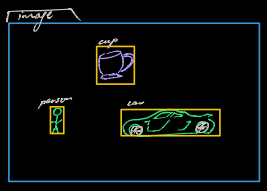
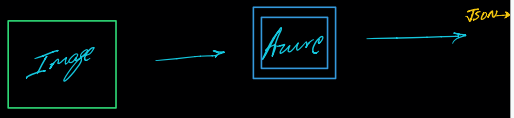
go through image segmentation first to get a better idea of this session.

companies have pretrained models and they're available as APIs. In this session we will use Azure's API

Azure API doesn't require credit card to try. Google does require it.

They have lots of features in the API & excellent documentation of all of that. I don't care, so I haven't noted it down. We can just look it up for any API.





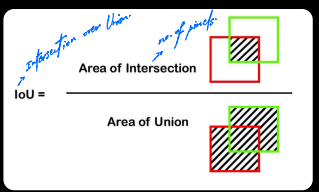

```
{
  "objects":[
    {
      "rectangle":{
        "x":730,
        "y":66,
        "w":135,
        "h":85
      },
      "object":"kitchen appliance",
      "confidence":0.501
    },
    {
      "rectangle":{
        "x":523,
        "y":377,
        "w":185,
        "h":46
      },
      "object":"computer keyboard",
      "confidence":0.51
    },
    {
      "rectangle":{
        "x":471,
        "y":218,
        "w":289,
        "h":226
      },
      "object":"Laptop",
      "confidence":0.85,
      "parent":{
        "object":"computer",
        "confidence":0.851
```


Limitations

It's important to note the limitations of object detection so you can avoid or mitigate the effects of false negatives (missed objects) and limited detail.

- Objects are generally not detected if they're small (less than 5% of the image).
- Objects are generally not detected if they're arranged closely together (a stack of plates, for example).
- Objects are not differentiated by brand or product names (different types of sodas on a store shelf, for example). However, you can get brand information from an image by using the Brand detection feature.

Performance metric



if we print json simply, it's ugly & non-indented

this function prints json cleanly

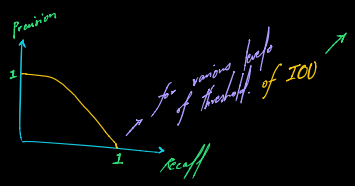
```
print(json.dumps(json_file))
```

chair , person , etc
class 1 class 2 , etc.

↓ Do this for each class

for each class → avg. precision

↓ area under this curve



At the end we have mean Average Precision (mAP)

↓ across all class

① Take image, predict all objects of class I

② get IOU for all boxes

③ for (IOU = 0.05 to 0.95, IOU += 0.05)

save (precision, recall)

↳ this is for that specific prediction

$\frac{TP}{TP + FP}$ } only from current prediction

④ Plot Pr - Recall graph & get area under curve

Not much detail about this in the live session.

Mask I/O

input → image

output → bounding box 1 , object class 1
bounding box 2 , object class 2
bounding box 3 , object class 3
:
so on

Dataset → COCO
80 classes
330K images
very good collection.

good for mAP

R-CNN
fast R-CNN
faster R-CNN

SSD
RetinaNet

Yolo V1
Yolo V2
Yolo V3 → April 2018

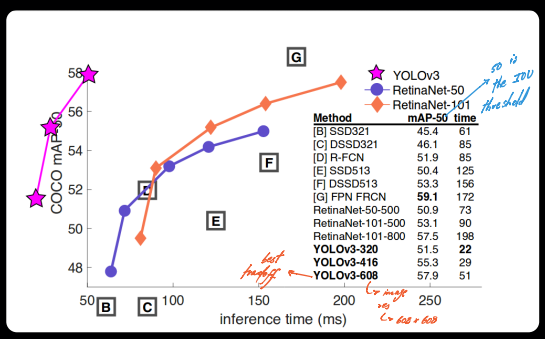
Trade - off

speed vs mAP → medical use

↓ self driving car
real time applications

Precision vs Average Precision

Precision refers to precision at a particular decision threshold. For example, if you count any model output less than 0.5 as negative, and greater than 0.5 as positive. But sometimes (especially if your classes are not balanced, or if you want to favor precision over recall or vice versa), you may want to vary this threshold. Average precision gives you average precision at all such possible thresholds, which is also similar to the area under the precision-recall curve. It is a useful metric to compare how well models are ordering the predictions, without considering any specific decision threshold.



