객체인식

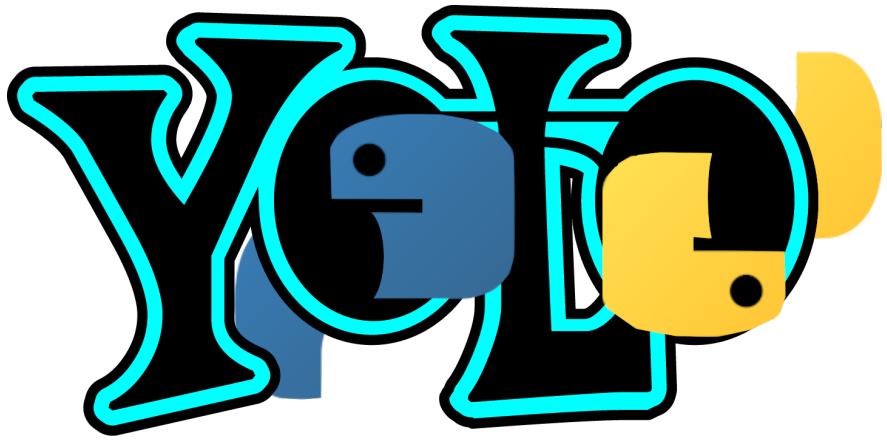
(주)인피닉스 – 강호용 연구원

Infinyx

One-Stage YOLO 알고리즘 소개

Infinyx
THINK LIFE SYNC AI

YOLO (You Only Look Once) 알고리즘의 특징 소개



특징

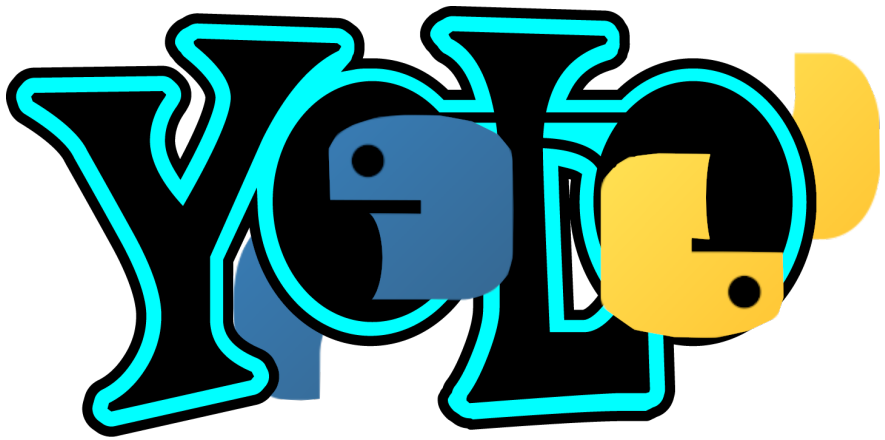
1.실시간 객체 탐지: YOLO 알고리즘은 이미지를 한 번의 전방향 패스로 처리하여 객체 탐지를 수행하기 때문에 빠른 속도로 실시간 객체 탐지가 가능합니다.

2.단일 네트워크: YOLO 알고리즘은 하나의 신경망 모델로 모든 작업을 처리합니다. 따라서 다른 알고리즘에 비해 구조가 간단하고 사용이 간편합니다.

3.반전 손실 함수: YOLO 알고리즘은 객체와 바운딩 박스에 대한 손실 함수를 사용하여 전체 이미지를 고려하여 객체 탐지를 수행합니다. 이로 인해 객체의 정확한 위치와 크기를 예측하는데 도움이 됩니다.

4.다양한 크기의 객체 탐지: YOLO 알고리즘은 여러 개의 Anchor Box를 사용하여 다양한 크기와 종횡비의 객체를 탐지할 수 있습니다.

