# YOLOv3
# An Incremental Improvement

Allen Wu
PyTorch Taipei @NTU
2018.10.18

# Reference

- YOLOv2 NTU 9/27 by Ray Kuo
- YOLOv3: An Incremental Improvement

# Agenda

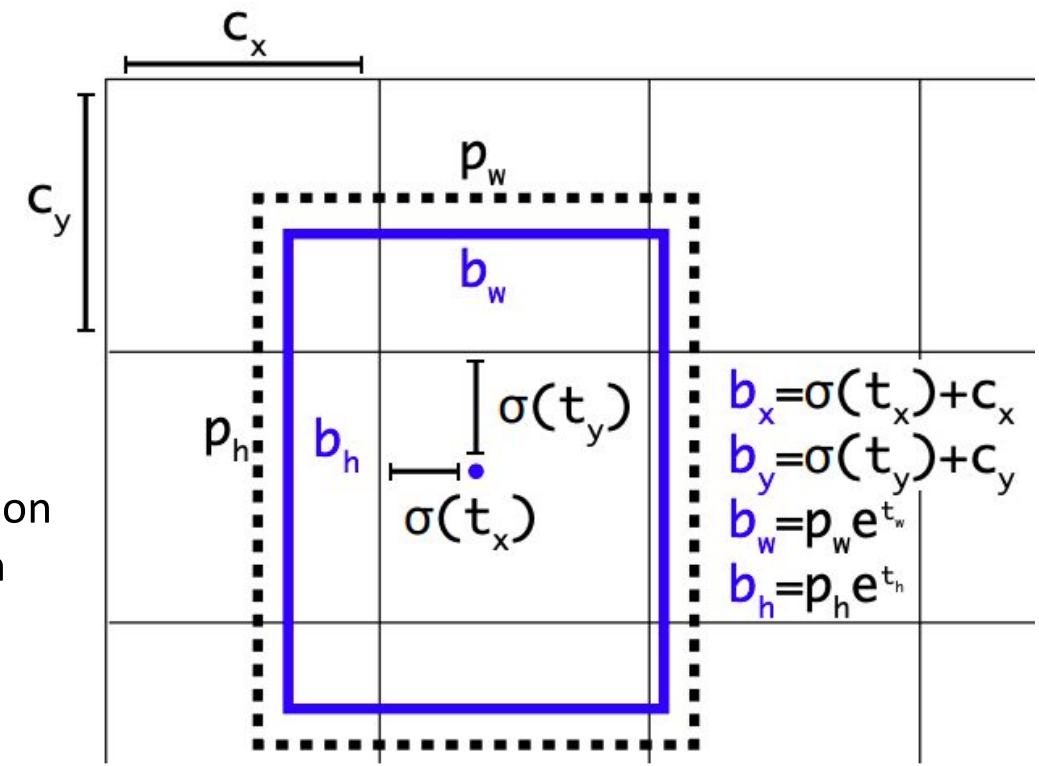
- Incremental Improvements
  - Bounding Box / Class Prediction
  - Predictions Across Scales (like FPN)
  - New Backbone: Darknet-53
    - Backbone Comparison

- Training for COCO
  - COCO Metrics
  - Performance Comparison

- What Didn’t Work
- More Incremental Improvements
  - Spatial Pyramid Pooling (SPP)
- Rebuttal
- Summary

# Bounding Box Prediction

- Following YOLOv2, using dimension clusters as anchor boxes ($p_w$ , $p_h$), 9 clusters for COCO.
- Predicting $t_x$ , $t_y$ , $t_w$ , $t_h$
- Bounding boxes with dimension priors and location prediction
- Training by sum of squared error loss ($\hat{t}_* - t_*$)

