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

Abstract

Object detection의 새로운 접근법 YOLO를 소개해보겠습니다. 기존에는 Object detection을 위해 classifier를 용도를 변경해 사용하는 선행작업을 거쳤다. 대신에 우리(YOLO)는 object detection을 공간적으로 분리된 bounding box들과, class probabilities에 관한 회귀문제로 변경했다. 하나의 신경망은 한번에 bounding box와 class probabilities를 예측하게 된다. 전체 pipline이 1개의 신경망이기 때문에 YOLO는 종단간(end-to-end)으로 detection performance가 최적화가 가능하다.

YOLO는 엄청 빠르다. YOLO는 초당 45개의 frame을 fast YOLO는 155개의 frame을 처리할 수 있지만, 다른 실시간 감지기(real-time detector)보다 2배의 mAP(평가지표?)를 달성했다. 최첨단 detection system들에 비해서 YOLO는 localization error는 많지만 배경에 대해 false positive를 범하기는 어렵다. 마지막으로 YOLO는 굉장히 general한 representation을 학습한다. YOLO는 다른 (DPM, R-CNN 등의) detection method들에 비해 자연이미지에서 다른 예술작품에 이르는 다양한 domain에서 generalizing 성능이 뛰어나다.

1. Introduction

사람은 image를 힐끗 보는 것만으로도 즉각적으로 object가 image의 어디에 있는지 무엇과 interact 하고 있는지 알 수 있다. 사람의 시각 system은 빠르고 정확하다. 그리고 우리에게 특별한 conscious없이 운전하는 어려운 task를 수행할 수 있도록한다. object detection에서의 빠르고 정확한 algorithm은 특별한 sensor나 부가적인 device 없이 차를 운전할 수 있도록 하고, 보조 장치가 사람에 실시간 장면 정보를 전달해주며, general한 반응형 로봇의 잠재성을 열어줍니다.

최근의 detection system은 classifier를 detection으로 용도를 변경합니다. object를 detect하기 위해서 최근의 detection system은 object에 classifier를 사용하거나 다양한 위치나 크기를 test image에 적용해 evaluate했습니다. DPM(deformable parts models)와 같은 system은 sliding window 방식으로 전체 image에 대해 동일하게 classifier가 적용하는 방법으로 접근했습니다.

더 최근의 R-CNN에서의 image상에서 처음 potential bounding box를 만들어내기 위한 region proposal method 를 실행하고 각각 bounding box에 대해 classifier를 실행한다. classification이후 post-processing은 bounding box를 정제하기 위해, 중복된 detection를 제거하기 위해, 다른 box들을 고려해 rescoring을 한다. 이러한 복잡한 pipeline들은 각각의 개별적인 component들이 개별적으로 학습이 되어야 하기 때문에 느리고 최적화하기 어렵다.

우리는 image pixel에서 bounding box 좌표까지 그리고 class probabilites 를 object detection으로 생각해 단일 회귀 문제로 생각했습니다. 우리의 system을 사용하면 image를 1번만 보더라도 object가 더이에 있는지 무엇인지를 알 수 있다.

YOLO는 굉장히 간단하다. Figure 1을 참고하면, 하나의 convolution network가 다양한 bounding box를 예측하고 각각의 상자에 대한 class probabilities를 동시에 예측한다. YOLO는 전체 images에 대해 학습하고, detection performance에 대해 최적화한다. 이 통합된 model은 object detection에서 전통적인 방법에 비해 몇가지의 이점이 있다.

첫번째로 YOLO는 굉장히 빠릅니다. 우리는 detection을 regression 문제로 바꿨기 때문에, 우리는 복잡한 pipline이 필요가 없습니다. predict detection을 test할 때 새로운 image를 neural network에서 실행하기만 하면 된다. (대충 titanX에서 45, 150 프레임 정도 나와서 실생활에서 써도 될 것 같다는 얘기). YOLO는 다른 real-time system에 비해서 2배나 높은 평균 정확도를 가졌습니다.

두번째로 YOLO는 object에 대한 추론을 global한 영역의 image에서 추론합니다. sliding window기법이나 region proposal를 기반으로 한 테크닉과 다르게 YOLO는 training과 test시에 전체 image를 봅니다. 그래서 생김새 뿐만 아니라 contextual informataion또한 encoding합니다. Fast R-CNN 같은 경우 큰 영역을 볼 수 없기 때문에 배경 부분을 물체로 인식할 수도 있습니다. YOLO는 이에 비해 배경에 대한 error가 Fast R-CNN에 비해 절반 정도이다.

세번째 YOLO는 object에 대한 generalizable representation을 학습합니다. 자연 image를 학습하고 artwork를 test할 때 YOLO는 DPM이나 R-CNN보다 더 월등한 성능을 냅니다. 왜냐하면 YOLO은 높은 generalizable하기 때문이다. 이것은 새로운 domain이나 예상치 못한 input에 대해서도 좋지 않은 성능을 내는 경우가 적어진다는 뜻입니다.

YOLO는 아직 최첨단 detection system에 비해서 정확도에서 떨어집니다. 빠르게 object를 인식하는 동안 YOLO는 어떤 object들의 위치를 예측하는 것을 어려워합니다. 특히 작은 물체에 대해서 더 그렇습니다. 우리는 이러한 tradeoff에 대해서 이후에 실험을 진행해보았습니다.

