**You Only Look Once Unified, Real-Time Object Detection**

Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi

**Abstract**

We present YOLO, a new approach to object detection. Prior work on object detection repurposes classifiers to perform detection. Instead, we frame object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. A single neural network predicts bounding boxes and class probabilities directly from full images in one evaluation. Since the whole detection pipeline is a single network, it can be optimized end-to-end directly on detection performance.

물체 탐지에 대한 새로운 접근 방식인 YOLO를 소개합니다. 개체 탐지에 대한 이전 작업에서는 탐지를 수행하기 위해 분류자를 용도 변경했습니다. 대신 공간적으로 분리된 경계 상자 및 관련 클래스 확률에 대한 회귀 문제로 객체 감지를 정의합니다. 단일 신경망은 하나의 평가에서 전체 이미지에서 직접 경계 상자와 클래스 확률을 예측합니다. 전체 탐지 파이프라인이 단일 네트워크이므로 탐지 성능에서 직접 엔드 투 엔드 최적화할 수 있습니다.

Our unified architecture is extremely fast. Our base YOLO model processes images in real-time at 45 frames per second. A smaller version of the network, Fast YOLO, processes an astounding 155 frames per second while still achieving double the mAP of other real-time detectors. Compared to state-of-the-art detection systems, YOLO makes more localization errors but is less likely to predict false positives on background. Finally, YOLO learns very general representations of objects. It outperforms other detection methods, including DPM and R-CNN, when generalizing from natural images to other domains like artwork.

NAT의 유니파이드 아키텍처는 매우 빠릅니다. 우리의 기본 YOLO 모델은 초당 45 프레임의 속도로 실시간으로 이미지를 처리합니다. 소규모 네트워크 버전인 Fast YOLO는 초당 155 프레임을 처리하면서도 다른 실시간 검출기의 mAP를 두 배로 늘립니다. 최신 탐지 시스템에 비해 YOLO는 더 많은 지역화 오류를 발생시키지만 배경에서 잘못된 긍정을 예측할 가능성은 낮습니다. 마지막으로, YOLO는 사물에 대한 매우 일반적인 표현을 배웁니다. 자연 이미지에서 아트워크와 같은 다른 도메인으로 일반화할 때 DPM, R-CNN 등의 다른 탐지 방법을 능가합니다.

1. **Introduction**

Humans glance at an image and instantly know what objects are in the image, where they are, and how they interact. The human visual system is fast and accurate, allowing us to perform complex tasks like driving with little conscious thought. Fast, accurate algorithms for object detection would allow computers to drive cars without specialized sensors, enable assistive devices to convey real-time scene information to human users, and unlock the potential for general purpose, responsive robotic systems.

인간은 이미지를 힐끗 보고 이미지 안에 어떤 물체가 있는지, 어디에 있는지, 어떻게 상호작용하는지 즉시 알 수 있습니다. 인간의 시각시스템은 빠르고 정확해서 의식적인 사고가 거의 없는 운전과 같은 복잡한 작업을 수행할 수 있습니다. 빠르고 정확한 물체 감지 알고리즘을 통해 컴퓨터는 특수 센서 없이 자동차를 운전할 수 있고, 보조 장치는 인간에게 실시간 장면 정보를 전달할 수 있으며, 범용 로봇 시스템의 잠재성을 실현할 수 있습니다.

Current detection systems repurpose classifiers to perform detection. To detect an object, these systems take a classifier for that object and evaluate it at various locations and scales in a test image. Systems like deformable parts models (DPM) use a sliding window approach where the classifier is run at evenly spaced locations over the entire image [10].

현재 탐지 시스템은 탐지를 수행하기 위해 분류자를 용도 변경합니다. 이러한 시스템은 객체를 감지하기 위해 해당 객체에 대한 분류자를 가져와 테스트 이미지의 다양한 위치와 축척에서 평가합니다. DPM(변형 부품 모델)과 같은 시스템은 전체 이미지에 걸쳐 균등한 간격으로 분류기를 실행하는 슬라이딩 윈도우 접근 방식을 사용합니다[10].

More recent approaches like R-CNN use region proposal methods to first generate potential bounding boxes in an image and then run a classifier on these proposed boxes. After classification, post-processing is used to refine the bounding boxes, eliminate duplicate detections, and rescore the boxes based on other objects in the scene [13]. These complex pipelines are slow and hard to optimize because each individual component must be trained separately.

R-CNN과 같은 보다 최근의 접근법은 지역 제안 방법을 사용하여 먼저 이미지에 잠재적 경계 상자를 생성한 다음 이러한 제안된 상자에 분류자를 실행합니다. 분류 후에는 경계 상자를 미세화하고 중복 탐지를 제거하며 씬(scene)의 다른 개체를 기반으로 상자에 점수를 매기는 데 사후 처리가 사용됩니다[13]. 이러한 복잡한 파이프라인은 각 구성 요소를 별도로 교육해야 하기 때문에 속도가 느리고 최적화하기가 어렵습니다.

We reframe object detection as a single regression problem, straight from image pixels to bounding box coordinates and class probabilities. Using our system, you only look once (YOLO) at an image to predict what objects are present and where they are.

개체 탐지를 이미지 픽셀에서 경계 상자 좌표 및 클래스 확률로 직접 단일 회귀 문제로 재구성합니다. 이 시스템을 사용하면 이미지를 한 번만 보고 어떤 물체가 있는지, 어디에 있는지 예측할 수 있습니다.

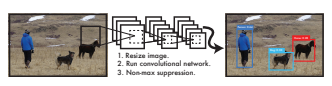


Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448 × 448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model’s confidence.

그림 1: YOLO 감지 시스템. YOLO로 이미지를 처리하는 것은 간단하고 간단합니다. 우리 시스템은 (1) 입력 이미지의 크기를 448 × 448로 조정하고, (2) 이미지에서 단일 컨볼루션 네트워크를 실행하고, (3) 모델의 신뢰도에 따라 결과 감지를 임계값으로 지정합니다.

YOLO is refreshingly simple: see Figure 1. A single convolutional network simultaneously predicts multiple bounding boxes and class probabilities for those boxes. YOLO trains on full images and directly optimizes detection performance. This unified model has several benefits over traditional methods of object detection.

YOLO는 매우 간단합니다. 그림 1을 참조하십시오. 단일 컨볼루션 네트워크는 여러 개의 경계 상자와 해당 상자에 대한 클래스 확률을 동시에 예측합니다. YOLO는 전체 이미지에서 훈련하고 탐지 성능을 직접 최적화합니다. 이 통합 모델은 기존의 객체 감지 방법에 비해 몇 가지 이점이 있습니다.

First, YOLO is extremely fast. Since we frame detection as a regression problem we don’t need a complex pipeline. We simply run our neural network on a new image at test time to predict detections. Our base network runs at 45 frames per second with no batch processing on a Titan X GPU and a fast version runs at more than 150 fps. This means we can process streaming video in real-time with less than 25 milliseconds of latency. Furthermore, YOLO achieves more than twice the mean average precision of other real-time systems. For a demo of our system running in real-time on a webcam please see our project webpage: <http://pjreddie.com/yolo/>.

첫째, YOLO는 매우 빠르다. 우리는 탐지를 회귀 문제로 설정하기 때문에 복잡한 파이프라인이 필요하지 않습니다. 테스트 시간에 새로운 이미지에서 신경망을 작동시켜 탐지를 예측합니다. Titan X GPU에서 일괄 처리 없이 초당 45프레임으로 실행되며 빠른 버전은 150fps 이상에서 실행됩니다. 즉, 25밀리초 미만의 대기 시간으로 스트리밍 비디오를 실시간으로 처리할 수 있습니다. 또한 YOLO는 다른 실시간 시스템의 평균 평균 정밀도의 2배 이상을 달성합니다. 웹캠에서 실시간으로 실행되는 시스템의 데모를 보려면 프로젝트 웹페이지 http://pjreddie.com/yolo/를 참조하십시오.

Second, YOLO reasons globally about the image when making predictions. Unlike sliding window and region proposal-based techniques, YOLO sees the entire image during training and test time so it implicitly encodes contextual information about classes as well as their appearance. Fast R-CNN, a top detection method [14], mistakes background patches in an image for objects because it can’t see the larger context. YOLO makes less than half the number of background errors compared to Fast R-CNN.