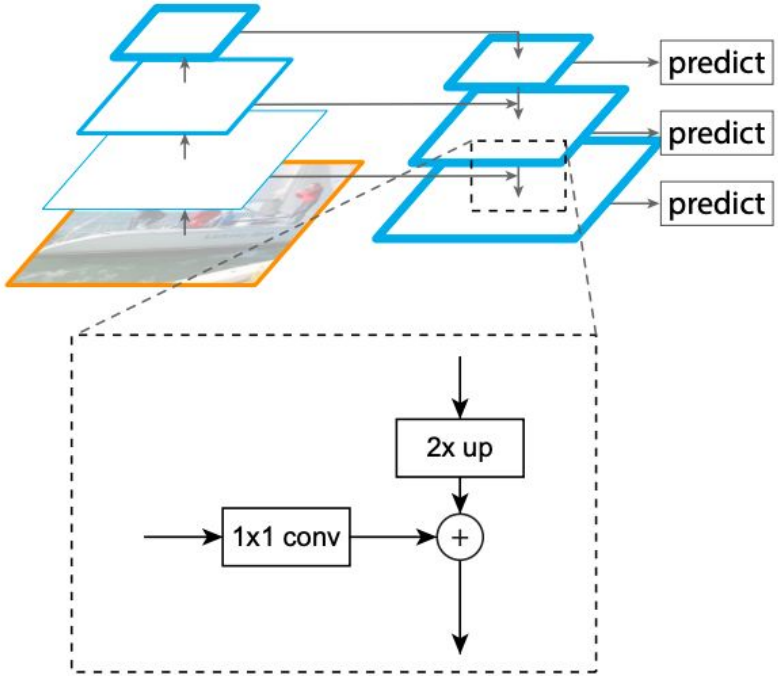


# Class Prediction

- Using **binary cross-entropy loss** for multi-label classification instead of (multiple) softmax.
- Moving to more complex domains like the Open Images Dataset: https://ai.googleblog.com/2016/09/introducing-open-images-dataset.html

# Predictions Across Scales (like FPN)

- YOLO was struggling on detecting small object.
- Extracting features from those scales using a similar concept to feature pyramid networks (FPN) which is good on different scales.
- YOLOv3 predicts boxes at 3 different scales.



# Predictions Across Scales (like FPN)

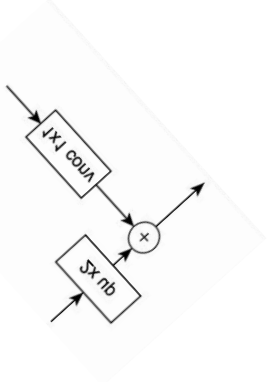
- Predicting B boxes at each scale with N x N grid, the tensor is: $N \times N \times [B * (4 + 1 + C)]$, where:
    - B = 3
    - 4 bounding box offsets
    - 1 objectness prediction
    - C = 80 since COCO has 80 classes.

- We just sort of chose 9 clusters and 3 scales arbitrarily and then divide up the clusters evenly across scales.
    - On the COCO dataset the 9 clusters were:
      (10×13),(16×30),(33×23),
      (30×61),(62×45),(59× 119),
      (116 × 90),(156 × 198),(373 × 326).

# New Backbone

- Feature Extractor: Darknet-53
  - 53 convolutional layers
  - Includes
    - 3 x 3 convolutional layer
    - 1 x 1 convolutional layer (Network In Network)
    - Residual layer (including shortcut 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 | | | |



# Backbone Comparison

- Based on ImageNet
- Run times are measured on a Titan X at 256 x 256.
- 1.5x faster than ResNet-101 (ResNet has too many layers and inefficient)
- High BFLOP/s means better utilizes the GPU

| Backbone | Top-1 | Top-5 | Bn Ops | BFLOP/s | FPS |
|---|---|---|---|---|---|
| Darknet-19 [15] | 74.1 | 91.8 | 7.29 | 1246 | **171** |
| ResNet-101[5] | 77.1 | 93.7 | 19.7 | 1039 | 53 |
| ResNet-152 [5] | **77.6** | **93.8** | 29.4 | 1090 | 37 |
| Darknet-53 | 77.2 | **93.8** | 18.7 | **1457** | 78 |