▼ Introduction 요약

YOLO는 single nerual network 구조이기 때문에 빠르다.

YOLO는 국지적인 요소만을 detecting 하는데 사용하는 것이 아닌 이미지 전체를 사용해 detection을 진행하게 되므로 background error가 적다.

YOLO는 새로운 domain이나 접해보지 못한 새로운 image에 대해서도 성능이 괜찮게 높습니다.

하지만 새로운 종횡비를 가지는 물체에 대해서는 조금 낮은 정확도를 가지고 있습니다.

하지만 최첨단 detection Model(sota)에 비해 정확도는 조금 떨어지는 경향이 있습니다.

2. Unified Detection

우리는 object detection의 분할된 components들을 single neural network로 합쳤습니다. 우리의 network는 전체 이미지로 부터 예측을 통해 나온 feature를 각각의 bounding box들을 예측하는 데에 사용합니다. feature를 bounding box의 class를 예측하는 데에도 동시에 사용합니다. 이것은 우리의 network가 전체 image와 image내의 모든 object에 대해서 gloabl하게 추론한다는 것을 의미합니다. YOLO는 end-to-end로 training할 수 있고, 높은 평균 정확도를 가지면서 realtime speed로 빠르도록 설계했습니다.

우리의 system은 input image를 SxS grid로 분할합니다. 만약 object의 중심이 grid cell의 중심에 있다면 grid cell은 해당 object를 detecting 합니다. 각 grid cell 은 B개의 bounding box를 예측하고, confidence socre를 각각의 bounding box에 대해 예측합니다. 이 confidence score는 이 model이 물체를 포함하는 예측을 얼마나 확신하는지, 박스에 대한 예측이 얼마나 정확할 지를 의미합니다. 우리는 confidence를 아래와 같이 정의합니다.

$$\Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}}$$

만약 object가 cell안에 없으면 confidence score는 0이됩니다.(수식에 따라) 그렇지 않으면 우리는 신뢰 점수가 bounding box와 같아지기를 원합니다.

▼ 위 내용 정리

confidence score는 말 그대로 cell 안에 잘 있는지 있다면 bounding box가 얼마나 잘 표현되었는지를 나타내는 score이다.

- cell 안에 잘 있는지

  class를 잘 분류 했는지를 묻는 항목으로 YOLO는 각 grid cell마다 C개의 class probability를 예측하게 되는데 이때 ground truth의 class와 현재 cell의 최대 class probability가 동일하면 Pr(Object) 식이 양수라 IOU를 곱해 값이 나오게 됨

니다. 만약 다른 class를 예측하게 되면 Pr(Object)가 0이 되어 전체 연산의 결과는 무조건 0이 나오게 됩니다.

- bounding box가 잘 표현되었는지
  - 이는 IOU 공식을 통해 나타내게 됩니다.

각각의 bounding box는 5개의 예측값을 가집니다(x, y, w, h, confidence). x,y는 box의 center의 좌표값을 의미하고 이것은 grid cell과 연관이 있습니다. w,h는 전체 image와 상대적(비례)입니다. 마지막으로 confidence는 예측 box와 실제 정답 사이의 IOU를 의미합니다.

각각의 grid cell은 C conditional class probablities, Pr(Class_i|Object)를 예측한다. 이러한 확률은 물체를 포함한 셀 격자에 한정된다. 우리는 B와 상관 없이 셀 격자마다 1개의 C class를 예측한다.

test시에 우리는 conditional class probabilities와 개별적인 box confidence prediction을 곱합니다.

$$\Pr(\text{Class}_i|\text{Object}) * \Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}} = \Pr(\text{Class}_i) * \text{IOU}_{\text{pred}}^{\text{truth}}$$

▼ 위 내용 정리

각 셀 격자는 (x, y, w, h, confidence) + C 크기의 vector를 가지게 되고,
이미지 전체는 (S*S*(5*B +C)) 크기의 tensor를 가지게 됩니다.

5*B는 각 grid cell은 B개의 bounding box를 예측하게 되고, 5는 (x, y, w, h, confidence)를 의미.

C는 class의 갯수로 grid cell에 포함된 물체가 각 class에 해당될 확률들의 벡터 크기, (물체1, 물체 2,..., 물체 C)라서 C개를 가지게 된다.

YOLO를 PASCAL VOC Dataset에 적용하면 S = 7, B = 2, C = 20으로 적용했으므로 model의 output은 7*7*(5*2 + 20) = 7*7*(30)이 나오게 된다.

▼ Unified Detection 요약

원래 class예측과 bounding box 예측이 각각 optimize 되는 방식에서 YOLO는 한번에 optimize할 수 있도록 single convolution network 구조로 만들어졌습니다.

전체 image를 SxS grid로 나눠서 각각의 cell에서 C개의 class를 예측하고 B개의 bounding box를 가지게 됨.