둘째, YOLO는 예측 시 이미지에 대해 전 세계적으로 추론합니다. 슬라이딩 윈도우 및 지역 제안 기반 기법과 달리 YOLO는 훈련 및 테스트 시간 동안 전체 이미지를 보기 때문에 수업의 모양뿐 아니라 상황에 맞는 정보를 암시적으로 인코딩합니다. 상위 탐지 방법[14]인 Fast R-CNN은 더 큰 컨텍스트를 볼 수 없기 때문에 이미지의 백그라운드 패치를 개체로 잘못 인식합니다. YOLO는 Fast R-CNN에 비해 백그라운드 오류 수가 절반도 되지 않습니다.

Third, YOLO learns generalizable representations of objects. When trained on natural images and tested on artwork, YOLO outperforms top detection methods like DPM and R-CNN by a wide margin. Since YOLO is highly generalizable it is less likely to break down when applied to new domains or unexpected inputs.

셋째, YOLO는 사물의 일반화된 표현을 학습합니다. 자연 이미지를 교육하고 아트워크를 테스트하면 YOLO가 DPM, R-CNN과 같은 상위 탐지 방법을 큰 차이로 능가합니다. YOLO는 일반화가 용이하기 때문에 새로운 도메인이나 예상치 못한 입력에 적용될 경우 고장이 발생할 가능성이 적습니다.

YOLO still lags behind state-of-the-art detection systems in accuracy. While it can quickly identify objects in images it struggles to precisely localize some objects, especially small ones. We examine these tradeoffs further in our experiments.

All of our training and testing code is open source. A variety of pretrained models are also available to download.

YOLO는 여전히 최첨단 탐지 시스템에 비해 정확도가 떨어집니다. 이미지에서 객체를 빠르게 식별할 수 있지만 일부 객체, 특히 작은 객체의 위치를 정확하게 파악하지 못합니다. 우리는 우리의 실험에서 이러한 단점을 더 검토한다.

우리의 모든 교육 및 테스트 코드는 오픈 소스입니다. 사전 교육된 다양한 모델도 다운로드할 수 있습니다.

1. **Unified Detection**

We unify the separate components of object detection into a single neural network. Our network uses features from the entire image to predict each bounding box. It also predicts all bounding boxes across all classes for an image simultaneously. This means our network reasons globally about the full image and all the objects in the image. The YOLO design enables end-to-end training and realtime speeds while maintaining high average precision.

우리는 물체 탐지의 별도 구성 요소를 하나의 신경망으로 통합합니다. 네트워크에서는 전체 이미지의 기능을 사용하여 각 경계 상자를 예측합니다. 또한 이미지의 모든 클래스에 걸쳐 모든 경계 상자를 동시에 예측합니다. 이는 전체 이미지와 이미지에 있는 모든 개체에 대한 당사의 네트워크 상의 전 세계적인 이유를 의미합니다. YOLO 설계는 높은 평균 정밀도를 유지하면서 엔드 투 엔드 교육 및 실시간 속도를 가능하게 합니다.

Our system divides the input image into an S × S grid. If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object.

우리 시스템은 입력 이미지를 S × S 그리드로 나눕니다. 개체의 중심이 그리드 셀에 속하는 경우 해당 그리드 셀이 해당 개체를 탐지하는 역할을 합니다.

Each grid cell predicts B bounding boxes and confidence scores for those boxes. These confidence scores reflect how confident the model is that the box contains an object and also how accurate it thinks the box is that it predicts. Formally we define confidence as Pr(Object) ∗ IOUtruth pred . If no object exists in that cell, the confidence scores should be zero. Otherwise we want the confidence score to equal the intersection over union (IOU) between the predicted box and the ground truth.

각 그리드 셀은 B 경계 상자와 해당 상자에 대한 신뢰 점수를 예측합니다. 이러한 신뢰 점수는 상자에 개체가 포함되어 있다는 모형의 신뢰도와 예측 상자의 정확성을 나타냅니다. 공식적으로 우리는 신뢰를 Pr (객체) → IO 진실 예측으로 정의한다. 만약 그 셀에 물체가 존재하지 않는다면, 신뢰 점수는 0이어야 한다. 그렇지 않으면 신뢰 점수가 예측 상자와 지상 진실 사이의 교집합 초과 합집합(IOU)과 같게 됩니다.

Each bounding box consists of 5 predictions: x, y, w, h, and confidence. The (x, y) coordinates represent the center of the box relative to the bounds of the grid cell. The width and height are predicted relative to the whole image. Finally the confidence prediction represents the IOU between the predicted box and any ground truth box.

각 경계 상자는 x, y, w, h 및 신뢰의 5가지 예측으로 구성됩니다. (x, y) 좌표는 그리드 셀의 경계에 상대적인 상자의 중심을 나타냅니다. 전체 이미지를 기준으로 너비와 높이가 예측됩니다. 마지막으로 신뢰 예측은 예측 상자와 지면 진실 상자 사이의 IOU를 나타냅니다.



Each grid cell also predicts C conditional class probabilities, Pr(Classi |Object). These probabilities are conditioned on the grid cell containing an object. We only predict one set of class probabilities per grid cell, regardless of the number of boxes B.

At test time we multiply the conditional class probabilities and the individual box confidence predictions, which gives us class-specific confidence scores for each box. These scores encode both the probability of that class appearing in the box and how well the predicted box fits the object.

각 그리드 셀은 또한 C 조건부 클래스 확률 Pr(Classi | Object)을 예측합니다. 이러한 확률은 객체를 포함하는 그리드 셀에서 조절됩니다. 우리는 박스 B의 수와 상관없이 그리드 셀당 클래스 확률 세트를 하나만 예측합니다.

테스트 시 조건부 클래스 확률과 개별 상자 신뢰 예측을 곱하여 각 상자에 대한 클래스별 신뢰 점수를 제공합니다. 이 점수는 상자에 해당 클래스가 나타날 확률과 예측 상자가 객체에 얼마나 적합한지 모두 인코딩합니다.

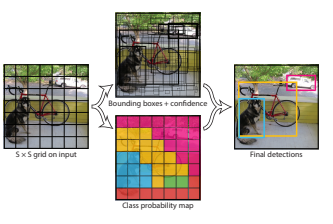


Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an S × S grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an S × S × (B ∗ 5 + C) tensor.

그림 2: 모델. 우리 시스템은 회귀 문제로 탐지를 모델링합니다. 이미지를 S × S 그리드로 나누고 각 그리드 셀에 대해 B 경계 상자, 해당 상자에 대한 신뢰도 및 C 클래스 확률을 예측합니다. 이러한 예측은 S × S × (B \* 5 + C) 텐서로 인코딩됩니다.

For evaluating YOLO on PASCAL VOC, we use S = 7, B = 2. PASCAL VOC has 20 labelled classes so C = 20. Our final prediction is a 7 × 7 × 30 tensor.

PASCAL VOC에 대한 YOLO 평가를 위해 S = 7, B = 2. PASCAL VOC에는 20개의 라벨 클래스가 있으므로 C = 20을 사용합니다. 우리의 최종 예측은 7 × 7 × 30 텐서입니다.

* 1. **Network Design**

We implement this model as a convolutional neural network and evaluate it on the PASCAL VOC detection dataset [9]. The initial convolutional layers of the network extract features from the image while the fully connected layers predict the output probabilities and coordinates.

우리는 이 모델을 컨볼루션 신경망으로 구현하고 PASCAL VOC 검출 데이터셋[9]에서 평가한다. 완전히 연결된 레이어가 출력 확률과 좌표를 예측하는 동안 네트워크의 초기 컨볼루션 레이어는 이미지에서 피쳐를 추출합니다.

Our network architecture is inspired by the GoogLeNet model for image classification [34]. Our network has 24 convolutional layers followed by 2 fully connected layers. Instead of the inception modules used by GoogLeNet, we simply use 1 × 1 reduction layers followed by 3 × 3 convolutional layers, similar to Lin et al [22]. The full network is shown in Figure 3.