YOLO (You Only Look Once) 알고리즘의 장점 소개



장점

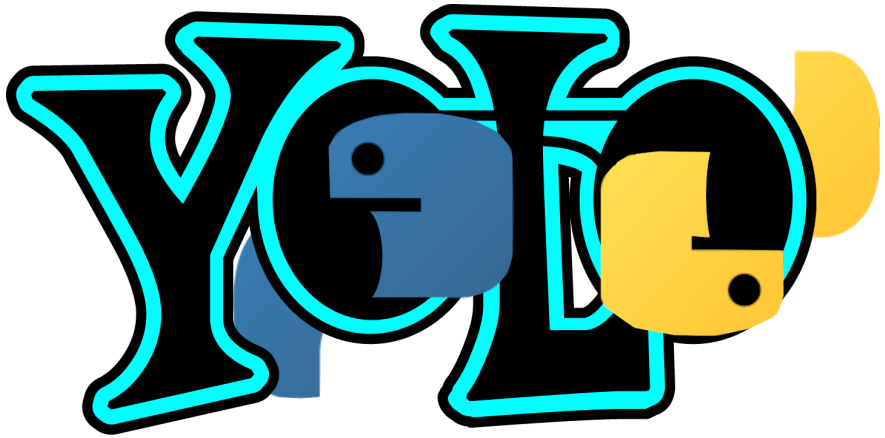
1.빠른 속도: YOLO 알고리즘은 단일 전방향 패스로 객체 탐지를 수행하기 때문에 높은 처리 속도를 갖습니다. 실시간 객체 탐지에 적합합니다.

2.단순한 구조: YOLO 알고리즘은 단일 네트워크로 구성되어 구조가 간단하며, 쉽게 사용할 수 있습니다.

3.높은 정확도: 반전 손실 함수를 사용하여 정확한 객체의 위치와 크기를 예측할 수 있어 높은 정확도를 보입니다.

4.다양한 크기의 객체 탐지: Anchor Box를 사용하여 다양한 크기와 종횡비의 객체를 탐지할 수 있어 다양한 상황에 적용 가능합니다.

YOLO (You Only Look Once) 알고리즘의 단점 소개



단점

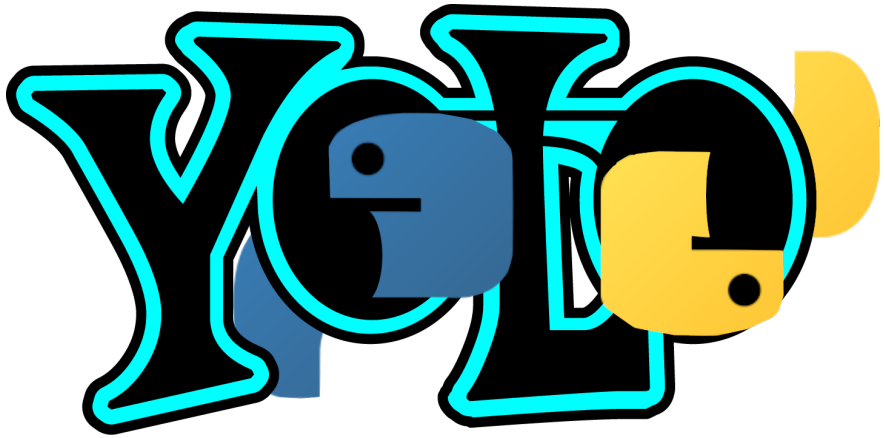
1.작은 객체 탐지 어려움: YOLO 알고리즘은 작은 객체보다는 큰 객체에 대해 더 좋은 성능을 보이는 경향이 있습니다. 작은 객체의 탐지는 어려울 수 있습니다.

2.객체 겹침 처리 어려움: 객체가 서로 겹치는 경우에 대한 처리가 어려울 수 있으며, 객체 분리가 어려울 수 있습니다.

3.손실 함수 불균형: 객체가 희소하게 분포되어 있는 경우 손실 함수의 불균형 문제가 발생할 수 있습니다.

4.큰 객체와 작은 객체 처리 불균형: YOLO 알고리즘은 큰 객체와 작은 객체를 동시에 처리하는데 어려움이 있을 수 있습니다.

YOLO (You Only Look Once) 알고리즘 정리



- 요약하면, YOLO 알고리즘은 빠른 속도와 높은 정확도를 갖는 실시간 객체 탐지 알고리즘이지만 작은 객체 탐지에 어려움이 있고, 객체 겹침 처리에도 한계가 있을 수 있습니다.

YOLO 알고리즘의 작동 원리 간단히 이해

입력 이미지를 그리드로 분할: 입력 이미지를 사전에 정해진 그리드로 분할합니다. 그리드의 각 셀은 하나의 예측 영역을 담당하게 됩니다.

