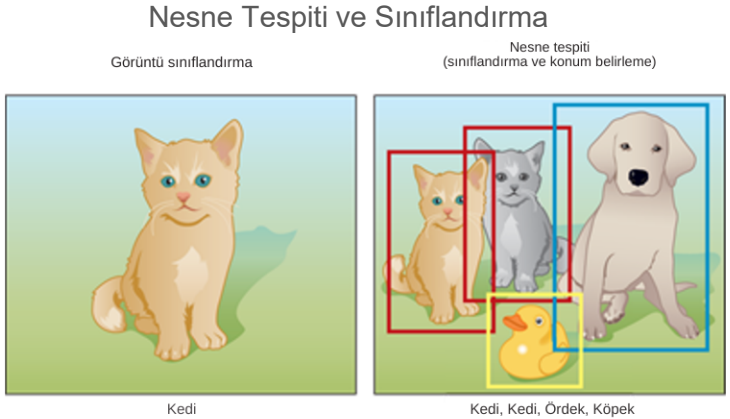
Yalnızca Bir Kez Bak (YOLO): Birleşik, Gerçek Zamanlı Nesne Tespiti

Source: https://www.cs.utexas.edu/~yukez/cs391r_fall2021/slides/pre_09-02_Shivang.pdf

Ele Alınan Problem: Nesne Tespiti

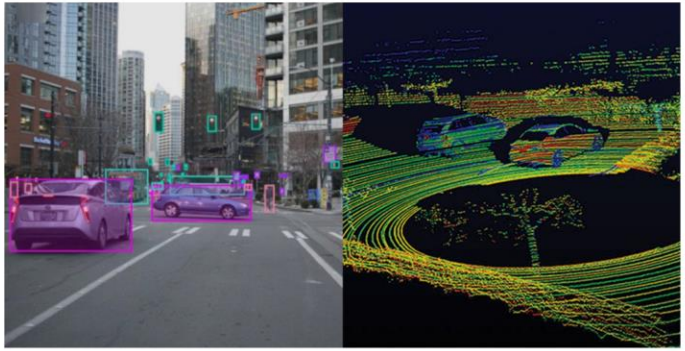
- Nesne tespiti, nesnelerin hem **konumunu belirleme** hem de **sınıflandırma** problemidir
- YOLO algoritmasının amacı, nesne tespitini hem **hızlı** hem de **yüksek doğrulukla** yapmaktır



Nesne Tespitinin Robotik İçin Önemi

- Görsel kip (modalite) çok güçlüdür
- İnsanlar yalnızca bu kipi kullanarak nesneleri gerçek zamanlı tespit edebilir ve algılayabilir (radara gerek duymadan)
- Gerçek zamanlı çalışan, hızlı tepki veren robot sistemleri istiyorsak (özel sensörler olmadan), neredeyse gerçek zamanlı görüntü tabanlı nesne tespiti büyük ölçüde yardımcı olabilir

Görüntü tabanlı ve LIDAR (otonom sürüş)



Önceki Nesne Tespiti Yaklaşımı

Bu işlem hattı, önceki SOTA nesne tespiti yöntemlerinin çoğunda kullanıldı:



Adım 1: Aday sınırlayıcı kutular üretmek için görüntüyü tara



Adım 2: Sınırlayıcı kutuyu bir sınıflandırıcıdan geçir

Adım 3: Son işleme uygula (gereksiz sınırlayıcı kutuları ele)

Temel Fikirler

Önceki Yaklaşımlar

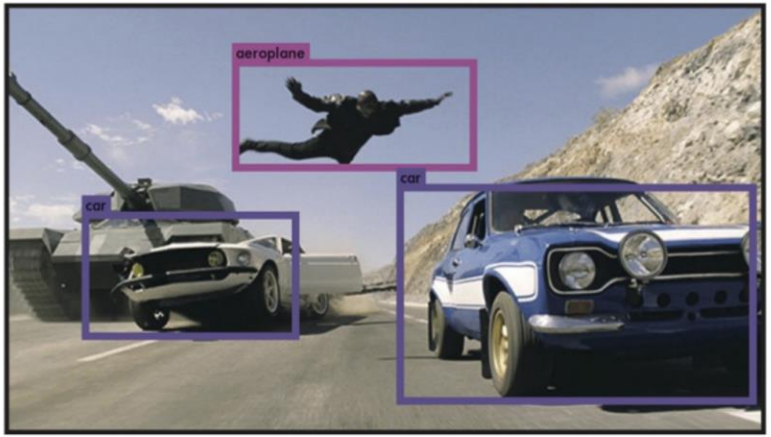
- Sınırlayıcı kutu üretimi ve sınıflandırma için ayrı modeller (daha karmaşık model hattı)
- Sınıflandırmayı birçok kez çalıştırmak gerekir (maliyetli hesaplama)
- Görüntünün yalnızca sınırlı bir kısmına bakar (tespit için bağlamsal bilgi eksik)