당사의 네트워크 아키텍처는 이미지 분류용 GoogleLeNet 모델에서 영감을 받았습니다 [34]. 우리 네트워크에는 24개의 컨볼루션 레이어가 있고 이어서 2개의 완전히 연결된 레이어가 있습니다. GoogleLeNet에 의해 사용되는 인셉션 모듈 대신에, 우리는 라인 외[22]와 유사하게 1 × 1 감소 계층 다음에 3 × 3 컨볼루션 계층을 사용합니다. 전체 네트워크는 그림 3에 나와 있습니다.

We also train a fast version of YOLO designed to push the boundaries of fast object detection. Fast YOLO uses a neural network with fewer convolutional layers (9 instead of 24) and fewer filters in those layers. Other than the size of the network, all training and testing parameters are the same between YOLO and Fast YOLO.

또한 빠른 객체 감지의 경계를 넓히도록 설계된 빠른 버전의 YOLO를 교육합니다. Fast YOLO는 더 적은 수의 컨볼루션 레이어(24개 대신 9개)와 더 적은 수의 필터를 가진 신경 네트워크를 사용합니다. 네트워크 크기 외에 모든 교육 및 테스트 매개변수는 YOLO와 Fast YOLO 간에 동일합니다.

The final output of our network is the 7 × 7 × 30 tensor of predictions.

우리 네트워크의 최종 출력은 예측의 7 × 7 × 30 텐서입니다.

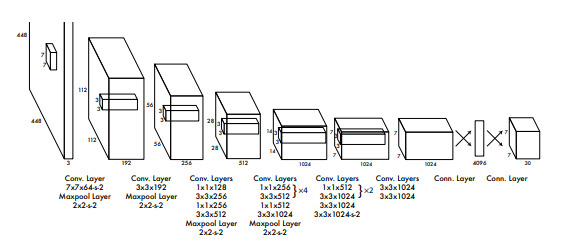


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1 × 1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224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

그림 3: 아키텍처. 우리의 탐지 네트워크에는 24개의 컨볼루션 레이어와 2개의 완전 연결 레이어가 있습니다. 1 × 1 컨볼루션 레이어를 교대로 사용하면 이전 레이어의 피쳐 공간이 줄어듭니다. ImageNet 분류 작업에서 합성곱 계층을 해상도의 절반(224 × 224 입력 이미지)으로 사전 훈련한 다음 감지를 위해 해상도를 두 배로 늘립니다.

**2.2. Training**

We pretrain our convolutional layers on the ImageNet 1000-class competition dataset [30]. For pretraining we use the first 20 convolutional layers from Figure 3 followed by a average-pooling layer and a fully connected layer. We train this network for approximately a week and achieve a single crop top-5 accuracy of 88% on the ImageNet 2012 validation set, comparable to the GoogLeNet models in Caffe’s Model Zoo [24]. We use the Darknet framework for all training and inference [26].

우리는 ImageNet 1000 클래스 경쟁 데이터셋에서 컨볼루션 계층을 사전 교육합니다 [30]. 사전 교육에서는 그림 3의 처음 20개의 컨볼루션 레이어를 사용한 다음 평균 풀링 레이어와 완전히 연결된 레이어를 사용합니다. 우리는 이 네트워크를 약 일주일 동안 교육하고 카페 모델 동물원[24]의 GoogleLeNet 모델과 비교할 수 있는 ImageNet 2012 검증 세트에서 최고 정확도 상위 5위까지 단일 크롭을 달성합니다. 우리는 모든 훈련과 추론을 위해 Darknet 프레임워크를 사용합니다 [26].

We then convert the model to perform detection. Ren et al. show that adding both convolutional and connected layers to pretrained networks can improve performance [29]. Following their example, we add four convolutional layers and two fully connected layers with randomly initialized weights. Detection often requires fine-grained visual information so we increase the input resolution of the network from 224 × 224 to 448 × 448.

그런 다음 모델을 변환하여 탐지를 수행합니다. Ren 등은 사전 훈련된 네트워크에 컨볼루션 및 연결된 레이어를 모두 추가할 경우 성능이 향상될 수 있음을 보여줍니다 [29]. 그들의 예에 따라, 우리는 무작위로 초기화된 가중치를 가진 네 개의 컨볼루션 레이어와 두 개의 완전히 연결된 레이어를 추가합니다. 탐지는 종종 세분화된 시각적 정보를 요구하기 때문에 네트워크의 입력 해상도를 224 × 224에서 448 × 448로 높입니다.

Our final layer predicts both class probabilities and bounding box coordinates. We normalize the bounding box width and height by the image width and height so that they fall between 0 and 1. We parametrize the bounding box x and y coordinates to be offsets of a particular grid cell location so they are also bounded between 0 and 1.

We use a linear activation function for the final layer and all other layers use the following leaky rectified linear activation:

최종 레이어는 클래스 확률과 경계 상자 좌표를 모두 예측합니다. 이미지 폭과 높이로 경계 상자 폭과 높이를 0과 1 사이로 정규화합니다. 우리는 경계 상자 x와 y 좌표가 특정 그리드 셀 위치의 오프셋이 되도록 매개변수를 지정하여 그것들이 또한 0과 1 사이에서 경계를 이루도록 합니다.

최종 레이어에 대해 선형 활성화 함수를 사용하고 다른 모든 레이어는 다음과 같이 누수 정류된 선형 활성화를 사용합니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

We optimize for sum-squared error in the output of our model. We use sum-squared error because it is easy to optimize, however it does not perfectly align with our goal of maximizing average precision. It weights localization error equally with classification error which may not be ideal. Also, in every image many grid cells do not contain any object. This pushes the “confidence” scores of those cells towards zero, often overpowering the gradient from cells that do contain objects. This can lead to model instability, causing training to diverge early on.

우리는 모델 출력의 합계 오차를 최적화합니다. 최적화가 쉽기 때문에 합제곱 오차를 사용하지만, 평균 정밀도를 극대화하려는 우리의 목표와 완벽하게 일치하지는 않습니다. 이상적이지 않을 수 있는 분류 오류와 동등하게 지역화 오류를 가중치 부여합니다. 또한 모든 이미지에서 많은 그리드 셀은 개체를 포함하지 않습니다. 이렇게 하면 해당 셀의 "자신감" 점수가 0으로 이동하며, 종종 객체가 포함된 셀의 기울기를 압도합니다. 이로 인해 모델이 불안정해질 수 있으며, 이로 인해 훈련이 조기에 분산될 수 있습니다.

To remedy this, we increase the loss from bounding box coordinate predictions and decrease the loss from confidence predictions for boxes that don’t contain objects. We use two parameters, λcoord and λnoobj to accomplish this. We set λcoord = 5 and λnoobj = .5.

이를 해결하기 위해 바운딩 박스 좌표 예측으로 인한 손실을 늘리고 객체가 포함되지 않은 박스에 대한 신뢰 예측으로 인한 손실을 줄입니다. 이를 위해 λcoord와 λnoobj의 두 가지 매개변수를 사용합니다. λcoord = 5 및 λnoobj =.5를 설정합니다.

Sum-squared error also equally weights errors in large boxes and small boxes. Our error metric should reflect that small deviations in large boxes matter less than in small boxes. To partially address this we predict the square root of the bounding box width and height instead of the width and height directly.

합제곱 오차도 큰 상자와 작은 상자의 오차의 가중치를 동일하게 매깁니다. 우리의 오류 메트릭은 큰 상자의 작은 편차가 작은 상자보다 덜 중요하다는 것을 반영해야 합니다. 이 문제를 부분적으로 해결하기 위해 우리는 너비와 높이가 아닌 경계 상자 폭과 높이의 제곱근을 예측합니다.

YOLO predicts multiple bounding boxes per grid cell. At training time we only want one bounding box predictor to be responsible for each object. We assign one predictor to be “responsible” for predicting an object based on which prediction has the highest current IOU with the ground truth. This leads to specialization between the bounding box predictors. Each predictor gets better at predicting certain sizes, aspect ratios, or classes of object, improving overall recall.