이때 B개의 각각의 confidence score를 예측하게 되고, 이때 confidence score는 class probabilities (Pr) 와 bounding box prediction의 정확도를 의미하는 IOU를 곱

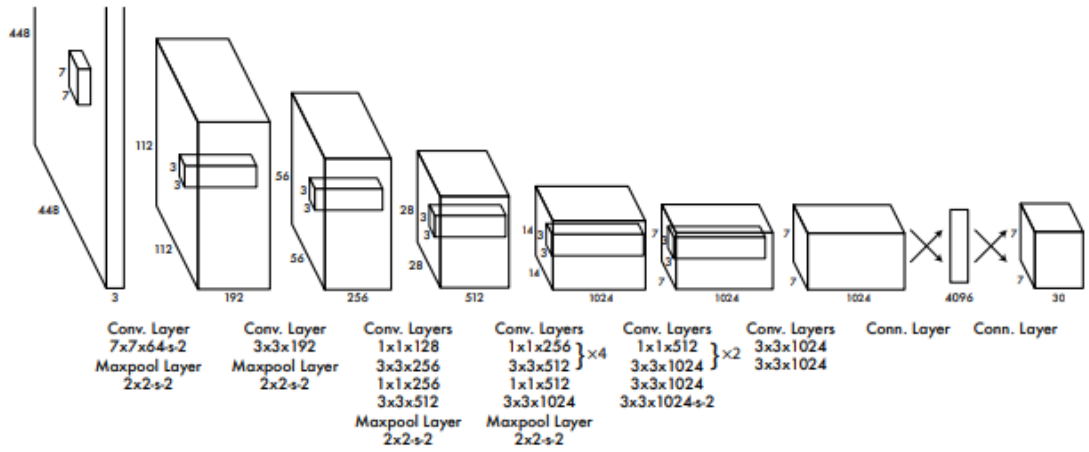
해서 나오게 된다.

최종 예측 출력 tensor의 dimension의 shape은 SxSx(5*B + C) 이다.

2.1 Network Design

우리는 이 모델을 convolutional nerual network로 구현했고, PASCAL VOC detection dataset으로 평가했다. 시작 convolutional layer는 image에서 feature를 추출하고, FC layer는 output의 확률과 좌표를 예측합니다. 우리의 network architecture는 image classification을 위해 GoogLeNet에 영감을 받았다. 우리의 network는 24개의 convolution layer와 2개의 FC layer로 구성되어 있다. GoogLeNet의 inception module 대신에 우리는 1x1 reduction layer와 3x3 convolution layer를 사용했다. 전체 네트워크는 Figure 3에 나와있다. 우리는 빠른 버전의 YOLO또한 설계했다. Fast YOLO는 더 적은 convolutional layer를 사용했고(9개) 더 적은 filter를 각 layer에 적용했다. 신경망의 크기를 제외한 다른 요소들은 동일하다.

- Figure 3



우리의 detection network는 24개의 convolutional layer를 가지고 있고, 2개의 FC layer를 가지고 있다. 선행 feature space를 줄이기 위해 1x1 convolution layer를 적용했고, ImageNet의 classification을 위해 convolutional layer를 입력크기 224x244의 크기를 가진 채 pretrained했다. 이후 다음부터는 448x448의 크기로 detection을 수행한다.

▼ Network Design 요약

GoogLeNet의 영감을 받아 network의 layer를 구성하였고 1x1 reduction layer와 3x3 convolution layer를 사용해 24개의 convolution layer와 2개의 FC layer로 구성되었다.

Fast YOLO는 정확도를 조금 포기했지만 더 빠른 속도를 가졌고, 9개의 convolution layer를 가졌고 나머지는 YOLO와 동일하다.

2.2 Training

우리는 YOLO의 convolutional layer를 pretrain하기 위해 ImageNet의 1000-class competition dataset을 사용했다. pretraining을 위해 우리는 FIgure 3의 첫 20개 convolution layer 와 average-pooling layer와 FC layer를 뒤따라 사용했다. 우리는 위 network를 1주일간 학습했고, top-5 acc가 88%를 달성했다. 우리는 Darknet의 framework를 사용해 training과 inference를 했습니다.

그런 다음 우리는 해당 model을 detection을 수행하도록 convert 했다. Ren et al은 pretrained model에 convolutional layer와 connected layer를 추가하는 것으로 performance를 향상 시킬 수 있음을 보였다. 위를 토대로 우리는 weight가 random initialize 된 4개의 conovolutional layer와 2개의 fully connected layer를 추가했다. detection은 종종 세밀한 visual 정보가 필요하므로 input size를 224x224에서 448x448로 올렸다.

우리의 최종 layer는 class probabilities와 bounding box coordinate를 예측합니다. 우리는 bounding box의 width와 height를 normalize해서 0~1사이의 값을 가지게 한다. 우리는 bounding box의 x,y 좌표를 offset으로 변환해 0~1사이의 값이 나오도록 또한 변환합니다.

우리는 activation function으로 Leaky ReLU를 사용했습니다.

우리는 sum-squared error를 output에 적용해 optimize 했습니다. 우리가 SSE를 사용한 이유는optimize하기 쉽기 때문입니다. 그러나 SSE를 사용하는 것은 우리의 목표인 평균 정확도를 최대화할 수는 없습니다. 현재 localization error와 classification error의 비중을 같게 뒀지만 이는 적합하지 않을 수 있다. 또한 많은 grid cell은 물체를 포함하고 있지 않으며, 이로 인해 각 grid cell의 confidence score는 0이 된다. 이것은 종종 물체를 포함한 셀의 gradient를 발산하게 만들었다.

이것을 해결하기 위해 우리는 bounding box 좌표 예측에 관한 loss 값을 증가시키고, obejct를 포함하고 있지 않은 box에 대한 confidence 예측 loss를 줄이는 방법을 채택했다. 우리는 $\lambda coord$와 $\lambda noobj$라는 매개변수를 사용했고, 각각 값을 $\lambda coord = 5$, $\lambda noobj = 0.5$로 지정했다.