YOLO algoritması

- Konum belirleme ve sınıflandırma için tek bir sinir ağı (daha az karmaşık işlem hattı)
- Çıkarım yalnızca bir kez yapılır (verimli hesaplama)
- Her seferinde tüm görüntüye bakar, bu da daha az yanlış pozitife yol açar (tespit için bağlamsal bilgiye sahiptir)

Biçimsel Problem Tanımı

- Bir görüntü verildiğinde, görüntüdeki her tespit edilebilir nesne için bir sınırlayıcı kutu üret
- Her sınırlayıcı kutu için 5 tahmin üret: x, y, w, h, güven. Ayrıca sınıfı da üret



- x, y (sınırlayıcı kutunun merkez koordinatları)
- w, h (genişlik ve yükseklik)
- güven (sınırlayıcı kutunun nesne içerme olasılığı)
- sınıf (sınırlayıcı kutudaki nesnenin sınıfı)

İlgili Çalışmalar

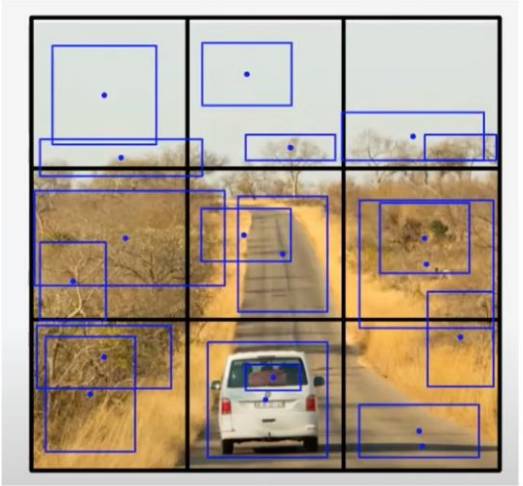
- R-CNN veya Bölge Tabanlı Evrişimli Ağ (Girshick vd. 2014):
  - Önceki kayan pencere yaklaşımını, adayları daha akıllıca seçen Seçici Arama (Selective Search) ile kullandı (yani daha az hesaplama)
  - Sınıflandırıcıya hâlâ görüntünün yalnızca sınırlı bir kısmını verir
  - Dezavantajlar: Büyük işlem hattı, yavaş, çok fazla yanlış pozitif
- Fast ve Faster R-CNN:
  - Daha önce açıklanan işlem hattının bölümlerini optimize eder
  - Dezavantajlar: doğruluk kaybı
- Deep Multibox (Szegedy vd. 2014):
  - İlgi alanlarını bulmak için bir CNN eğitir
  - Dezavantajlar: Sınıflandırmayı ele almaz, yalnızca konum belirleme yapar

İlgili Çalışmalar

- MultiGrasp (Redmon vd. 2014)
  - YOLO'ya benzer
  - Çok daha basit bir görev (birden çok nesne değil, yalnızca tek nesne tahmin etmesi gerekir)

YOLO’ya Genel Bakış

- Önce görüntü SxS boyutunda bir ızgaraya bölünür
- Her ızgara hücresi için B adet sınırlayıcı kutu üretilir
- Her sınırlayıcı kutu için 5 tahmin vardır: x, y, w, h, güven



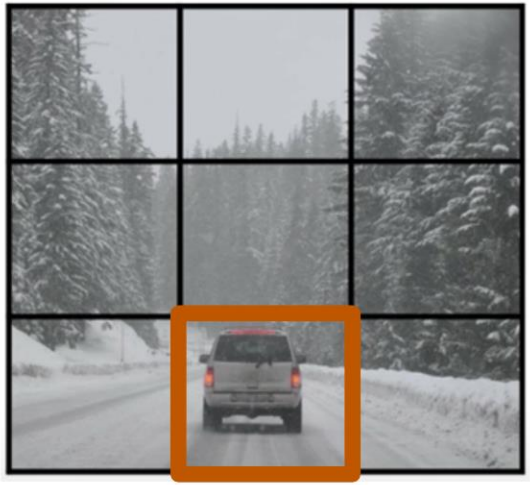
S = 3, B = 2

YOLO Eğitimi

- YOLO bir regresyon algoritmasıdır. X nedir? Y nedir?
- X basittir: görüntü genişliği (piksel) * yüksekliği (piksel) * RGB değerleri
- Y, S * S * (B * 5 + C) boyutunda bir tensördür
- B*5 + C terimi, tahminleri + bir ızgara hücresi için tahmin edilen sınıf dağılımını temsil eder