YOLO는 그리드 셀당 여러 개의 경계 상자를 예측합니다. 교육 시간에는 경계 상자 예측 변수 한 개만 각 개체에 대해 담당할 수 있도록 합니다. 우리는 예측에 근거하여 지면 진실에 대한 전류 IOU가 가장 높은 물체를 예측하는 "책임감"을 가진 예측 변수 하나를 지정합니다. 따라서 경계 상자 예측 변수 간의 전문화가 이루어집니다. 각 예측 변수는 특정 크기, 가로 세로 비율 또는 개체 클래스를 더 잘 예측하여 전체 리콜을 개선합니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

During training we optimize the following, multi-part loss function: where ✶ obj i denotes if object appears in cell i and ✶obj ij denotes that the jth bounding box predictor in cell i is “responsible” for that prediction.

훈련 중에 우리는 다음과 같은 다중 부품 손실 함수를 최적화한다. 여기서 obobji는 셀 i에 물체가 표시되는지 여부를 나타내고 jobjij는 셀 i의 j번째 경계 상자 예측 변수가 해당 예측에 "책임" 있음을 나타낸다.

Note that the loss function only penalizes classification error if an object is present in that grid cell (hence the conditional class probability discussed earlier). It also only penalizes bounding box coordinate error if that predictor is “responsible” for the ground truth box (i.e. has the highest IOU of any predictor in that grid cell).

손실 함수는 물체가 그리드 셀에 존재하는 경우에만 분류 오류를 처벌한다는 점에 유의한다(따라서 앞에서 설명한 조건부 등급 확률). 또한 해당 예측 변수가 접지 진실 상자에 대해 "책임감" 있는 경우에만 경계 상자 좌표 오차를 처벌합니다(즉, 해당 그리드 셀에서 예측 변수 중 가장 높은 IOU를 갖습니다).

We train the network for about 135 epochs on the training and validation data sets from PASCAL VOC 2007 and 2012. When testing on 2012 we also include the VOC 2007 test data for training. Throughout training we use a batch size of 64, a momentum of 0.9 and a decay of 0.0005.

PASCAL VOC 2007 및 2012의 교육 및 검증 데이터 세트에 대해 약 135개 시대에 걸쳐 네트워크를 교육합니다. 2012년 테스트 시 VOC 2007 테스트 데이터도 함께 포함시켜 교육합니다. 교육 과정 내내 배치 크기 64, 운동량 0.9, 붕괴율 0.0005를 사용합니다.

Our learning rate schedule is as follows: For the first epochs we slowly raise the learning rate from 10−3 to 10−2. If we start at a high learning rate our model often diverges due to unstable gradients. We continue training with 10−2 for 75 epochs, then 10−3 for 30 epochs, and finally 10−4 for 30 epochs.

당사의 학습 속도 일정은 다음과 같습니다. 첫 번째 시대는 학습률을 10-3에서 10-2로 서서히 높입니다. 우리가 높은 학습률로 시작하면 불안정한 구배로 인해 우리의 모델이 자주 갈라집니다. 75개 시대는 10-2, 30개 시대는 10-3, 30개 시대는 10-4로 훈련을 계속한다.

To avoid overfitting we use dropout and extensive data augmentation. A dropout layer with rate = .5 after the first connected layer prevents co-adaptation between layers [18]. For data augmentation we introduce random scaling and translations of up to 20% of the original image size. We also randomly adjust the exposure and saturation of the image by up to a factor of 1.5 in the HSV color space.

과부하를 방지하기 위해 중퇴 및 광범위한 데이터 확장을 사용합니다. 레이트가 =.5인 드롭아웃 레이어는 첫 번째 연결된 레이어 이후에 레이어 간 협업을 방지합니다 [18]. 데이터 확장을 위해 원본 이미지 크기의 최대 20%까지 랜덤 스케일링 및 변환 기능을 도입합니다. 또한 HSV 컬러 공간에서 이미지의 노출과 채도를 최대 1.5배까지 임의로 조절합니다.

**2.3. Inference**

Just like in training, predicting detections for a test image only requires one network evaluation. On PASCAL VOC the network predicts 98 bounding boxes per image and class probabilities for each box. YOLO is extremely fast at test time since it only requires a single network evaluation, unlike classifier-based methods.

교육에서와 마찬가지로 테스트 이미지에 대한 탐지를 예측하려면 하나의 네트워크 평가만 필요합니다. PASCAL VOC에서 네트워크는 이미지당 98개의 경계 상자 및 각 상자에 대한 클래스 확률을 예측합니다. YOLO는 분류자 기반 방식과 달리 하나의 네트워크 평가만 요구하기 때문에 테스트 시 속도가 매우 빠릅니다.

The grid design enforces spatial diversity in the bounding box predictions. Often it is clear which grid cell an object falls in to and the network only predicts one box for each object. However, some large objects or objects near the border of multiple cells can be well localized by multiple cells. Non-maximal suppression can be used to fix these multiple detections. While not critical to performance as it is for R-CNN or DPM, non-maximal suppression adds 2-3% in mAP.

그리드 설계는 경계 상자 예측에 공간 다양성을 적용합니다. 종종 객체가 어느 그리드 셀에 속하는지 명확하고 네트워크는 각 객체에 대해 하나의 상자만 예측합니다. 그러나 여러 셀의 경계 부근에 있는 일부 큰 객체나 객체는 여러 셀에 의해 잘 지역화될 수 있습니다. 최대가 아닌 억제를 사용하여 이러한 여러 탐지를 수정할 수 있습니다. R-CNN 또는 DPM과 같이 성능에 중요하지는 않지만, 최대값이 아닌 억제는 mAP에서 2-3%를 추가합니다.

**2.4. Limitations of YOLO**

YOLO imposes strong spatial constraints on bounding box predictions since each grid cell only predicts two boxes and can only have one class. This spatial constraint limits the number of nearby objects that our model can predict. Our model struggles with small objects that appear in groups, such as flocks of birds.

각 그리드 셀은 두 개의 상자만 예측하고 클래스를 하나만 가질 수 있으므로 YOLO는 경계 상자 예측에 강력한 공간 제약을 가합니다. 이 공간 제약으로 인해 모델이 예측할 수 있는 주변 객체의 수가 제한됩니다. 우리의 모델은 새떼처럼 무리를 지어 나타나는 작은 물체들과 싸웁니다.

Since our model learns to predict bounding boxes from data, it struggles to generalize to objects in new or unusual aspect ratios or configurations. Our model also uses relatively coarse features for predicting bounding boxes since our architecture has multiple downsampling layers from the input image.

NAT 모델은 데이터에서 경계 상자를 예측하는 방법을 학습하기 때문에 새로운 또는 비정상적인 가로 세로 비율 또는 구성의 객체로 일반화하기가 어렵습니다. 또한 아키텍처에는 입력 이미지에서 여러 개의 다운샘플링 레이어가 있기 때문에 이 모델은 경계 상자를 예측하는 데 비교적 거친 기능을 사용합니다.

Finally, while we train on a loss function that approximates detection performance, our loss function treats errors the same in small bounding boxes versus large bounding boxes. A small error in a large box is generally benign but a small error in a small box has a much greater effect on IOU. Our main source of error is incorrect localizations.

마지막으로, 검출 성능에 가까운 손실 함수에 대해 교육하는 동안 손실 함수는 작은 경계 상자 대 큰 경계 상자의 오류를 동일하게 처리합니다. 큰 상자의 작은 오류는 일반적으로 양호하지만 작은 상자의 작은 오류는 IOU에 훨씬 더 큰 영향을 미칩니다. 주요 오류 원인은 잘못된 위치 지정입니다.

1. **Comparison to Other Detection Systems**

Object detection is a core problem in computer vision. Detection pipelines generally start by extracting a set of robust features from input images (Haar [25], SIFT [23], HOG [4], convolutional features [6]). Then, classifiers [36, 21, 13, 10] or localizers [1, 32] are used to identify objects in the feature space. These classifiers or localizers are run either in sliding window fashion over the whole image or on some subset of regions in the image [35, 15, 39]. We compare the YOLO detection system to several top detection frameworks, highlighting key similarities and differences.