SSE는 큰 bounding box와 작은 bounding box에 대해서 같은 가중치 error를 가진다. 우리의 오류 metric은 큰 상자에서의 작은 error가 작은 상자에서의 같은 크기의 error 보다 덜 중요하다는 것을 반영해햐합니다. 이 문제를 반영하기 위해 우리는 width와 height에 제곱근을 씌워서 예측을 하게 됩니다.

YOLO는 각각의 grid cell에서 다양한 bounding box를 예측하게 된다. training 시에 우리는 1개의 물체에 대해 1개의 bounding box predictor를 원한다. 우리는 가장 높은 IOU값을 갖게 예측하는 "responsible"한 예측을 하는 predictor를 할당한다. 이로인해 bounding box predictor 사이에서 specialization이 발생한다. 각각의 predictor는 특정 크기, 종 횡비, 혹은 class를 더 잘 예측할 수 있고, 이는 전체적인 recall을 상승시킨다.

학습을 통해 우리는 multi-part loss function을 optimize한다.

$$
\begin{aligned}
&\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2 \qquad (3)
\end{aligned}
$$

▼ 위 수식 정리

multi-loss function으로 모든 bounding box에 대해 얼마나 오차가 큰지 계산한다.

첫번째 term은 x,y 좌표값에 대한 loss를 계산하고
두번째 term은 w,h의 크기에 관한 loss를 계산하고
세번째 term은 class (confidence)에 관한 loss를 계산하고
네번째항은 term은 object가 없는(?) 것에 관한 loss를 계산하고
다섯번째 항은 간단한 classifiaction loss이다.(?)

손실함수가 객체가 해당 그리드 셀에 있는 경우에만 loss 를 부여합니다. predictor가 responsible할 경우에만 coordinate error를 부여합니다.

우리는 network를 135 epoch동안 학습했고, validation 도 PASCAL VOC 2007과 2012를 사용해 진행했다. 우리가 2012를 test할 때 우리는 2007또한 포함시켰다. training하는 동안 우리는 batchsize = 64, momentum=0.9, decay=-0.00005으로 학습시켰다.

우리의 learning rate schedule은 아래와 같다. 첫 epoch시에는 0.001에서 0.01로 천천히 증가시켰다. 만약 너무 높은 learning rate로 시작했다면 우리의 모델은 종종 unstable gradient에 의해 loss가 diverge했을 것이다. leraning rate 0.01로 75 epoch를 실행한뒤 0.001로 30epoch를 진행하고 마지막 30 epoch를 0.0001로 진행한다.

Overfitting을 피하기 위해 우리는 dropout과 extensive data augmentation을 진행했다. layer 사이에 co-adaptation을 방지 하기 위해 첫번째 FC layer이후 dropout=0.5로 지정했다. Data augmentation을 위해 우리는 random scaling과 원래 이미지의 20%정도의 translation을 적용했다. 우리는 또한 HSV 색상 space에서 random한 exposure(빛 노출)과 1.5의 saturation(채도)을 적용했습니다.

▼ Training 요약

Ren et al의 의견에 따라 추가적인 convolution layer와 FC layer를 추가해 성능을 높였다.

Bounding box를 나타내는 x,y,w,h를 normalize했다.

activation function으로 Leaky ReLU를 사용했다.

SSE로 optimize를 진행했을 때 종종 diverse하는 경향이 있어서 이를 해결하기 위해 bounding box에 대한 예측 loss를 증가시키고 class 예측에 대한 loss를 줄였다. (5:0.5)

→ w, h에 sqaure root를 쓴 이유

135 epoch만큼 학습하는데 learning rate schedule은
1~75 epoch에는 0.001~0.01로 점차 증가.
76~105 epoch에는 0.001
106~135 epoch에는 0.0001로 진행함.

co-adaptation을 막기 위해 droput을 진행함.

Data augmentation을 적용해 training함.

2.3 Inference

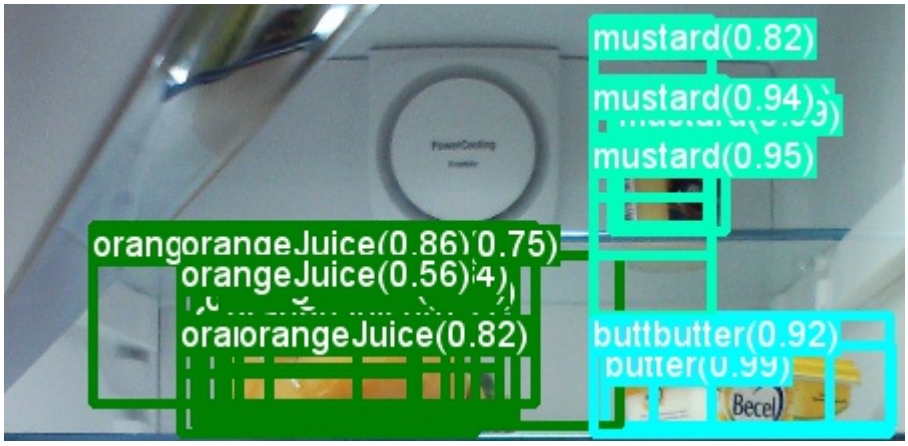
training할 때 처럼 predicting detection을 하는 것은 한 번의 network evaluation이면 된다. PASCAL VOC에서는 image 1개당 우리는 98개의 bounding box를 그리고 class probabilities를 예측했다. YOLO는 test time에서 굉장히 빠르다. 왜냐하면 YOLO는 classifier-based network와 다르게 single network evaluation만 필요하기 때문이다.