각 그리드 셀에서 바운딩 박스 예측: 각 그리드 셀마다 여러 개의 Anchor Box를 미리 정의합니다. 이 Anchor Box들은 다양한 크기와 종횡비를 가지며, 객체 탐지를 위해 사용됩니다. 각 그리드 셀마다 Anchor Box들로부터 바운딩 박스를 예측합니다.

객체 확률 예측: 각 그리드 셀은 해당 셀에 속하는 객체의 존재 여부를 예측합니다. 즉, 해당 셀에 객체가 있을 확률을 예측합니다.

YOLO 알고리즘의 작동 원리 간단히 이해

클래스 예측: 객체가 존재하는 그리드 셀에서 해당 객체의 클래스를 예측합니다.
이를 위해 일반적으로 softmax 함수를 사용하여 다중 클래스 분류를 수행합니다.

바운딩 박스와 클래스 예측 통합: 모든 그리드 셀의 바운딩 박스와 클래스 예측 결과를 통합합니다.
이로 인해 하나의 예측 결과가 생성됩니다.

바운딩 박스 필터링: 예측된 바운딩 박스들 중에서 일정한 임계값(threshold) 이상의 신뢰도를 가지는 바운딩 박스들만 선택하여 최종 객체 탐지 결과를 얻습니다.

요약하면, YOLO 알고리즘은 이미지를 그리드로 분할하고 각 그리드 셀에서 바운딩 박스, 객체 확률, 클래스 등을 예측하여 객체 탐지를 수행하는 알고리즘입니다. 이러한 예측 결과를 통합하고 필터링하여 최종 객체 탐지 결과를 얻게 됩니다.

YOLO 모델의 구조와 레이어 설명

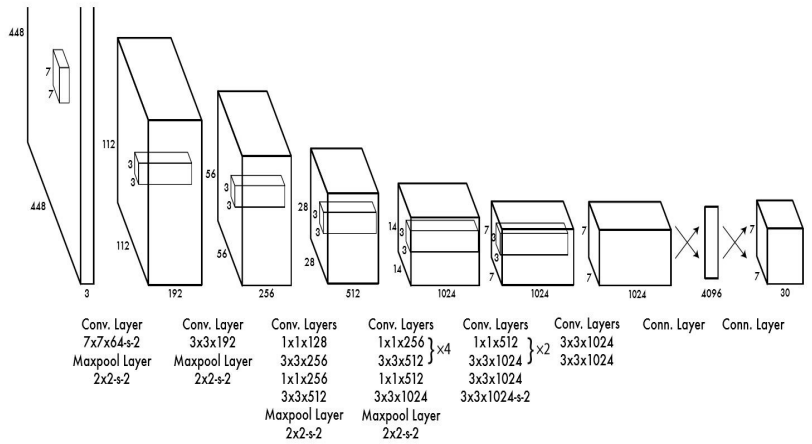


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

YOLO (You Only Look Once) 모델은 여러 개의 Convolutional 레이어와 Fully Connected 레이어로 구성된 딥러닝 아키텍처입니다. 기본적으로는 Darknet라는 네트워크 구조를 사용합니다.

1. Input Layer:

YOLO 모델은 이미지를 입력으로 받습니다. 입력 이미지의 크기는 고정되어 있어야 합니다.

2. Convolutional 레이어 (Convolutional Layers):

YOLO 모델은 여러 개의 Convolutional 레이어를 통해 이미지의 특징을 추출합니다. Convolutional 레이어는 이미지를 작은 필터로 합성곱하여 특징 맵을 생성합니다. 이를 통해 이미지의 공간적 정보를 반영하는 특징들을 추출할 수 있습니다.