객체 감지는 컴퓨터 비전의 핵심 문제입니다. 검출 파이프라인은 일반적으로 입력 이미지에서 일련의 강력한 특성(Haar [25], SHIFT [23], HOG [4], convolution features [6])을 추출하는 것으로 시작한다. 그런 다음 분류자 [36, 21, 13, 10] 또는 로컬라이저[1, 32]를 사용하여 피쳐 공간의 객체를 식별합니다. 이러한 분류자 또는 로컬라이저는 전체 이미지에 슬라이딩 윈도우 방식으로 실행되거나 이미지의 일부 하위 집합에서 실행됩니다 [35, 15, 39]. YOLO 검출 시스템을 여러 상위 검출 프레임워크와 비교하여 주요 유사점과 차이점을 강조합니다.

**Deformable parts models.** Deformable parts models (DPM) use a sliding window approach to object detection [10]. DPM uses a disjoint pipeline to extract static features, classify regions, predict bounding boxes for high scoring regions, etc. Our system replaces all of these disparate parts with a single convolutional neural network. The network performs feature extraction, bounding box prediction, nonmaximal suppression, and contextual reasoning all concurrently. Instead of static features, the network trains the features in-line and optimizes them for the detection task. Our unified architecture leads to a faster, more accurate model than DPM.

변형 가능한 부품 모델. 변형 가능한 부품 모델(DPM)은 슬라이딩 윈도우 접근 방식을 사용하여 물체를 감지합니다[10]. DPM은 분리 파이프라인을 사용하여 정적 기능을 추출하고, 영역을 분류하며, 점수가 높은 영역에 대한 경계 상자를 예측합니다. 우리 시스템은 이질적인 부분들을 하나의 콘볼루션 신경망으로 대체합니다. 네트워크는 피쳐 추출, 경계 상자 예측, 비최대 억제 및 상황 추론을 동시에 수행합니다. 정적 피쳐 대신 네트워크는 인라인 방식으로 피쳐를 교육하고 탐지 작업에 최적화합니다. NAT의 유니파이드 아키텍처는 DPM보다 더 빠르고 정확한 모델로 이어집니다.

**R-CNN.** R-CNN and its variants use region proposals instead of sliding windows to find objects in images. Selective Search [35] generates potential bounding boxes, a convolutional network extracts features, an SVM scores the boxes, a linear model adjusts the bounding boxes, and non-max suppression eliminates duplicate detections. Each stage of this complex pipeline must be precisely tuned independently and the resulting system is very slow, taking more than 40 seconds per image at test time [14].

R-CNN. R-CNN과 그 변형 모델들은 슬라이딩 창 대신 지역 제안을 사용하여 이미지의 객체를 찾습니다. 선택적 검색[35]은(는) 잠재적 경계 상자를 생성하고, 컨볼루션 네트워크에서 기능을 추출하며, SVM은 상자를 점수화하고, 선형 모델은 경계 상자를 조정하며, 최대값이 아닌 억제는 중복 탐지를 제거합니다. 이 복잡한 파이프라인의 각 단계는 독립적으로 정밀하게 튜닝되어야 하며 결과 시스템은 테스트 시간에 이미지당 40초 이상 소요될 정도로 매우 느립니다 [14].

YOLO shares some similarities with R-CNN. Each grid cell proposes potential bounding boxes and scores those boxes using convolutional features. However, our system puts spatial constraints on the grid cell proposals which helps mitigate multiple detections of the same object. Our system also proposes far fewer bounding boxes, only 98 per image compared to about 2000 from Selective Search. Finally, our system combines these individual components into a single, jointly optimized model.

YOLO는 R-CNN과 몇 가지 유사점을 공유한다. 각 그리드 셀은 잠재적 경계 상자를 제안하고 이러한 상자에 컨볼루션 특성을 사용하여 점수를 매긴다. 그러나, 우리의 시스템은 동일한 물체의 다중 탐지를 완화하는 데 도움이 되는 그리드 셀 제안에 공간 제약을 가한다. 우리 시스템은 또한 훨씬 적은 수의 경계 상자를 제안하는데, 이는 선택 검색의 약 2000년에 비해 이미지당 98개밖에 되지 않습니다. 마지막으로, NAT 시스템은 이러한 개별 구성요소를 공동으로 최적화된 단일 모델로 결합합니다.

**Other Fast Detectors** Fast and Faster R-CNN focus on speeding up the R-CNN framework by sharing computation and using neural networks to propose regions instead of Selective Search [14] [28]. While they offer speed and accuracy improvements over R-CNN, both still fall short of real-time performance.

다른 고속 검출기 R-CNN은 계산을 공유하고 선택 검색 대신 신경망을 사용하여 영역을 제안함으로써 R-CNN 프레임워크의 속도를 높이는 데 초점을 맞춥니다 [14] [28]. R-CNN에 비해 속도와 정확도가 향상됐지만 실시간 성능에는 미치지 못한다.

Many research efforts focus on speeding up the DPM pipeline [31] [38] [5]. They speed up HOG computation, use cascades, and push computation to GPUs. However, only 30Hz DPM [31] actually runs in real-time.

Instead of trying to optimize individual components of a large detection pipeline, YOLO throws out the pipeline entirely and is fast by design.

Detectors for single classes like faces or people can be highly optimized since they have to deal with much less variation [37]. YOLO is a general purpose detector that learns to detect a variety of objects simultaneously.

많은 연구가 DPM 파이프라인의 속도를 높이는 데 초점을 맞추고 있습니다 [31] [38] [5]. HOG 연산 속도를 높이고, 캐스케이드를 사용하며, GPU에 연산을 푸시합니다. 그러나 실제로 실시간으로 실행되는 DPM은 30Hz [31]에 불과합니다.

YOLO는 대규모 탐지 파이프라인의 개별 구성요소를 최적화하기보다는 파이프라인을 완전히 폐기하고 설계상 속도가 빠릅니다.

얼굴이나 사람과 같은 단일 클래스에 대한 검출기는 훨씬 적은 변동을 처리해야 하기 때문에 매우 최적화될 수 있다[37]. YOLO는 다양한 물체를 동시에 감지하는 것을 학습하는 범용 검출기입니다.

**Deep MultiBox.** Unlike R-CNN, Szegedy et al. train a convolutional neural network to predict regions of interest [8] instead of using Selective Search. MultiBox can also perform single object detection by replacing the confidence prediction with a single class prediction. However, MultiBox cannot perform general object detection and is still just a piece in a larger detection pipeline, requiring further image patch classification. Both YOLO and MultiBox use a convolutional network to predict bounding boxes in an image but YOLO is a complete detection system.

Deep MultiBox. R-CNN과 달리, Szegedy 등은 선택적 검색을 사용하는 대신 관심 영역을 예측하기 위해 [8] 컨볼루션 신경망을 훈련시킨다. MultiBox는 신뢰 예측을 단일 클래스 예측으로 대체하여 단일 개체 탐지를 수행할 수도 있습니다. 그러나 MultiBox는 일반적인 개체 탐지를 수행할 수 없으며 더 큰 탐지 파이프라인의 일부에 불과하므로 추가 이미지 패치 분류가 필요합니다. YOLO와 MultiBox 모두 이미지의 경계 상자를 예측하기 위해 컨볼루션 네트워크를 사용하지만, YOLO는 완전한 탐지 시스템입니다.

**OverFeat.** Sermanet et al. train a convolutional neural network to perform localization and adapt that localizer to perform detection [32]. OverFeat efficiently performs sliding window detection but it is still a disjoint system. OverFeat optimizes for localization, not detection performance. Like DPM, the localizer only sees local information when making a prediction. OverFeat cannot reason about global context and thus requires significant post-processing to produce coherent detections.

오버 Feat. Sermanet 등은 국소화를 수행하고 검출 수행을 위해 해당 국소화기를 조정하도록 컨볼루션 신경망을 훈련시킵니다 [32]. OverFeat는 슬라이딩 윈도우 감지를 효율적으로 수행하지만 여전히 분리 시스템입니다. OverFeat는 탐지 성능이 아니라 로컬라이제이션에 최적화됩니다. DPM과 마찬가지로 localizer는 예측을 할 때만 로컬 정보를 봅니다. OverFeat는 전역 컨텍스트에 대해 추론할 수 없으므로 일관성 있는 탐지를 생성하려면 상당한 사후 처리가 필요합니다.