grid design은 bounding box preidction에서 공간적인 다양성을 enforce(야기)합니다. 가끔 object가 어느 셀 격자 안에 있는지 명확하고 한개의 object에 대해 1개의 box가 예상되는 경우가 있습니다. 그러나 어떤 큰 object나 여러 셀의 경계에 존재하는 object인 경우 물체의 위치는 복수개의 cell들에 의해 위치가 정해질 수도 있습니다. 이런 multiple detection

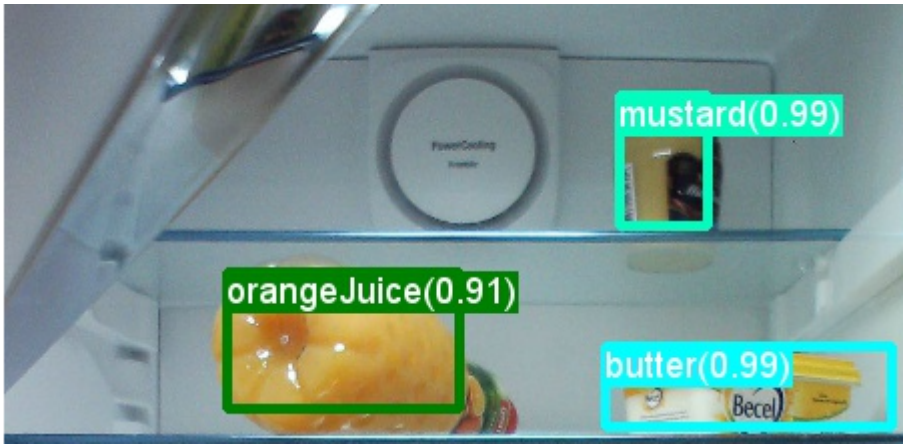
을 수정하는데 non-maximal suppression(NMS)가 사용될 수도 있습니다. 이는 mAP값을 2~3%정도 향상하지만 R-CNN이나 DPM에서 보여준 성능 향상보다는 저조합니다.

▼ non-maximal suppression(NMS)

- 적용 전



- 적용 후



▼ Inference 요약

SxSxB 개의 bounding box를 예측하게 되므로 다양하게 bounding box를 예측하게 됩니다.

다양한 cell에서 같은 물체가 detection 될 수 있습니다. 이런 multiple detection을 수정하는데에 NMS가 사용해 정확도를 조금 올릴 수 있습니다.

2.4 Limitation of YOLO

YOLO에서 각 grid cell은 2개의 box를 예측하고 하나의 class만을 가질 수 있기 때문에 bounding box를 예측할 때 강한 공간제약을 야기합니다. 이 spatial constraint(제약)는 예측할 수 있는 주변의 물체의 수를 제한합니다. 그래서 세때와 같은 작은 문제들이 그룹을 지어 다닐 경우 우리 model은 struggle(제대로 동작하지 않는다.)한다.

우리의 모델이 bounding box를 data를 통해 배우기 때문에 새로운 aspect ration나 configuration을 만나면 generalize 하기 어렵다. 우리의 model은 비교적 거친(조잡한) feature를 bounding box를 예측하는데 사용된다. 왜냐하면 우리 architecture가 input image로 부터 multiple downsampling layer를 거치기 때문이다.

마지막으로 approximate detection performance를 나타내는 loss function으로 학습을 하는 동안 우리의 loss function은 작은 bounding box와 큰 bounding 박스의 error를 같게 본다. 큰 상자의 작은 error는 비교적 느슨하게 생각할 수 있지만, 작은 상자에서의 작은 error은 IOU때문에 더 큰 효과가 발생되게 된다. 작은 상자에서의 error의 주된 이유는 정확하지 않은 localization(위치 값)이다.

▼ Limitation of YOLO 요약

각 grid cell은 B개의 bounding box밖에 가질 수 없기 때문에 세때와 object들을 struggle된다.

bounding box를 data를 보고 배우기 때문에 새로운 종횡비를 가진 object나 새로운 구성을 만나게 되면 generalize하기 어렵습니다.

큰 bounding box와 작은 bounding box에 대한 동일한 loss 가중치를 가지게 된다.

3. Comparison to Other Detection Systems

Object detection은 computer vision의 중요한 문제이다. Detection pipline은 일반적으로 input image에서 강력한 feature 를 뽑는 것으로 시작한다. 그러면 classifiers나 localizer는 feature space에서 물체를 인식합니다. 이들은 전체 image를 sliding window 방식이나 이미지의 일정 부분을 보는 방식으로 작동합니다. 우리는 YOLO와 여러가지 가장 좋은 detection framework와 장점과 단점을 비교했습니다.

Deformable parts models.