# COCO Metrics

- **Average Precision (AP):**
  - AP % AP at IoU=.50:.05:.95 **(primary challenge metric)**
    - Specifically we use 10 IoU thresholds of .50:.05:.95. This is a break from tradition, where AP is computed at a single IoU of .50 (which corresponds to our metric APIoU=.50). **Averaging over IoUs rewards detectors with better localization**.
  - $AP^{IoU=.50}$ % AP at IoU=.50 (PASCAL VOC metric)
  - $AP^{IoU=.75}$ % AP at IoU=.75 (strict metric)

- Reference:
  - http://cocodataset.org/#detection-eval
  - https://github.com/rafaelpadilla/Object-Detection-Metrics

COCO
Common Objects in Context

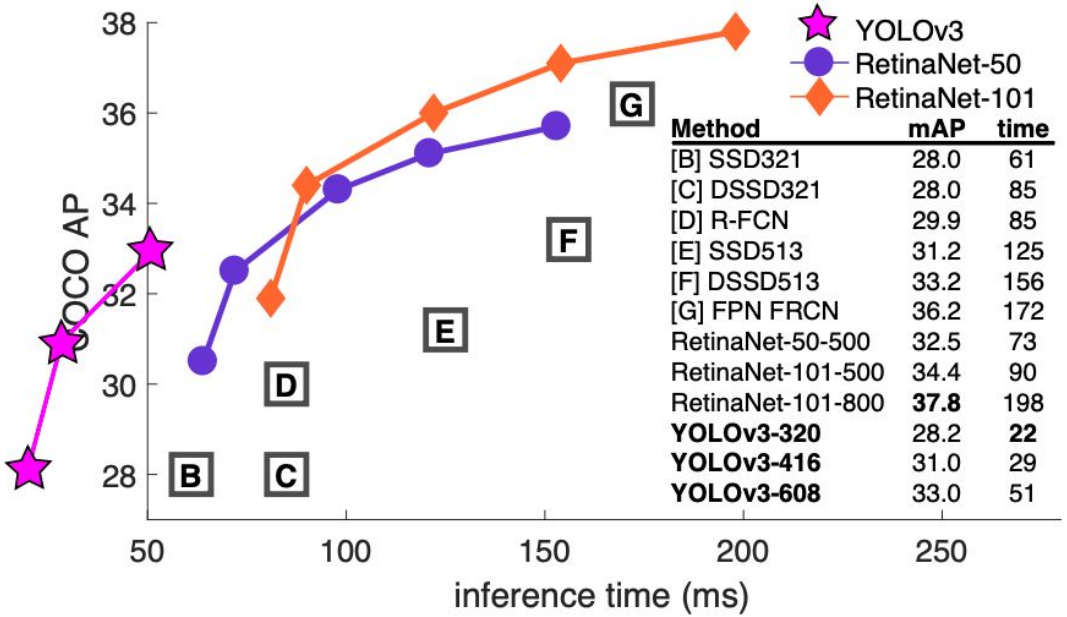
# Training for COCO – RetinaNet is quite good

- Training with **NO** hard negative mining.
- YOLOv3 is quite a bit behind other models like **RetinaNet** …