**MultiGrasp**. Our work is similar in design to work on grasp detection by Redmon et al [27]. Our grid approach to bounding box prediction is based on the MultiGrasp system for regression to grasps. However, grasp detection is a much simpler task than object detection. MultiGrasp only needs to predict a single graspable region for an image containing one object. It doesn’t have to estimate the size, location, or boundaries of the object or predict it’s class, only find a region suitable for grasping. YOLO predicts both bounding boxes and class probabilities for multiple objects of multiple classes in an image.

멀티그랩. 우리의 작업은 Redmon 등이 손아귀 감지 작업과 설계가 유사합니다 [27]. 경계 상자 예측에 대한 우리의 그리드 접근법은 회귀 분석을 위해 MultiGrasp 시스템을 기반으로 합니다. 그러나 그립 감지는 객체 감지보다 훨씬 간단한 작업입니다. MultiGrap은 하나의 객체가 포함된 이미지에 대해 하나의 움켜쥘 수 있는 영역만 예측하면 됩니다. 개체의 크기, 위치 또는 경계를 추정하거나 클래스의 값을 예측할 필요가 없으며, 움켜쥐기에 적합한 영역만 찾을 수 있습니다. YOLO는 이미지에서 여러 클래스의 여러 개체에 대한 경계 상자 및 클래스 확률을 모두 예측합니다.

1. **Experiments**

First we compare YOLO with other real-time detection systems on PASCAL VOC 2007. To understand the differences between YOLO and R-CNN variants we explore the errors on VOC 2007 made by YOLO and Fast R-CNN, one of the highest performing versions of R-CNN [14]. Based on the different error profiles we show that YOLO can be used to rescore Fast R-CNN detections and reduce the errors from background false positives, giving a significant performance boost. We also present VOC 2012 results and compare mAP to current state-of-the-art methods. Finally, we show that YOLO generalizes to new domains better than other detectors on two artwork datasets.

먼저 PASCAL VOC 2007의 다른 실시간 탐지 시스템과 YOLO를 비교합니다. YOLO와 R-CNN 변형 간의 차이를 이해하기 위해, 우리는 R-CNN의 최고 성능 버전 중 하나인 YOLO와 Fast R-CNN에 의해 만들어진 VOC 2007에 대한 오류를 조사합니다 [14]. 서로 다른 오류 프로파일을 기반으로 하여, YOLO를 사용하여 Fast R-CNN 탐지 결과를 복원하고 백그라운드 거짓 긍정에서 발생하는 오류를 줄일 수 있으며, 이로 인해 성능이 크게 향상될 수 있음을 알 수 있습니다. 또한 VOC 2012 결과를 제시하고 mAP를 현재 최신 방법과 비교합니다. 마지막으로, YOLO는 두 개의 아트워크 데이터셋에서 다른 검출기보다 새로운 도메인을 더 잘 일반화한다는 것을 보여줍니다.

* 1. **Comparison to Other Real-Time Systems**

Many research efforts in object detection focus on making standard detection pipelines fast. [5] [38] [31] [14] [17] [28] However, only Sadeghi et al. actually produce a detection system that runs in real-time (30 frames per second or better) [31]. We compare YOLO to their GPU implementation of DPM which runs either at 30Hz or 100Hz. While the other efforts don’t reach the real-time milestone we also compare their relative mAP and speed to examine the accuracy-performance tradeoffs available in object detection systems.

물체 탐지에 대한 많은 연구 활동은 표준 탐지 파이프라인을 빠르게 만드는 데 초점을 맞추고 있습니다. [5] [38] [31] [14] [17] [28] 그러나 Sadeghi 등만이 실제로 실시간(초당 30프레임 이상)으로 실행되는 검출 시스템을 생산합니다 [31]. 우리는 YOLO를 30Hz 또는 100Hz에서 실행되는 DPM의 GPU 구현과 비교합니다. 다른 작업은 실시간 이정표에 도달하지 못하지만, 상대적인 mAP와 속도를 비교하여 객체 감지 시스템에서 사용할 수 있는 정확도와 성능의 단점도 조사합니다.

Fast YOLO is the fastest object detection method on PASCAL; as far as we know, it is the fastest extant object detector. With 52.7% mAP, it is more than twice as accurate as prior work on real-time detection. YOLO pushes mAP to 63.4% while still maintaining real-time performance.

We also train YOLO using VGG-16. This model is more accurate but also significantly slower than YOLO. It is useful for comparison to other detection systems that rely on VGG-16 but since it is slower than real-time the rest of the paper focuses on our faster models.

Fastest DPM effectively speeds up DPM without sacrificing much mAP but it still misses real-time performance by a factor of 2 [38]. It also is limited by DPM’s relatively low accuracy on detection compared to neural network approaches.

R-CNN minus R replaces Selective Search with static bounding box proposals [20]. While it is much faster than R-CNN, it still falls short of real-time and takes a significant accuracy hit from not having good proposals.

Fast R-CNN speeds up the classification stage of R-CNN but it still relies on selective search which can take around 2 seconds per image to generate bounding box proposals. Thus it has high mAP but at 0.5 fps it is still far from realtime.

빠른 YOLO는 PASCAL에서 가장 빠른 물체 감지 방법이며 현존하는 물체 감지기 중 가장 빠른 것으로 알려져 있습니다. 52.7%의 mAP로 실시간 탐지에 대한 이전 작업보다 2배 이상 정확합니다. YOLO는 실시간 성능을 유지하면서 mAP를 63.4%로 끌어올립니다.

우리는 또한 VGG-16을 사용하여 YOLO를 훈련합니다. 이 모델은 YOLO보다 정확도가 높지만 속도가 상당히 느립니다. VGG-16에 의존하는 다른 탐지 시스템과 비교할 때 유용하지만, 실시간에 비해 속도가 느리기 때문에 나머지 문서는 더 빠른 모델에 초점을 맞춥니다.

가장 빠른 DPM은 많은 mAP를 그대로 유지하면서 DPM 속도를 효과적으로 높이지만 실시간 성능은 2배 이상 저하됩니다 [38]. 또한 DPM이 신경 네트워크 접근 방식에 비해 탐지 정확도가 상대적으로 낮기 때문에 제한됩니다.

R-CNN에서 R을 뺀 값이 선택적 검색을 정적 경계 상자 제안으로 대체합니다 [20]. R-CNN보다 속도가 훨씬 빠르지만 아직 실시간에는 미치지 못하고 제안도 제대로 되지 않아 적중률이 상당하다.

빠른 R-CNN은 R-CNN의 분류 단계를 가속화하지만, 여전히 경계 상자 제안을 생성하는 데 이미지당 약 2초가 걸릴 수 있는 선택적 검색에 의존합니다. mAP가 높지만 0.5fps로 실시간과는 거리가 멀다.

텍스트, 테이블이(가) 표시된 사진

자동 생성된 설명

Table 1: Real-Time Systems on PASCAL VOC 2007. Comparing the performance and speed of fast detectors. Fast YOLO is the fastest detector on record for PASCAL VOC detection and is still twice as accurate as any other real-time detector. YOLO is 10 mAP more accurate than the fast version while still well above real-time in speed.

표 1: PASCAL VOC 2007의 실시간 시스템. 고속 감지기의 성능과 속도 비교. Fast YOLO는 PASCAL VOC 검출에 대해 기록된 가장 빠른 검출기이며 여전히 다른 실시간 검출기보다 2배 정확합니다. YOLO는 빠른 버전보다 10mAP 더 정확하지만 여전히 실시간 속도를 훨씬 능가합니다.

The recent Faster R-CNN replaces selective search with a neural network to propose bounding boxes, similar to Szegedy et al. [8] In our tests, their most accurate model achieves 7 fps while a smaller, less accurate one runs at 18 fps. The VGG-16 version of Faster R-CNN is 10 mAP higher but is also 6 times slower than YOLO. The ZeilerFergus Faster R-CNN is only 2.5 times slower than YOLO but is also less accurate.