YOLO 모델의 구조와 레이어 설명

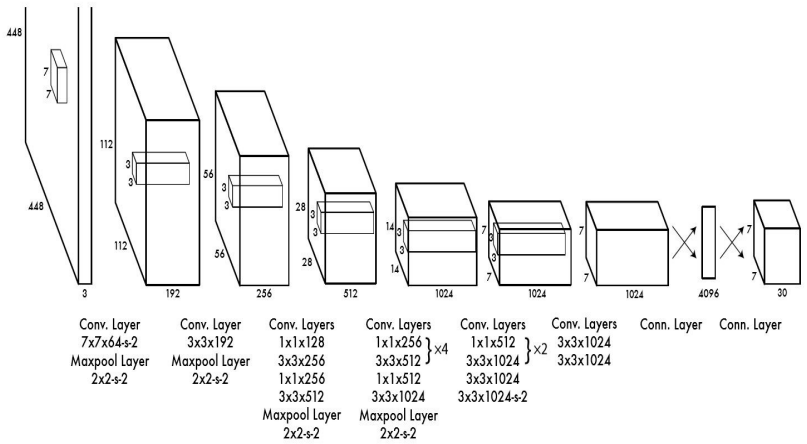


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

Downsampling 레이어 (Downsampling Layers)

YOLO 모델은 일정한 간격으로 Downsampling 레이어를 삽입하여 특징 맵의 크기를 줄입니다. 이는 객체의 위치와 크기에 상대적인 정보를 추출하기 위한 것입니다. 일반적으로 MaxPooling이나 Strided Convolution 방식을 사용합니다.

Residual 레이어 (Residual Layers):

YOLO 모델은 Residual 레이어를 사용하여 네트워크의 깊이를 확장하면서도 그라디언트 소실 문제를 완화합니다. Residual 레이어는 스킵 연결을 통해 층을 건너뛰는 방식으로 동작합니다.

YOLO 모델의 구조와 레이어 설명

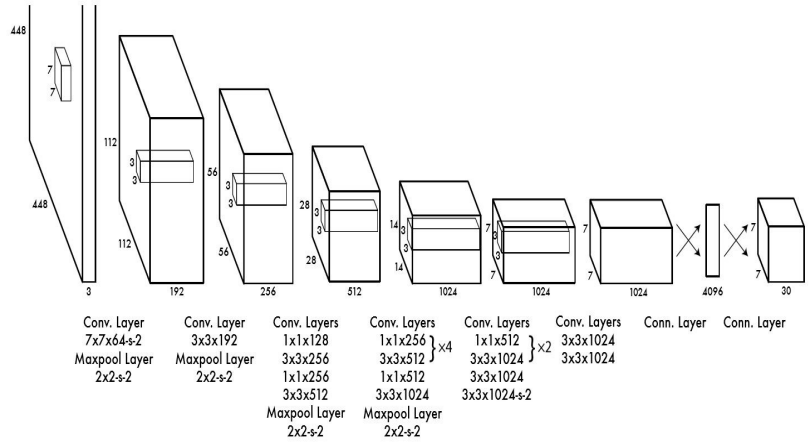


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

Detection 레이어 (Detection Layer)

YOLO 모델의 Detection 레이어는 객체 탐지를 수행합니다. 여러 개의 Anchor Box를 사용하여 바운딩 박스를 예측하고, 객체 확률과 클래스 확률을 예측합니다.

Output Layer

YOLO 모델의 출력 레이어는 객체 탐지 결과를 생성합니다. 바운딩 박스, 객체 확률, 클래스 확률 등의 정보를 포함하고 있습니다.

YOLO 모델은 기본적으로 이러한 레이어들을 조합하여 객체 탐지를 수행합니다. Darknet 구조를 기반으로 하지만, YOLO v3 등의 다양한 변형 모델이 존재하며, 각 모델은 구조나 레이어 설정에서 조금씩 차이가 있을 수 있습니다.

YOLO 모델 학습을 위한 데이터셋 준비 방법

- YOLOv5, YOLOv8 에서 사용하는 라벨 양식

<class_id> <center_x> <center_y> <width> <height>

class_id: 객체의 클래스를 나타내는 정수값. 데이터셋에 존재하는 클래스의 개수에 따라 0부터 N-1까지의 클래스 ID를 사용합니다.

center_x: 바운딩 박스의 중심점의 x 좌표를 이미지 너비에 대한 상대적인 값으로 나타냅니다. 0에서 1 사이의 값을 가집니다.

center_y: 바운딩 박스의 중심점의 y 좌표를 이미지 높이에 대한 상대적인 값으로 나타냅니다. 0에서 1 사이의 값을 가집니다.