Her ızgara hücresi için bunun gibi bir vektörümüz var. Bu örnekte B = 2 ve C = 2

ρ_1
b_x_1
b_y_1
b_h_1
b_w_1
ρ_2
b_x_2
b_y_2
b_h_2
b_w_2
c_1
c_2

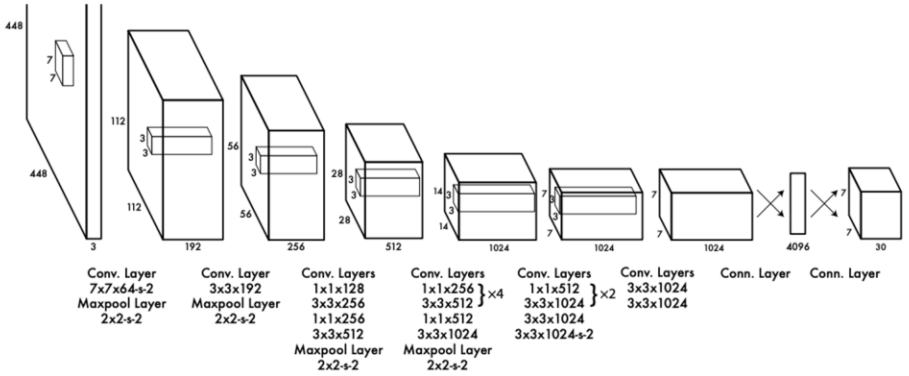


Gerçek etiket örneği:

1
b_x_1
b_y_1
b_h_1
b_w_1
0
?
?
?
?
c_1 = 1
c_2 = 0

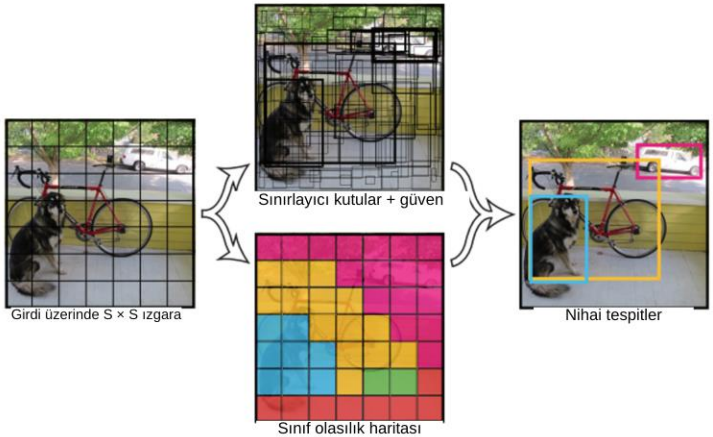
YOLO Mimarisi

- Girdi ve çıktıyı bildiğimize göre artık modeli ele alabiliriz
- Girdimiz 448 x 448 x 3 boyutundadır.
- Uygulamada 7 evrişim katmanı kullanılır
- Makaledeki parametreler: S = 7, B = 2, C = 20
- Çıktı: S*S*(5B+C) = 7*7*(5*2+20) = 7*7*30



YOLO ile Tahmin

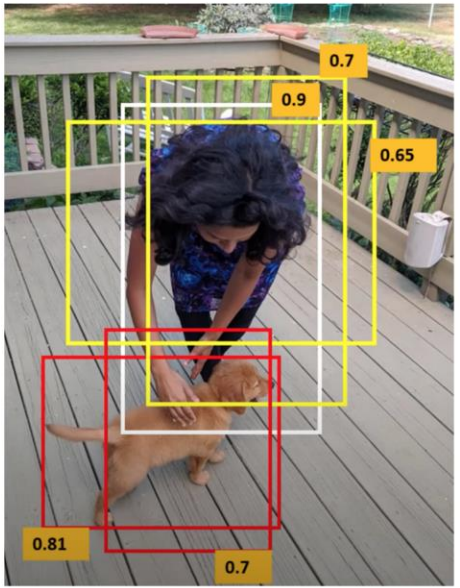
- Ardından çıktıyı kullanarak nihai tespitleri yaparız
- Düşük P(Nesne) değerine sahip sınırlayıcı kutuları elemek için bir eşik kullanılır
- Sınırlayıcı kutunun sınıfını bulmak için skor hesaplanır: kutunun merkezinin bulunduğu ızgara hücresinin Pr(Sınıf|Nesne) dağılımı üzerinden argmax alınır