최근 Faster R-CNN은 Szegedy 등과 유사한 경계 상자를 제안하기 위해 선택적 검색을 신경 네트워크로 대체합니다. [8] 테스트에서 가장 정확한 모델은 7fps를 달성하고, 더 작고 덜 정확한 모델은 18fps에서 실행됩니다. Faster R-CNN의 VGG-16 버전은 10mAP 높지만 YOLO보다 6배 느립니다. Zeiler Fergus Faster R-CNN은 YOLO보다 2.5배 느릴 뿐 아니라 정확성도 떨어집니다.

* 1. **VOC 2007 Error Analysis**

To further examine the differences between YOLO and state-of-the-art detectors, we look at a detailed breakdown of results on VOC 2007. We compare YOLO to Fast RCNN since Fast R-CNN is one of the highest performing detectors on PASCAL and it’s detections are publicly available.

We use the methodology and tools of Hoiem et al. [19] For each category at test time we look at the top N predictions for that category. Each prediction is either correct or it is classified based on the type of error:

• Correct: correct class and IOU > .5

• Localization: correct class, .1 < IOU < .5

• Similar: class is similar, IOU > .1

• Other: class is wrong, IOU > .1

• Background: IOU < .1 for any object

YOLO와 첨단 검출기 간의 차이를 추가로 검토하기 위해 VOC 2007에 대한 결과의 상세한 분석을 살펴본다. PASCAL에서 Fast R-CNN이 최고 성능의 검출기 중 하나이며 탐지 기능을 공개적으로 사용할 수 있기 때문에 YOLO를 Fast RCN과 비교합니다.

우리는 Hoem 등의 방법과 도구를 사용합니다. [19] 각 범주의 테스트 시간에 대해 해당 범주의 상위 N개 예측값을 살펴봅니다. 각 예측은 정확하거나 오차 유형에 따라 분류됩니다.

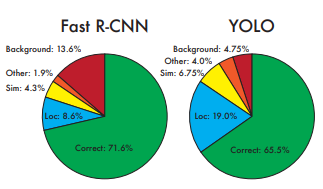


Figure 4: Error Analysis: Fast R-CNN vs. YOLO These charts show the percentage of localization and background errors in the top N detections for various categories (N = # objects in that category).

그림 4: 오류 분석: Fast R-CNN 대 YOLO 이 차트는 다양한 범주(N = 해당 범주의 개체 #개)에 대한 상위 N개 탐지의 현지화 및 배경 오류 비율을 보여줍니다.

Figure 4 shows the breakdown of each error type averaged across all 20 classes. YOLO struggles to localize objects correctly. Localization errors account for more of YOLO’s errors than all other sources combined. Fast R-CNN makes much fewer localization errors but far more background errors. 13.6% of it’s top detections are false positives that don’t contain any objects. Fast R-CNN is almost 3x more likely to predict background detections than YOLO.

그림 4는 전체 20개 클래스에서 평균화된 각 오류 유형에 대한 분석을 보여줍니다. YOLO는 객체를 올바르게 위치시키기 위해 애씁니다. 로컬라이제이션 오류는 다른 모든 소스를 합친 것보다 YOLO의 오류를 더 많이 설명합니다. 빠른 R-CNN은 로컬라이제이션 오류는 훨씬 적지만 백그라운드 오류는 훨씬 더 많습니다. 최상위 탐지 중 13.6%는 개체를 포함하지 않는 잘못된 긍정입니다. 고속 R-CNN은 YOLO보다 배경 탐지를 예측할 가능성이 거의 3배 높다.

**4.3. Combining Fast R-CNN and YOLO**

YOLO makes far fewer background mistakes than Fast R-CNN. By using YOLO to eliminate background detections from Fast R-CNN we get a significant boost in performance. For every bounding box that R-CNN predicts we check to see if YOLO predicts a similar box. If it does, we give that prediction a boost based on the probability predicted by YOLO and the overlap between the two boxes.

YOLO는 Fast R-CNN에 비해 백그라운드 실수가 훨씬 적습니다. Fast R-CNN에서 백그라운드 탐지를 제거하기 위해 YOLO를 사용하면 성능이 크게 향상됩니다. R-CNN이 예측한 모든 경계 상자에 대해 YOLO가 유사한 상자를 예측하는지 확인합니다. 만약 그렇다면, 우리는 YOLO가 예측한 확률과 두 상자 사이의 겹침에 근거하여 그 예측에 힘을 실어줍니다.

테이블이(가) 표시된 사진

자동 생성된 설명

Table 2: Model combination experiments on VOC 2007. We examine the effect of combining various models with the best version of Fast R-CNN. Other versions of Fast R-CNN provide only a small benefit while YOLO provides a significant performance boost.

표 2: VOC 2007에 대한 모델 조합 실험. 우리는 다양한 모델을 최고의 Fast R-CNN 버전과 조합할 때의 효과를 조사합니다. Fast R-CNN의 다른 버전은 작은 이점만 제공하는 반면 YOLO는 상당한 성능 향상을 제공합니다.

The best Fast R-CNN model achieves a mAP of 71.8% on the VOC 2007 test set. When combined with YOLO, its mAP increases by 3.2% to 75.0%. We also tried combining the top Fast R-CNN model with several other versions of Fast R-CNN. Those ensembles produced small increases in mAP between .3 and .6%, see Table 2 for details.

최상의 Fast R-CNN 모델은 VOC 2007 테스트 세트에서 71.8%의 mAP를 달성합니다. YOLO와 결합하면 mAP가 3.2~75.0% 증가한다. 또한 상위 Fast R-CNN 모델과 다른 여러 버전의 Fast R-CNN을 결합해 보았습니다. 이러한 앙상블은 0.3 - 0.6% 사이에서 mAP가 소폭 증가했습니다. 자세한 내용은 표 2를 참조하십시오.

테이블이(가) 표시된 사진

자동 생성된 설명

Table 3: PASCAL VOC 2012 Leaderboard. YOLO compared with the full comp4 (outside data allowed) public leaderboard as of November 6th, 2015. Mean average precision and per-class average precision are shown for a variety of detection methods. YOLO is the only real-time detector. Fast R-CNN + YOLO is the forth highest scoring method, with a 2.3% boost over Fast R-CNN.

표 3: PASCAL VOC 2012 순위표. YOLO는 2015년 11월 6일 현재 전체 comp4(외부 데이터 허용) 공개 순위표와 비교했습니다. 다양한 탐지 방법에 대한 평균 평균 정밀도와 클래스별 평균 정밀도가 표시됩니다. YOLO는 유일한 실시간 탐지기입니다. Fast R-CNN + YOLO는 Fast R-CNN에 비해 2.3% 향상된 4번째로 높은 점수 방법입니다.

The boost from YOLO is not simply a byproduct of model ensembling since there is little benefit from combining different versions of Fast R-CNN. Rather, it is precisely because YOLO makes different kinds of mistakes at test time that it is so effective at boosting Fast R-CNN’s performance.

Unfortunately, this combination doesn’t benefit from the speed of YOLO since we run each model seperately and then combine the results. However, since YOLO is so fast it doesn’t add any significant computational time compared to Fast R-CNN.

Fast R-CNN의 여러 버전을 조합해 얻을 수 있는 이점이 적기 때문에 단순히 모델 앙상블의 부산물이 아니라 테스트 시 다른 종류의 실수를 하기 때문에 Fast R-CNN의 성능을 끌어올리는 데 효과적이다.

유감스럽게도 이 조합은 각 모델을 개별적으로 실행한 다음 결과를 결합하기 때문에 YOLO의 속도에 도움이 되지 않습니다. 그러나 YOLO는 속도가 빨라 Fast R-CNN에 비해 계산 시간이 크게 늘어나지 않는다.

* 1. **VOC 2012 Results**

On the VOC 2012 test set, YOLO scores 57.9% mAP. This is lower than the current state of the art, closer to the original R-CNN using VGG-16, see Table 3. Our system struggles with small objects compared to its closest competitors. On categories like bottle, sheep, and tv/monitor YOLO scores 8-10% lower than R-CNN or Feature Edit. However, on other categories like cat and train YOLO achieves higher performance.

Our combined Fast R-CNN + YOLO model is one of the highest performing detection methods. Fast R-CNN gets a 2.3% improvement from the combination with YOLO, boosting it 5 spots up on the public leaderboard.