YOLO 모델 학습을 위한 데이터셋 준비 방법

- YOLOv5, YOLOv8에서 사용하는 라벨 양식

<class_id> <center_x> <center_y> <width> <height>

width: 바운딩 박스의 너비를 이미지 너비에 대한 상대적인 값으로 나타냅니다. 0에서 1 사이의 값을 가집니다.

height: 바운딩 박스의 높이를 이미지 높이에 대한 상대적인 값으로 나타냅니다. 0에서 1 사이의 값을 가집니다.

예를 들어, 클래스 ID가 1이고 바운딩 박스의 중심점이 (50, 100)이며 너비가 80, 높이가 120인 경우 라벨은 다음과 같이 표현됩니다: 1 0.25 0.5 0.4 0.6

이러한 라벨 형식을 사용하여 YOLOv5 모델을 학습시킬 수 있습니다. 라벨 파일은 각 이미지 파일과 동일한 이름을 가지며 ".txt" 확장자를 가집니다.

SSD 알고리즘 소개

Infinyx
THINK LIFE SYNC AI

참고 : 정확도와 속도의 trade-off 이해

- 정확도와 속도는 객체 탐지 모델을 평가하고 사용할 때 중요한 trade-off(상충 관계) 관계를 갖습니다.

- 정확도: 정확도는 모델이 얼마나 정확하게 객체를 탐지하는지를 나타내는 지표입니다. 객체 탐지 모델의 정확도는 탐지된 바운딩 박스와 실제 바운딩 박스 사이의 IoU(Intersection over Union) 값을 기준으로 측정됩니다.

- 속도: 속도는 모델이 객체 탐지를 얼마나 빠르게 수행하는지를 나타냅니다. 실시간 응용이나 대량의 이미지 처리와 같이 빠른 속도가 필요한 경우가 있습니다. 객체 탐지는 하나의 이미지에 대해 수천 개의 바운딩 박스를 예측해야 하기 때문에 높은 계산 비용을 요구합니다.

정확도와 속도는 상충 관계이기 때문에, 높은 정확도를 달성하려면 더 많은 계산을 수행해야 하므로 속도가 느려질 수 있습니다.

반대로 빠른 속도를 달성하려면 모델의 복잡성을 줄이거나 계산을 최적화하면서 정확도를 포기해야 할 수 있습니다.

따라서 실제로 객체 탐지 모델을 적용할 때에는 애플리케이션의 요구사항과 제한 사항을 고려하여 정확도와 속도 사이에서 적절한 trade-off를 선택해야 합니다.

SSD 알고리즘의 작동 원리 소개

- 객체 탐지를 위한 딥러닝 기반 알고리즘으로, 하나의 단일 네트워크를 사용하여 이미지 속에서 다양한 객체를 탐지하는 데 특화되어 있습니다.

다양한 스케일의 특징 맵 생성: SSD는 입력 이미지를 다양한 스케일로 변환하여 여러 개의 특징 맵을 생성합니다. 이를 통해 작은 객체부터 큰 객체까지 다양한 크기의 객체를 탐지할 수 있습니다. SSD에서는 VGG 등의 기존 컨볼루션 네트워크나 MobileNet과 같은 경량화 네트워크를 사용하여 특징 추출을 수행합니다.

다양한 스케일의 앵커 박스 생성: 각 특징 맵에서는 여러 개의 앵커 박스를 생성합니다. 앵커 박스는 다양한 크기와 종횡비를 가지며, 각 위치에서 객체의 위치를 예측하는 데 사용됩니다. SSD는 특징 맵의 각 픽셀 위치에 대해 여러 개의 앵커 박스를 적용하여 다양한 크기와 종횡비의 객체를 탐지할 수 있습니다.

객체 예측과 바운딩 박스 조정: 각 앵커 박스는 객체의 위치와 클래스를 예측하기 위해 컨볼루션 레이어를 통과합니다. SSD에서는 객체의 위치와 크기를 바운딩 박스로 표현하며, 객체의 클래스를 예측하는데 Softmax 함수를 사용합니다. 이를 통해 객체 탐지를 클래스 분류와 바운딩 박스 조정 문제로 변환합니다.

 # SSD 알고리즘의 작동 원리 소개

바운딩 박스 필터링과 NMS(Nearest Neighbor Suppression): 여러 개의 앵커 박스 중에서 신뢰도가 높은 객체를 선택하고 겹치는 바운딩 박스를 제거하기 위해 NMS를 적용합니다.