| | backbone | AP | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ |
|---|---|---|---|---|---|---|---|
| *Two-stage methods* | | | | | | | |
| Faster R-CNN+++ [5] | ResNet-101-C4 | 34.9 | 55.7 | 37.4 | 15.6 | 38.7 | 50.9 |
| Faster R-CNN w FPN [8] | ResNet-101-FPN | 36.2 | 59.1 | 39.0 | 18.2 | 39.0 | 48.2 |
| Faster R-CNN by G-RMI [6] | Inception-ResNet-v2 [21] | 34.7 | 55.5 | 36.7 | 13.5 | 38.1 | 52.0 |
| Faster R-CNN w TDM [20] | Inception-ResNet-v2-TDM | 36.8 | 57.7 | 39.2 | 16.2 | 39.8 | **52.1** |
| *One-stage methods* | | | | | | | |
| YOLOv2 [15] | DarkNet-19 [15] | 21.6 | 44.0 | 19.2 | 5.0 | 22.4 | 35.5 |
| SSD513 [11, 3] | ResNet-101-SSD | 31.2 | 50.4 | 33.3 | 10.2 | 34.5 | 49.8 |
| DSSD513 [3] | ResNet-101-DSSD | 33.2 | 53.3 | 35.2 | 13.0 | 35.4 | 51.1 |
| RetinaNet [9] | ResNet-101-FPN | 39.1 | 59.1 | 42.3 | 21.8 | 42.7 | 50.2 |
| RetinaNet [9] | ResNeXt-101-FPN | **40.8** | **61.1** | **44.1** | **24.1** | **44.2** | 51.2 |
| YOLOv3 608 × 608 | Darknet-53 | 33.0 | 57.9 | 34.4 | 18.3 | 35.4 | 41.9 |

# Performance on COCO AP

Figure adapted from "*Focal loss for dense object detection*".



