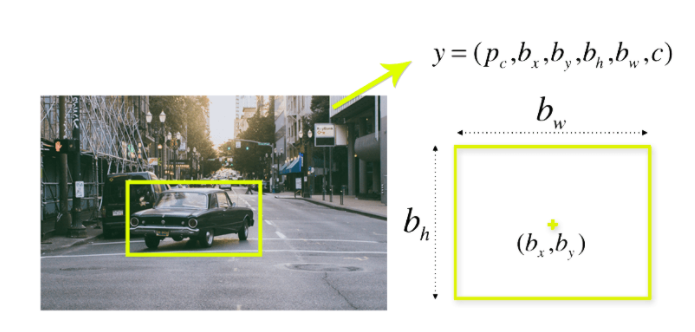
YOLO algorithm의 기본적인 원리

1. Residual blocks

Input image는 동일한 크기의 정사각형 모양의 여러 개의 grid로 나누어 진다. 그리고 각 grid에서 object를 탐지한다고 한다.

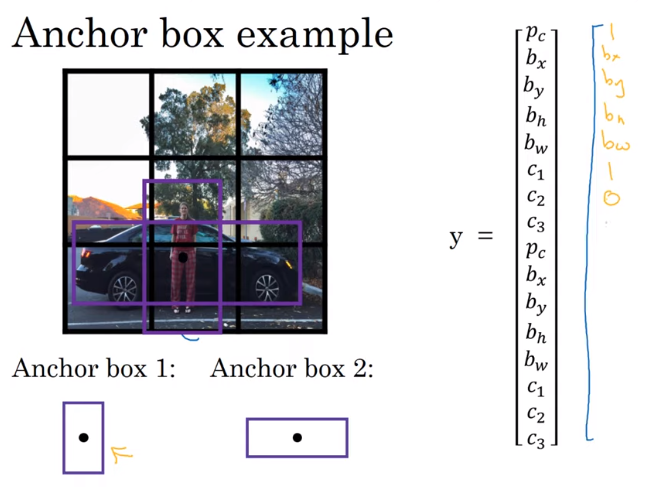


1. Bounding box regression

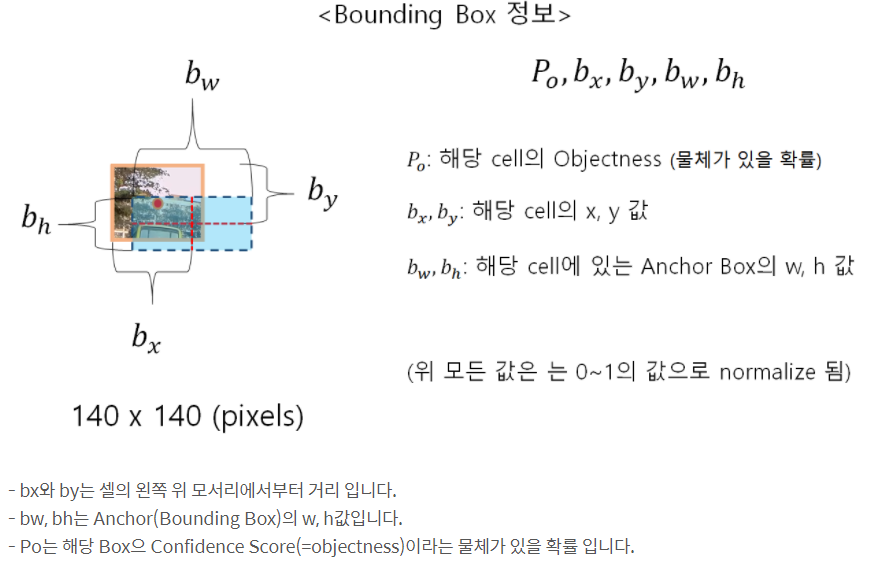


각각의 Bounding box는 탐지한 물체를 둘러싸고, 예측한 물체의 종류(class)와 그 물체로 예측한 확률 값을 속성으로 가지게 된다.

이때, 각각의 grid cell에서 물체가 여기 있을 것이다! 라는 Bounding box를 제시하는데, 이를 Anchor box라고 한다. 예를 들어 각각의 셀에 2개의 Anchor box가 있다고 하면,



위 그림과 같이 각 셀에 저장된 정보(y)는 길이 16인 배열로 표현할 수 있을 것이다. 각 parameter가 의미하는 바는 아래 그림에 설명되어 있다.