NMS는 IoU(Intersection over Union) 값을 기준으로 겹치는 바운딩 박스를 제거하여 최종 객체 탐지 결과를 얻습니다.

SSD 알고리즘은 단일 네트워크를 사용하고 이미지를 다양한 스케일로 변환하여 다양한 크기와 종횡비의 객체를 탐지할 수 있으며, 높은 정확도와 빠른 속도를 동시에 제공하는 장점을 가지고 있습니다.

SSD의 특징과 장점

- 싱글 샷 학습 (Single Shot Learning): SSD는 이름 그대로 단일 네트워크를 사용하여 객체 탐지를 수행합니다. 기존의 R-CNN 계열 알고리즘과 달리, SSD는 객체의 위치와 클래스를 한 번의 순전파만으로 예측합니다. 이로 인해 높은 속도와 빠른 추론이 가능합니다.

- 다양한 스케일의 특징 맵 활용: SSD는 입력 이미지를 다양한 스케일로 변환하여 여러 개의 특징 맵을 생성합니다. 이를 통해 작은 객체부터 큰 객체까지 다양한 크기의 객체를 탐지할 수 있습니다. 이는 다양한 크기의 객체를 처리하기에 적합하며, 작은 객체에 대한 정확도도 높입니다.

- 앵커 박스를 통한 다중 스케일 객체 탐지: SSD는 각 특징 맵에서 앵커 박스를 생성하여 객체의 위치와 클래스를 예측합니다. 앵커 박스는 다양한 크기와 종횡비를 가지며, 특징 맵의 각 위치에 대해 여러 개의 앵커 박스를 적용하여 다중 스케일 객체 탐지를 수행합니다.

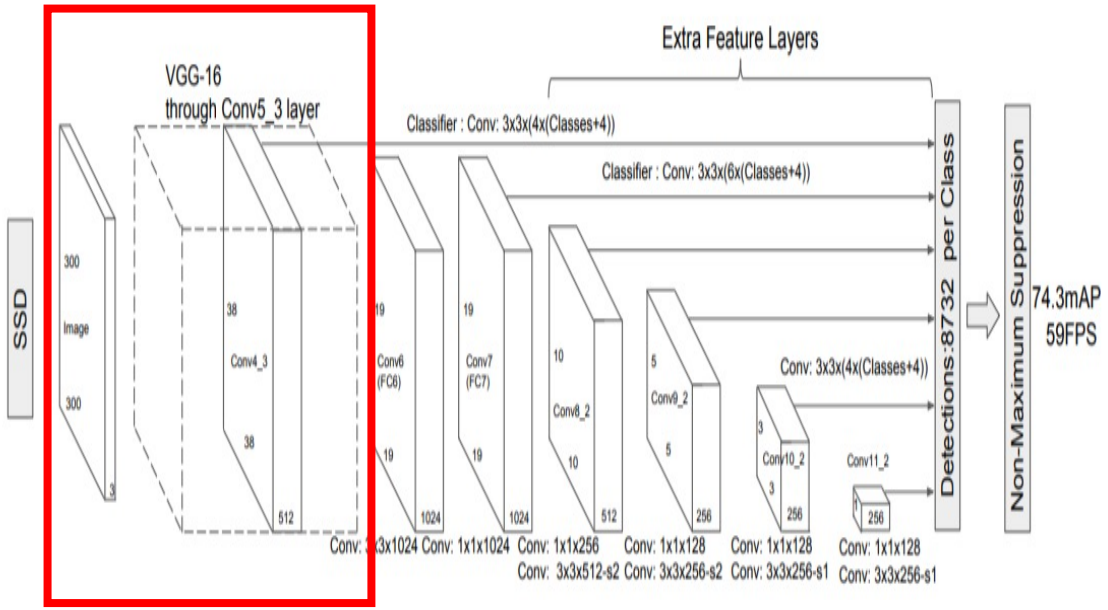
 # SSD의 특징과 장점

높은 정확도와 빠른 속도: SSD는 정확도와 속도 사이에서 적절한 trade-off를 제공합니다. 작은 크기의 객체에 대해서도 높은 정확도를 달성하면서도 빠른 추론 속도를 제공하는 특징으로, 실시간 객체 탐지나 모바일 기기에서의 객체 탐지에 많이 사용됩니다.