YOLO V3 - feature extractor (DarkNet-53)

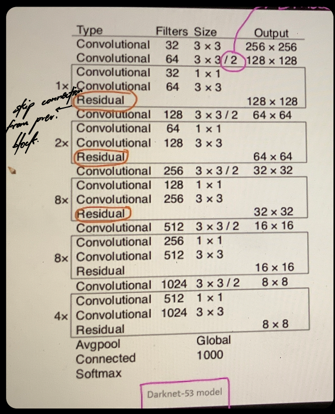
Fully convolutional Network (FCN) → no maxpools

conv -> Batch Norm → LeakyRelu

no maxpooling → they use stride = 2 for CNN
kinda works the same.

pretrained on ImageNet.

skip connections allow deeper 53 layer architecture.
↓
YOLO v2 used
Darknet19 (it didn't have
skip connections)



	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Darknet-53 model

skip connection from prev. block.

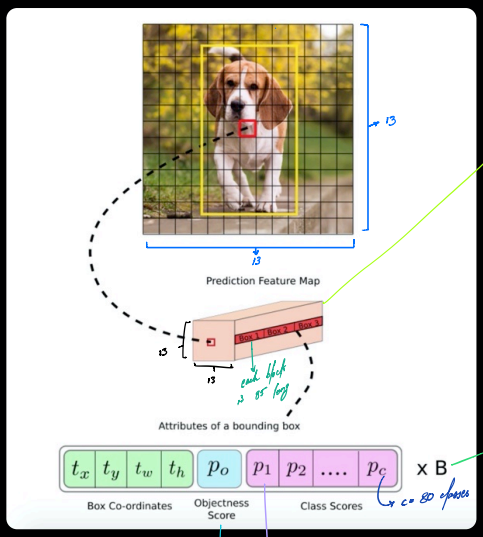
53 CNN layers

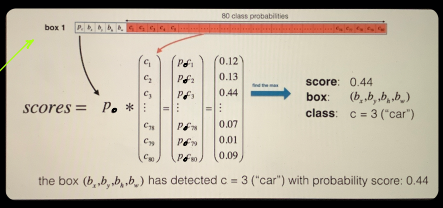
why not use ResNet as
feature extractor? It already has stellar performance.
It has same Top1, Top5 but half
the FPS. Performance sucks. Thats why.
↓
YOLO V3 → 78 FPS
YOLO V2 → 171 FPS
ResNeT152 → 37 FPS → MEH

we'll use YOLOv3416 as an
example, it's balanced speed & accuracy.

input image → 416 × 416 × 3
↓ because of CNN stride = 2 → 5 of them in between
feature extracted → 13 × 13 × 1024
↳ 1024 filters in the last layer.
↓
from DarkNet53
pre trained on ImageNet

Bounding boxes & output





→ no. of boxes.
(↳ this block of image can be home to multiple bounding box centres.

for each bounding box, we need $(5 + 80) = 85$ numbers.

We can change that 1024
(↳ depending on how many boxes we want for each block of image.

probability of this box having something of interest

prob. for each class

probability (class 10 object) $= P_o * P_{10}$

for eg. if I want 5 BB for each block is.

$13 \times 13 \times (5 * (5 + 80))$
↓ no. of boxes =5 ↓ classes

$\therefore 13 \times 13 \times 425$

in yolo they use 13 = 5
smaller 13 = faster. → that much less stuffs to predict.
larger 13 = larger mAP

all this is differentiable, so it works.

darknet gives us 13x13x1024 tho...

these are other convolutions involved

425
1x1 conv

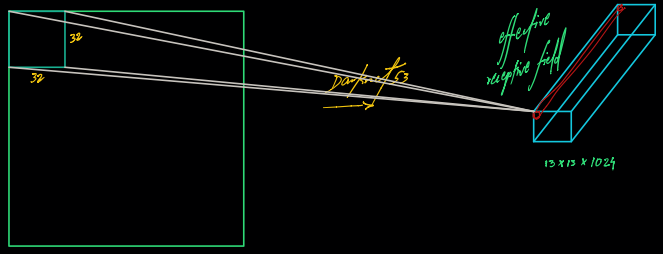
$\therefore 13 \times 13 \times 425$ → they used this thing to optimize together

this is simplified version

→ interesting result of YOLO v3

If there's only one object, that "Pc" ↓ probability of having an object of interest will be zero or close to zero

YOLO is designed to work on square images but we can always resize the input to be a square



what if the object spans multiple receptive fields?

↓

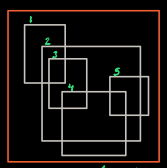
the center of BB has to inside receptive field, the height/width can be anything

↓

anchor boxes solve some of the issue

first they tried to predict (bx, by, bw, bh) for a bounding box → Turned out to be too hard to train.

solution → have anchor boxes.



5 anchor boxes

let's represent bb as a modification of anchor boxes

↳ (cx, cy, pw, ph)

The increased performance by 5% or 4% by using anchor boxes.

range → 0 to ∞ → it'll be utmost one priority away. They want bb to be very close to anchor boxes

sigmoid function

σ(tx) + cx

σ(ty) + cy

(bx, by, bw, bh)

$p_w \cdot e^{t_w}$

$p_h \cdot e^{t_h}$