DPM은 sliding window 방식으로 object를 detection합니다. DPM은 서로 다른 pipline을 사용해 static feature를 뽑아내고, region을 classify하고, 높은 점수를 얻기 위한 bounding box 또한 예측합니다. 우리 system은 여러가지 이질적인 영역을 single convolutional nerual network로 대체되었습니다. YOLO의 network는 feature extract, bounding bx predcition, non-maximal suppression, 문맥 파악을 동시에 수행한다. static feature 대신

에 network는 detection task에 대해 in-line으로 feature를 학습하고 최적화한다. YOLO의 통합된 architecture는 DMP보다 더 빠르고 더 정학하다.

R-CNN. R-CNN은 sliding window 방식이 아닌 region proposal을 사용한다. Selective Search가 potential 한 bounding box를 만들고, convoltuional network가 feature extract를 수행하며, SVM은 박스를 evaluation하고, linear modeldms bounding box를 조절하며, non-maxlimal suppression은 중복되는 box를 삭제한다. 이 복잡한 pipline은 각각 개별 조절 되어야 하므로 굉장히 느리다.

YOLO는 R=CNN과 공통좀이 있습니다. SxS grid cell의 각각의 cell에서 potential bounding box들을 제안하고, convolutional feature로 평가합니다. 하지만 YOLO는 같은 물체를 중복해서 detection하는 것을 방지하기 위해 grid cell proposal을 공간적으로 제한합니다. 우리의 시스템은 98개의 bounding box를 제안하지만 Selective search를 사용한 R-CNN은 2000개의 bounding box를 제안했습니다. 마지막으로 YOLO는 모든 개별적인 요소를 합쳐 하나의 최적화된 model로 만들었습니다.

Other Fast Detector

Fast, Faster R-CNN은 computation을 공유하고, selective search 대신에 nerual netowrk를 regions propsal을 사용하면서 속도를 높이는데 중점을 두었습니다. R-CNN보다 정확도가 향상되었지만 아직 real-time에 적용하기에는 performance가 부족합니다.

Deep MultiBox.

단일 object를 감지하는데 full image를 고려해서 detection합니다.

OverFeat.

MultiGrasp.

4. Experiments

먼저 YOLO를 다른 real-time object detection model과 비교해보겟습니다.

4.1. Comparison to Other Real-Time Systems

Fast YOLO는 PASCAL datset을 기준으로 가장 빠른 object detection model입니다. mAP는 52.7%이고 같은 frame의 DPM보다 2배 정도 높은 정확도를 가졌습니다. YOLO는 real-time 성능을 그대로 유지하되 mAP를 63.4%까지 높인 모델입니다.

VGG-16을 사용해서 YOLO를 학습시켜보았습니다. 이 model은 더 정확하지만 기존의 YOLO보다 느려서 real-time 의 성능을 내기에는 어려웠습니다. 이 논문에서는 속도가 빠른 YOLO에 집중해 설명합니다.

Fastest DPM은 mAP를 살짝 하락시키면서 속도를 향상시킨 모델입니다. 하지만 real-time detection에 사용하기에는 여전히 느립니다. 또한 nerual network를 사용한 object detection model에 비해 accuracy도 떨어집니다.

R-CNN miuns R 모델은 selective Search 대신 static bounding box proposal를 사용했습니다. R-CNN에 비해서는 더 빠르지만 real-time detection에 사용하기에는 역시 부족합니다. 초당 0.5 frame을 처리할 수 있습니다.

Real-Time Detectors	Train	mAP	FPS
100Hz DPM [31]	2007	16.0	100
30Hz DPM [31]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [38]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[28]	2007+2012	73.2	7
Faster R-CNN ZF [28]	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21

위 표의 각종 object detection 정확도(mAP)와 속도(FPS)를 보여줍니다. FPS가 적어도 30은되어야지 real-time detection에 사용할 수 있을 듯 합니다. 이때 vgg-16 YOLO나 Fast R-CNN같은 ACC가 높은 model 같은 경우에도 FPS가 너무 낮아 real-time detection에는 부적합한 것을 알 수 있습니다.

▼ Comparison to Other Real-Time Systems 요약

YOLO는 기존 모델에 비해 속도는 월등히 빠르고, 정확도도 꽤 높은 수준이다.

4. 2. VOC 2007 Error Analysis

YOLO와 기존 객체 검출 모델을 더 비교해보겠습니다. 먼저 파스칼 VOC 2007 데이터 셋에 대해 YOLO와 Fast R-CNN의 성능을 비교해보겠습니다.

아래와 기준으로 category를 나눠서 생각해보겠습니다.