간단한 구조와 쉬운 구현: SSD는 비교적 간단한 구조를 갖고 있으며, 모델 구현이 비교적 쉽습니다.
또한, SSD의 구조는 다른 객체 탐지 모델과 결합하여 사용하기 쉽습니다.

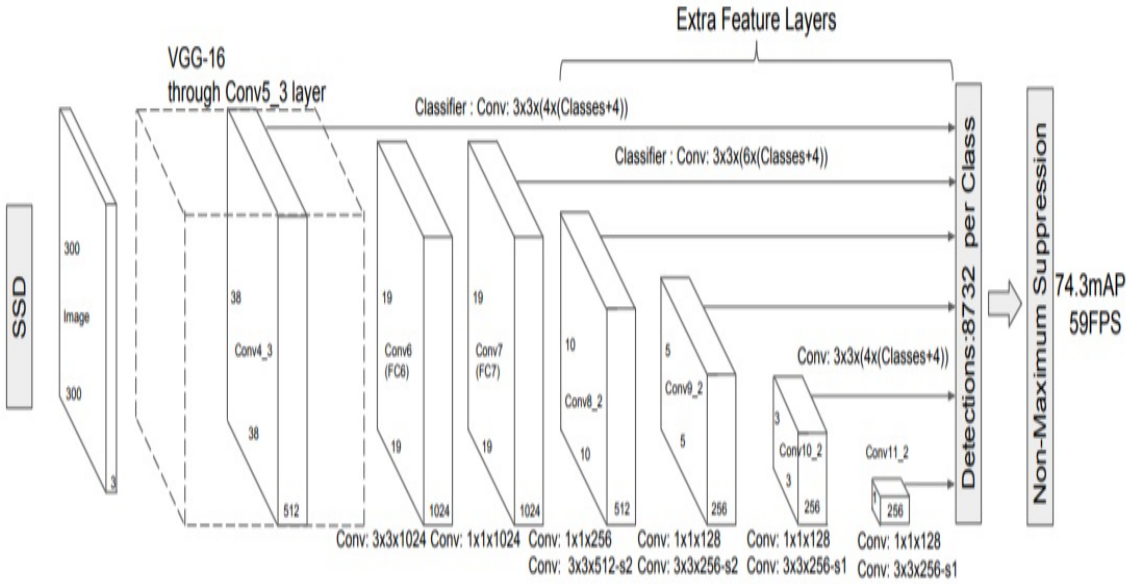
SSD의 특징과 장점으로 인해 실시간 객체 탐지와 모바일 기기에서의 객체 탐지 등 다양한 응용 분야에서 널리 사용되고 있습니다.

네트워크 아키텍처와 레이어 구조 설명



- 네트워크 아키텍처: SSD는 기본적으로 VGG 또는 ResNet 등의 컨볼루션 네트워크를 사용하여 특징 추출을 수행합니다.
- 이후 다양한 스케일의 특징 맵을 생성하는데, 각 스케일의 특징 맵에서는 객체의 위치와 클래스를 예측하기 위해 컨볼루션 레이어와 앵커 박스를 사용합니다.

네트워크 아키텍처와 레이어 구조 설명



- 특징 추출 레이어: SSD에서는 이미지를 다양한 스케일로 변환하여 특징 맵을 생성하는데, 이때 VGG 또는 ResNet과 같은 기존의 컨볼루션 네트워크를 사용합니다.

→ 이렇게 생성된 특징 맵은 객체의 다양한 크기와 종횡비를 잡아낼 수 있도록 합니다.

- 객체 예측 레이어: 각 스케일의 특징 맵에서는 객체의 위치와 클래스를 예측하기 위한 객체 예측 레이어를 추가합니다.

- 이 레이어에서는 컨볼루션 레이어를 통해 앵커 박스를 생성하고, 객체의 위치와 크기를 바운딩 박스로 표현하며, 객체의 클래스를 예측하는데 Softmax 함수를 사용합니다.

→ 이를 통해 객체 탐지를 클래스 분류와 바운딩 박스 조정 문제로 변환하여 다중 객체 탐지를 수행합니다.


감사합니다.