여기서 눈치챌 수 있는 것은 Anchor Box의 중심이 각 Grid cell의 범위를 벗어날 수 없다는 점이다. 따라서 탐지하고자 하는 물체의 중심이 있는 Grid cell의 Anchor Box가 결국 그 물체에 대한 유효한 Bounding Box가 될 거라는 것을 눈치챌 수 있다.[[3]](#footnote-3)

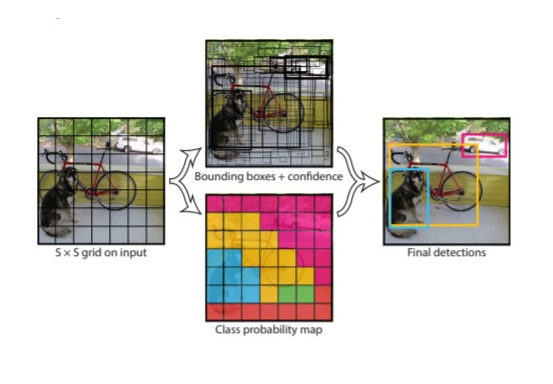
이때, 위위그림에서 c1, c2, c3는 각 bounding box가 각 클래스를 포함할 확률을 나타내는 것이다. 예를 들어 개 95%, 고양이 3%, 호랑이 2% 이런 식.

사실 Grid를 잘게 쪼갠다면 하나의 cell에 2개 이상의 물체의 중심점이 겹치는 일은 거의 없다. 따라서 Anchor box가 가지는 의의는 특정한 모양에 대해(예를 들면, 넓적한 모양이라던지 또는 길쭉한 모양이라던지) 잘 찾아낼 수 있도록 유도를 해줄 수 있다는 것이다. [[4]](#footnote-4)

1. Intersection over Union

IoU = 교집합(Intersection) / 합집합(Union set)

IoU 값이 1에 가까워 지도록 알고리즘이 작동한다.



딥러닝 강좌

<https://www.youtube.com/watch?v=PIft4URoQcw&list=PLSAJwo7mw8jn8iaXwT4MqLbZnS-LJwnBd&index=19>

Introduction to Yolo version 3

<https://www.youtube.com/watch?v=vRqSO6RsptU>

single Neural Network 사용

53개의 convolutional layer들을 포함, 이를 Darknet-53라고 부른다.

추가적으로 자체적으로 53개의 layer들을 포함하고 있으며 따라서 총 106개의 layer가 있다.

Detections at layers: 82, 94, 106

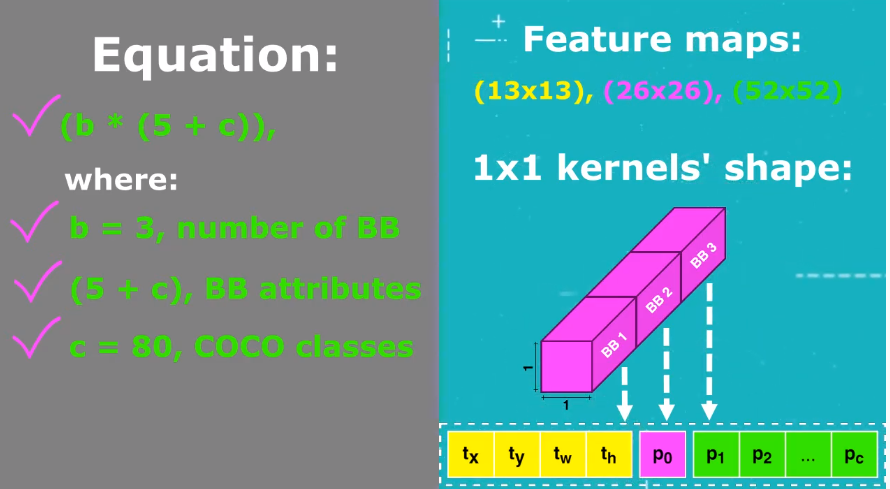
No pooling layers -> convolutional layers instead(filter 안의 숫자들 합하기)

* Prevents loss of low level features, ability to detect small objects

Input network size의 너비와 높이는 32로 나누어 떨어져야 함. Input image size자체는 아무거나 가능. 네트워크에 들어갈 때 저절로 resize됨

82, 94, 106 layer에서 32, 16, 8의 stride를 적용하여 input을 downsizing한다. 각각 큰 물체, 중간 크기 물체, 작은 물체를 탐지하는 역할을 한다.

Bounding box의 개수는 3개



P0는 objectness score

Training을 할 때 정답으로 주어지는 bounding box가 있다. Feature map에서 object가 존재하는 것으로 해당되는 영역 중에서 중간에 있는 것을 이 물체를 예측한 Cell이라고 하고, Cell의 objectness를 1로 둔다. 그러면 3개의 Bounding box 중에서 어떤 것일까?

1. <http://taewan.kim/post/cnn/#1-1-%ED%95%A9%EC%84%B1%EA%B3%B1-convolution> [↑](#footnote-ref-1)
2. https://www.section.io/engineering-education/introduction-to-yolo-algorithm-for-object-detection/ [↑](#footnote-ref-2)
3. https://mickael-k.tistory.com/27 [↑](#footnote-ref-3)
4. Anchor Box를 고르는 건 직접해도 되지만 자동적으로 Anchor Box를 골라주는 심화적인 방법들이 존재한다. https://www.youtube.com/watch?v=RTlwl2bv0Tg [↑](#footnote-ref-4)