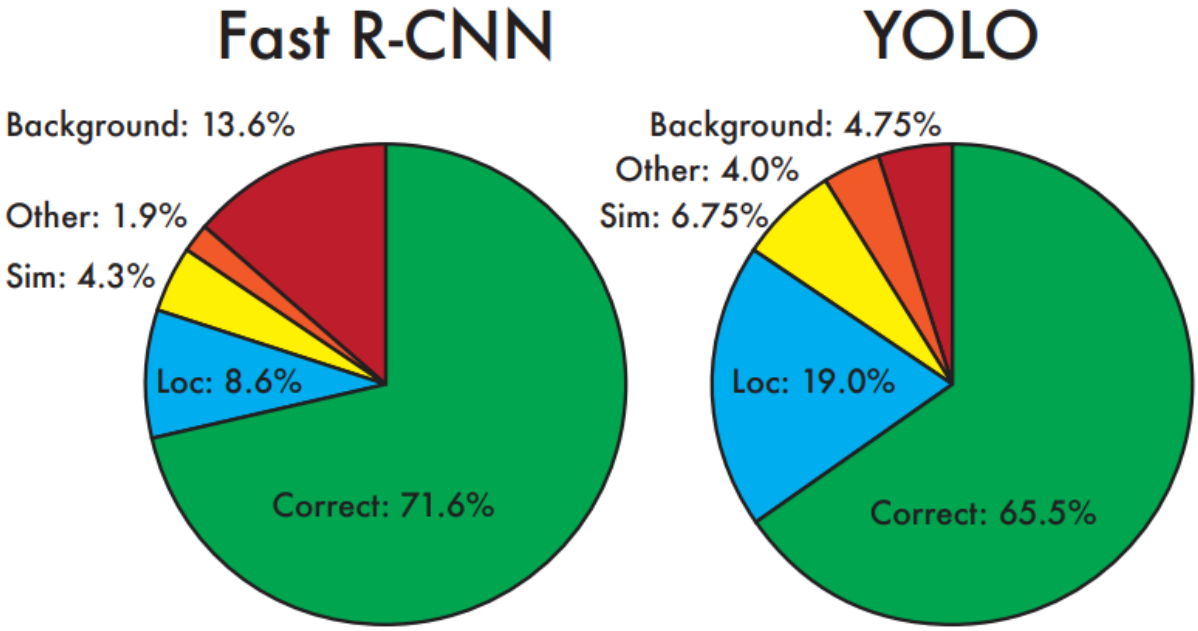
Correct : class가 정확하며 IOU > 0.5 인 경우

Localization : class가 정확하고, 0.1 < IOU < 0.5 인 경우

Similar : class가 유사하고 IOU > 0.1 인 경우

Other : class는 틀렸으나, IOU > 0.1 인 경우

Background : 어떤 Object라도 IOU < 0.1 인 경우



YOLO는 localization error가 상대적으로 Fast R-CNN에 비해 큽니다. 반면 background error가 굉장히 작습니다. background error는 배경에 아무 물체가 없는데 물체가 있다고 판단하는 false positive error입니다.

4. 3. Combining Fast R-CNN and YOLO

YOLO는 Fast R-CNN에 비해 background error가 훨씬 적습니다. 따라서 두개를 적절히 결합한다면 굉장히 높은 성능을 낼 수 있을 것입니다. R-CNN이 예측하는 모든 bounding box에 대해 YOLO도 유 사하게 예측하는지를 체크하면 됩니다. 앙상블(Fact R-CNN, YOLO)을 통해 mAP 성능을 향상시킬 수 있습니다.

이때 YOLO가 Fast R-CNN에 비해 비약적으로 빠르므로 Fast R-CNN단독으로 쓰기 보다는 YOLO와 ensemble을 하는 것이 더 좋을 것 같다.

4. 4. VOC 2012 Results

VOC 2012 test	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
MR_CNN_MORE_DATA [11]	**73.9**	**85.5**	**82.9**	**76.6**	**57.8**	**62.7**	**79.4**	77.2	86.6	**55.0**	**79.1**	**62.2**	87.0	**83.4**	**84.7**	78.9	45.3	73.4	65.8	80.3	74.0
HyperNet_VGG	71.4	84.2	78.5	73.6	55.6	53.7	78.7	**79.8**	87.7	49.6	74.9	52.1	86.0	81.7	83.3	**81.8**	**48.6**	**73.5**	59.4	79.9	65.7
HyperNet_SP	71.3	84.1	78.3	73.3	55.5	53.6	78.6	79.6	87.5	49.5	74.9	52.1	85.6	81.6	83.2	81.6	48.4	73.2	59.3	79.7	65.6
Fast R-CNN + YOLO	70.7	83.4	78.5	73.5	55.8	43.4	79.1	73.1	**89.4**	49.4	75.5	57.0	**87.5**	80.9	81.0	74.7	41.8	71.5	68.5	**82.1**	67.2
MR_CNN_S_CNN [11]	70.7	85.0	79.6	71.5	55.3	57.7	76.0	73.9	84.6	50.5	74.3	61.7	85.5	79.9	81.7	76.4	41.0	69.0	61.2	77.7	72.1
Faster R-CNN [28]	70.4	84.9	79.8	74.3	53.9	49.8	77.5	75.9	88.5	45.6	77.1	55.3	86.9	81.7	80.9	79.6	40.1	72.6	60.9	81.2	61.5
DEEP_ENS_COCO	70.1	84.0	79.4	71.6	51.9	51.1	74.1	72.1	88.6	48.3	73.4	57.8	86.1	80.0	80.7	70.4	46.6	69.6	**68.8**	75.9	71.4
NoC [29]	68.8	82.8	79.0	71.6	52.3	53.7	74.1	69.0	84.9	46.9	74.3	53.1	85.0	81.3	79.5	72.2	38.9	72.4	59.5	76.7	68.1
Fast R-CNN [14]	68.4	82.3	78.4	70.8	52.3	38.7	77.8	71.6	89.3	44.2	73.0	55.0	**87.5**	80.5	80.8	72.0	35.1	68.3	65.7	80.4	64.2
UMICH_FGS_STRUCT	66.4	82.9	76.1	64.1	44.6	49.4	70.3	71.2	84.6	42.7	68.6	55.8	82.7	77.1	79.9	68.7	41.4	69.0	60.0	72.0	66.2
NUS_NIN_C2000 [7]	63.8	80.2	73.8	61.9	43.7	43.0	70.3	67.6	80.7	41.9	69.7	51.7	78.2	75.2	76.9	65.1	38.6	68.3	58.0	68.7	63.3
BabyLearning [7]	63.2	78.0	74.2	61.3	45.7	42.7	68.2	66.8	80.2	40.6	70.0	49.8	79.0	74.5	77.9	64.0	35.3	67.9	55.7	68.7	62.6
NUS_NIN	62.4	77.9	73.1	62.6	39.5	43.3	69.1	66.4	78.9	39.1	68.1	50.0	77.2	71.3	76.1	64.7	38.4	66.9	56.2	66.9	62.7
R-CNN VGG BB [13]	62.4	79.6	72.7	61.9	41.2	41.9	65.9	66.4	84.6	38.5	67.2	46.7	82.0	74.8	76.0	65.2	35.6	65.4	54.2	67.4	60.3
R-CNN VGG [13]	59.2	76.8	70.9	56.6	37.5	36.9	62.9	63.6	81.1	35.7	64.3	43.9	80.4	71.6	74.0	60.0	30.8	63.4	52.0	63.5	58.7
YOLO	57.9	77.0	67.2	57.7	38.3	22.7	68.3	55.9	81.4	36.2	60.8	48.5	77.2	72.3	71.3	63.5	28.9	52.2	54.8	73.9	50.8
Feature Edit [33]	56.3	74.6	69.1	54.4	39.1	33.1	65.2	62.7	69.7	30.8	56.0	44.6	70.0	64.4	71.1	60.2	33.3	61.3	46.4	61.7	57.8
R-CNN BB [13]	53.3	71.8	65.8	52.0	34.1	32.6	59.6	60.0	69.8	27.6	52.0	41.7	69.6	61.3	68.3	57.8	29.6	57.8	40.9	59.3	54.1
SDS [16]	50.7	69.7	58.4	48.5	28.3	28.8	61.3	57.5	70.8	24.1	50.7	35.9	64.9	59.1	65.8	57.1	26.0	58.8	38.6	58.9	50.7
R-CNN [13]	49.6	68.1	63.8	46.1	29.4	27.9	56.6	57.0	65.9	26.5	48.7	39.5	66.2	57.3	65.4	53.2	26.2	54.5	38.1	50.6	51.6