VOC 2012 테스트 세트에서 YOLO는 57.9% mAP를 획득했습니다. 이는 현재 기술 상태보다 낮으며, VGG-16을 사용하는 원래 R-CNN에 더 가깝습니다(표 3 참조). 우리의 시스템은 가장 가까운 경쟁사들에 비해 작은 물체들과 씨름하고 있습니다. 병, 양 및 TV/모니터 YOLO와 같은 범주에서 R-CNN 또는 Feature Edit보다 8-10% 낮은 점수를 받았습니다. 그러나 고양이와 기차와 같은 다른 범주에서는 YOLO가 더 높은 성과를 달성합니다.

Fast R-CNN + YOLO 모델이 결합된 최고 성능 탐지 방법 중 하나입니다. Fast R-CNN은 YOLO와의 결합으로 2.3% 향상되어 공개 리더보드 5계단 상승했습니다.

* 1. **Generalizability: Person Detection in Artwork**

Academic datasets for object detection draw the training and testing data from the same distribution. In real-world applications it is hard to predict all possible use cases and the test data can diverge from what the system has seen before [3]. We compare YOLO to other detection systems on the Picasso Dataset [12] and the People-Art Dataset [3], two datasets for testing person detection on artwork.

객체 감지를 위한 학술 데이터셋은 동일한 분포에서 교육 및 테스트 데이터를 끌어옵니다. 실제 애플리케이션에서는 가능한 모든 사용 사례를 예측하는 것이 어렵고 테스트 데이터는 시스템이 이전에 확인한 것과 다를 수 있습니다 [3]. Picasso Dataset[12] 및 People-Art Dataset[3]의 다른 탐지 시스템과 YOLO를 비교하여 아트워크에서 사람 탐지 테스트를 수행합니다.

Figure 5 shows comparative performance between YOLO and other detection methods. For reference, we give VOC 2007 detection AP on person where all models are trained only on VOC 2007 data. On Picasso models are trained on VOC 2012 while on People-Art they are trained on VOC 2010.

R-CNN has high AP on VOC 2007. However, R-CNN drops off considerably when applied to artwork. R-CNN uses Selective Search for bounding box proposals which is tuned for natural images. The classifier step in R-CNN only sees small regions and needs good proposals.

DPM maintains its AP well when applied to artwork. Prior work theorizes that DPM performs well because it has strong spatial models of the shape and layout of objects. Though DPM doesn’t degrade as much as R-CNN, it starts from a lower AP.

그림 5는 YOLO와 다른 검출 방법의 성능을 비교한 것입니다. 참고로 모든 모델이 VOC 2007 데이터에 대해서만 교육되는 사람에 대한 VOC 2007 탐지 AP를 제공합니다. 피카소 모델은 VOC 2012에서 교육되고 피플-아트 모델은 VOC 2010에서 교육됩니다.

R-CNN은 VOC 2007에서 높은 AP를 가지고 있습니다. 하지만 예술작품에 적용하면 R-CNN이 상당히 떨어진다. R-CNN은 자연 이미지에 맞게 조정된 경계 상자 제안에 대한 선택적 검색을 사용합니다. R-CNN의 분류자 단계는 작은 영역만 보고 좋은 제안이 필요합니다.

DPM은 아트워크에 적용 시 AP를 잘 유지합니다. 이전 연구에서는 DPM이 개체의 모양과 레이아웃에 대한 강력한 공간 모델을 가지고 있기 때문에 성능이 우수하다고 이론화했습니다. DPM은 R-CNN만큼 성능이 저하되지 않지만 낮은 AP에서 시작합니다.

YOLO has good performance on VOC 2007 and its AP degrades less than other methods when applied to artwork. Like DPM, YOLO models the size and shape of objects, as well as relationships between objects and where objects commonly appear. Artwork and natural images are very different on a pixel level but they are similar in terms of the size and shape of objects, thus YOLO can still predict good bounding boxes and detections.

YOLO는 VOC 2007에서 성능이 우수하고 아트워크에 적용하면 AP가 다른 방식에 비해 저하가 적다. DPM과 마찬가지로 YOLO는 개체의 크기 및 모양뿐만 아니라 개체 간의 관계와 개체가 일반적으로 나타나는 위치를 모델링합니다. 아트워크와 자연이미지는 픽셀 단위로 차이가 많이 나지만 물체의 크기와 모양은 비슷하기 때문에 YOLO는 여전히 좋은 경계상자와 탐지를 예측할 수 있다.

1. **Real-Time Detection In The World**

YOLO is a fast, accurate object detector, making it ideal for computer vision applications. We connect YOLO to a webcam and verify that it maintains real-time performance, including the time to fetch images from the camera and display the detections.

The resulting system is interactive and engaging. While YOLO processes images individually, when attached to a webcam it functions like a tracking system, detecting objects as they move around and change in appearance. A demo of the system and the source code can be found on our project website: <http://pjreddie.com/yolo/>.

YOLO는 빠르고 정확한 물체 감지기로 컴퓨터 비전 응용 분야에 이상적입니다. YOLO를 웹캠에 연결하여 카메라에서 이미지를 가져와 탐지 결과를 표시하는 시간을 포함하여 실시간 성능을 유지하는지 확인합니다.

그 결과 시스템은 상호작용적이고 몰입적입니다. YOLO는 이미지를 개별적으로 처리하지만 웹캠에 부착하면 사물이 움직이면서 감지되고 모습이 변한다. 시스템과 소스 코드에 대한 데모는 프로젝트 웹사이트 http://pjreddie.com/yolo/에서 확인할 수 있습니다.

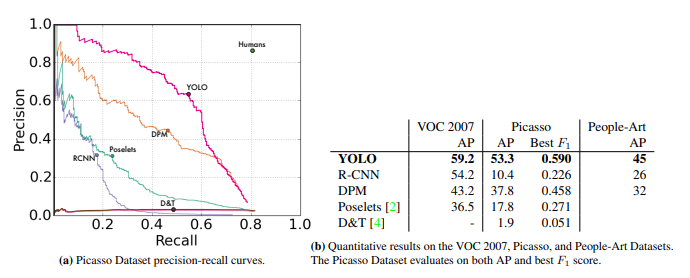


Figure 5: Generalization results on Picasso and People-Art datasets.

그림 5: Picasso 및 People-Art 데이터 세트에 대한 일반화 결과.

텍스트, 다른, 다양한, 같은이(가) 표시된 사진

자동 생성된 설명

Figure 6: Qualitative Results. YOLO running on sample artwork and natural images from the internet. It is mostly accurate although it does think one person is an airplane.

그림 6: 정성적 결과. 샘플 아트웍과 인터넷의 자연스러운 이미지에서 실행되는 YOLO. 한 사람이 비행기라고 생각하지만 대부분 정확합니다.

1. **Conclusion**

We introduce YOLO, a unified model for object detection. Our model is simple to construct and can be trained directly on full images. Unlike classifier-based approaches, YOLO is trained on a loss function that directly corresponds to detection performance and the entire model is trained jointly.

Fast YOLO is the fastest general-purpose object detector in the literature and YOLO pushes the state-of-the-art in real-time object detection. YOLO also generalizes well to new domains making it ideal for applications that rely on fast, robust object detection.

Acknowledgements: This work is partially supported by ONR N00014-13-1-0720, NSF IIS-1338054, and The Allen Distinguished Investigator Award.

객체 감지를 위한 통합 모델인 YOLO를 소개합니다. 우리의 모델은 구성이 간단하며 전체 이미지에 대해 직접 교육할 수 있습니다. 분류기 기반 접근법과 달리 YOLO는 검출 성능에 직접 대응하는 손실 함수에 대해 훈련하고 전체 모델을 공동으로 훈련한다.

Fast YOLO는 문헌에서 가장 빠른 범용 객체 검출기이며, YOLO는 실시간 객체 검출에서 최첨단 기술을 제공합니다. 또한 YOLO는 새로운 도메인을 잘 일반화하므로 빠르고 강력한 객체 감지에 의존하는 애플리케이션에 이상적입니다.

승인: 이 작업은 ONR N00014-13-1-0720, NSF IIS-1338054 및 The Allen Distributed Investigator Award에서 부분적으로 지원됩니다.