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

Maksimum Olmayanların Bastırılması (NMS)

- Çoğu zaman nesneler tek bir ızgara hücresine düşer, ancak yine de gereksiz kutular oluşabilir (nadir durum; bunun için nesnenin birden çok hücreye yakın olması gerekir)
- Yüksek örtüşmeye sahip sınırlayıcı kutular elenir (en yüksek güvene sahip kutu tutulur)
- Nihai mAP skoruna %2-3 katkı sağlar



YOLO Amaç Fonksiyonu

- YOLO için aşağıdaki kaybı en aza indirmemiz gerekir
- Karesel hata toplamı kullanılır

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

Koordinat Kaybı: Tahmin edilen x,y,w,h ile gerçek x,y,w,h arasındaki farkı en aza indirir. YALNIZCA hücrede nesne varsa ve kutu tahminden sorumluysa

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

Güven Kaybı: YALNIZCA nesne varsa güvene dayalı kayıp

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

Nesne Yok Kaybı: nesne yoksa güvene dayalı kayıp

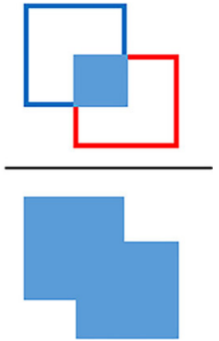
$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2$$

Sınıf kaybı: hücredeki nesnenin gerçek sınıfıyla arasındaki kaybı en aza indirir

Deney Düzeni

- Yazarlar YOLO'yu yukarıda açıklanan önceki çalışmalarla PASCAL VOC 2007 ve VOC 2012 üzerinde, ayrıca alan dışı bir sanat veri kümesinde karşılaştırır
- IOU değeri 0,5'in üzerindeyse ve sınıf doğruysa tespit doğru sayılır
- İki başarım metriği kullanılır:
  - mAP skoru: ortalama kesinlik ortalaması
  - FPS: saniyedeki kare sayısı
- Fast YOLO eklenir: daha az parametreye sahiptir

$$IoU = \frac{\text{kesişim alanı}}{\text{birleşim alanı}} =$$



Deney Sonuçları

- Temel YOLO, gerçek zamanlı dedektörleri büyük farkla geride bırakır
- Gerçek zamanlı olmayanların çoğundan da daha iyi sonuç verir

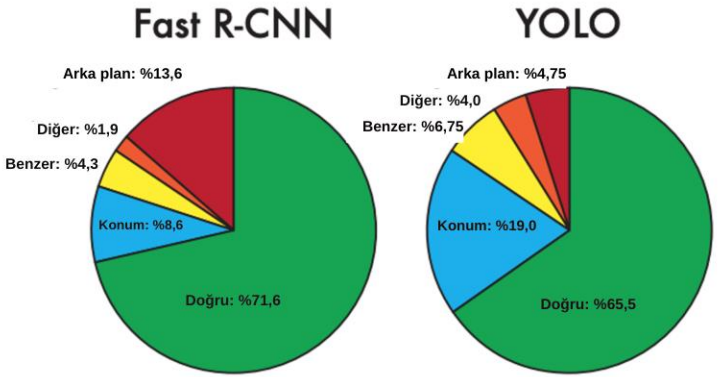
Gerçek Zamanlı Dedektörler	Eğitim	mAP	FPS
100Hz DPM [31]	2007	16.0	100
30Hz DPM [31]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Gerçek Zamanlı Olmayanlar			
Fastest DPM [38]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[28]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

Deney Sonuçları

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

Deney Sonuçları - Hata Analizi

- Çok daha az arka plan hatası yapar (arka planda yanlış pozitif tahmin etme olasılığı daha düşük)
  - Herhangi bir gerçek etiketle IOU ÇOK küçük
- Ancak çok daha fazla konum hatası yapar
  - Sınıf doğru, IOU görece küçük



Konum hatası



Arka plan hatası



Deney Sonuçları - Alan Dışı

- YOLO ve rakipleri (doğal görüntülerle eğitilmiş) sanat eserleri üzerinde çalıştırıldı
- Küresel bağlamın büyük ölçüde yardımcı olduğu sanatsal veri kümelerinde iyi sonuç verir