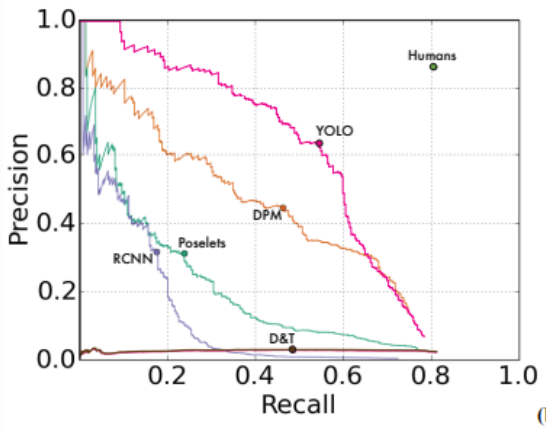
위 표에 따르면 VGG-16을 사용한 R-CNN과 YOLO의 mAP가 비슷합니다.
즉 속도 측면에서는 YOLO가 빠르고 정확도 측면에서는 Fast R-CNN과 YOLO를 결합한 모델이 좋다.

4. 5. Generalizability: Person Detection in Artwork

object detection 연구를 위해 사용하는 dataset은 training set과 test set이 동일한 distribution을 가지게 됩니다. 하지만 실제 이미지 데이터는 훈련 data set과 test set이 다를 수 있습니다.

여기서는 피카소 데이터 셋과 일반 예술 작품을 사용했습니다. 훈련 단계에서는 실제 이미지로 학습했지만 테스트 단계에서는 예술 작품을 활용해 테스트해보는 것입니다.

아래 표는 YOLO과 다른 객체 검출 모델의 성능을 측정한 것입니다. 파스칼 VOC 2007에서 학습한 YOLO, R-CNN, DPM 등의 성능을 서로 비교했습니다. R-CNN은 VOC 2007에서는 높은 정확도를 보이지만 예술작품에 대해서는 굉장히 낮은 정확도를 보입니다. DPM은 예술 작품에 대해서도 정확도가 크게 떨어지지는 않았습니다. 다만 VOC 2007에서의 정확도도 그리 높은 편은 아닙니다. 반면 YOLO는 VOC 2007에서도 가장 높은 정확도를 보였고, 예술 작품에 대해서도 정확도가 크게 떨어지지 않았습니다.



(a) Picasso Dataset precision-recall curves.

	VOC 2007	Picasso		People-Art
	AP	AP	Best F_1	AP
YOLO	**59.2**	**53.3**	**0.590**	**45**
R-CNN	54.2	10.4	0.226	26
DPM	43.2	37.8	0.458	32
Poselets [2]	36.5	17.8	0.271	
D&T [4]	-	1.9	0.051	

(b) Quantitative results on the VOC 2007, Picasso, and People-Art Datasets. The Picasso Dataset evaluates on both AP and best F_1 score.

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

▼ Generalizability: Person Detection in Artwork 요약

YOLO는 훈련 단계에서 접하지 못한 새로운 이미지도 잘 검출한다.

요약