	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	



Sonuçların Tartışılması

- Artı: YOLO, görüntüde tespit için diğer algoritmalardan çok daha hızlıdır
- Artı: YOLO'nun yalnızca yerel bilgi yerine küresel bilgi kullanması, nesne tespiti yaparken bağlamsal bilgiyi anlamasını sağlar
  - Bu sayede sanat eserleri gibi alanlarda daha iyi sonuç verir
- Eksi: YOLO, nesne tespitinde SOTA modellerin gerisinde kaldı
  - Bunun nedeni çok sayıda konum hatası yapması ve küçük nesneleri tespit edememesidir

Eleştiri / Sınırlılıklar / Açık Sorunlar

- Başarımı SOTA'nın gerisinde kalıyor
- Verilerin sınırlayıcı kutularla etiketlenmesini gerektirir; çok sayıda sınıf için toplanması zordur
  - Önceki çalışmalar görüntü sınıflandırıcı kullandığı için daha iyi genelleyebiliyordu
  - 2014 COCO veri kümesi (çok büyük bir veri kümesi) bu sorunu kısmen giderdi
- Deneylerle ilgili: tahmin edilen sınıf sayısı çok sınırlı
  - YOLO v1'in genellenebilir olduğuna ikna olmuş değilim
- YOLO'nun güven çıktısı sınıfın güveni değil P(Nesne)'dir; bu da yorumlanabilirliği azaltır
- YOLO'nun bir diğer sınırlılığı, SxS ızgarada yalnızca B kutu tahmin edilebildiği için görüntüdeki nesnelere uzamsal kısıtlar getirmesidir
- Mimari yalnızca kutu tahmin ettiğinden, düzensiz şekiller için daha az kullanışlı olabilir

Gelecek Çalışmalar / Okumalar

- Bu çalışmanın bir uzantısı, görüntü bölütlemeye bakıp buradaki fikirlerin oraya aktarılıp aktarılamayacağını incelemek olabilir
  - YOLACT (Bolya vd. 2019): Gerçek zamanlı görüntü bölütleme
- YOLO iki kez güncellendi
  - Küçük nesnelerin tespiti, genellenebilirlik ve konum belirleme ile ilgili birçok sorunu çözer



YOLACT örneği

Ek Okumalar

- YOLO v2 (https://arxiv.org/abs/1612.08242) (çalışmayı büyük ölçüde genişletir) (Redmon vd. 2016)
  - Genellenebilirlik sorununu ele alır, 9000 sınıfa sahiptir
  - Sınıf olasılık dağılımı ızgara başına değil, sınırlayıcı kutu başına
  - Yüksek çözünürlüklü sınıflandırıcı (yüksek çözünürlükte ince ayar)
  - Toplu normalizasyon (batch norm)
  - MSCOCO ile eğitildi (YOLO v1'den sonra yayımlandı)
- YOLO v3 (https://arxiv.org/abs/1804.02767)
  - "Artımsal İyileştirme"
  - Sınıflar için bağımsız lojistik sınıflandırıcılar kullanır
    - Sınıflarda daha fazla özgüllük sağlar

	YOLO							YOLOv2
batch norm?		✓	✓	✓	✓	✓	✓	✓
hi-res classifier?			✓	✓	✓	✓	✓	✓
convolutional?				✓	✓	✓	✓	✓
anchor boxes?				✓	✓			
new network?					✓	✓	✓	✓
dimension priors?						✓	✓	✓
location prediction?						✓	✓	✓
passthrough?							✓	✓
multi-scale?							✓	✓
hi-res detector?								✓
VOC2007 mAP	63.4	65.8	69.5	69.2	69.6	74.4	75.4	76.8	**78.6**

Table 2: The path from YOLO to YOLOv2. Most of the listed design decisions lead to significant increases in mAP. Two exceptions are switching to a fully convolutional network with anchor boxes and using the new network. Switching to the anchor box style approach increased recall without changing mAP while using the new network cut computation by 33%.

Özet

- Nesne tespiti, bir görüntüdeki birden çok nesneyi tespit etme problemidir
- Neredeyse gerçek zamanlı nesne tespiti, karmaşık sensörler olmadan hızlı tepki veren robotlar sağlar
- Önceki çalışmalar, optimize edilmesi gereken çok sayıda parçadan oluşan büyük bir mimariye dayanır
- YOLO, tüm görevleri tek bir modelde ve tüm görüntü üzerinde tek bir çıkarımla yapan birleşik bir mimari önerir
- Yazarlar büyük bir hız artışı sağladıklarını ve mAP açısından önceki çalışmaların çoğunu geçebildiklerini gösterir