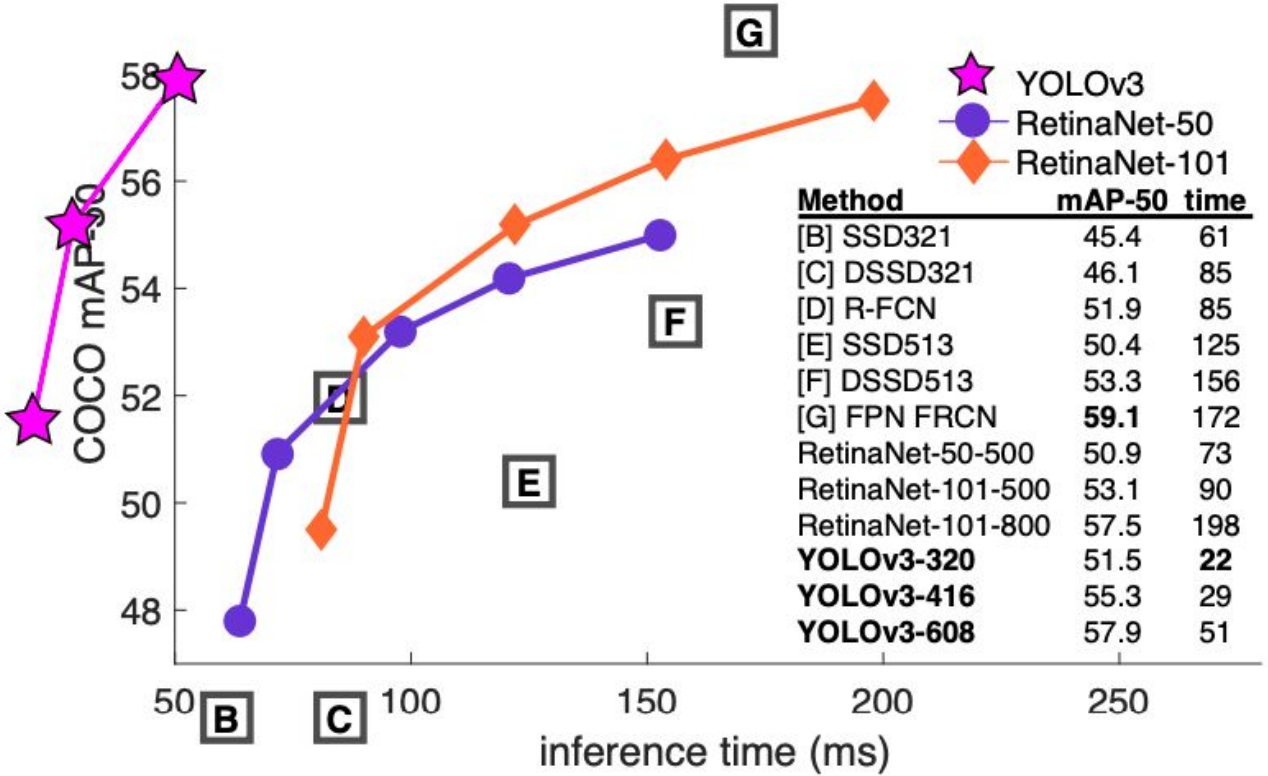

| Method | mAP | time |
|---|---|---|
| [B] SSD321 | 28.0 | 61 |
| [C] DSSD321 | 28.0 | 85 |
| [D] R-FCN | 29.9 | 85 |
| [E] SSD513 | 31.2 | 125 |
| [F] DSSD513 | 33.2 | 156 |
| [G] FPN FRCN | 36.2 | 172 |
| RetinaNet-50-500 | 32.5 | 73 |
| RetinaNet-101-500 | 34.4 | 90 |
| RetinaNet-101-800 | **37.8** | 198 |
| **YOLOv3-320** | 28.2 | **22** |
| **YOLOv3-416** | 31.0 | 29 |
| **YOLOv3-608** | 33.0 | 51 |

# Training for COCO – Fast and better

- YOLOv3 On par with SSD, **3.8x faster** than RetinaNet.
- YOLOv3 is **strong** at $AP_{50}$
- YOLO still struggles to get the boxes perfectly …

| | backbone | AP | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ |
|---|---|---|---|---|---|---|---|
| *Two-stage methods* | | | | | | | |
| Faster R-CNN+++ [5] | ResNet-101-C4 | 34.9 | 55.7 | 37.4 | 15.6 | 38.7 | 50.9 |
| Faster R-CNN w FPN [8] | ResNet-101-FPN | 36.2 | 59.1 | 39.0 | 18.2 | 39.0 | 48.2 |
| Faster R-CNN by G-RMI [6] | Inception-ResNet-v2 [21] | 34.7 | 55.5 | 36.7 | 13.5 | 38.1 | 52.0 |
| Faster R-CNN w TDM [20] | Inception-ResNet-v2-TDM | 36.8 | 57.7 | 39.2 | 16.2 | 39.8 | **52.1** |
| *One-stage methods* | | | | | | | |
| YOLOv2 [15] | DarkNet-19 [15] | 21.6 | 44.0 | 19.2 | 5.0 | 22.4 | 35.5 |
| SSD513 [11, 3] | ResNet-101-SSD | 31.2 | 50.4 | 33.3 | 10.2 | 34.5 | 49.8 |
| DSSD513 [3] | ResNet-101-DSSD | 33.2 | 53.3 | 35.2 | 13.0 | 35.4 | 51.1 |
| RetinaNet [9] | ResNet-101-FPN | 39.1 | 59.1 | 42.3 | 21.8 | 42.7 | 50.2 |
| RetinaNet [9] | ResNeXt-101-FPN | **40.8** | **61.1** | **44.1** | **24.1** | **44.2** | 51.2 |
| YOLOv3 608 × 608 | Darknet-53 | 33.0 | 57.9 | 34.4 | 18.3 | 35.4 | 41.9 |

# Performance on COCO mAP-50




| Method | mAP-50 | time |
|---|---|---|
| [B] SSD321 | 45.4 | 61 |
| [C] DSSD321 | 46.1 | 85 |
| [D] R-FCN | 51.9 | 85 |
| [E] SSD513 | 50.4 | 125 |
| [F] DSSD513 | 53.3 | 156 |
| [G] FPN FRCN | **59.1** | 172 |
| RetinaNet-50-500 | 50.9 | 73 |
| RetinaNet-101-500 | 53.1 | 90 |
| RetinaNet-101-800 | 57.5 | 198 |
| **YOLOv3-320** | 51.5 | **22** |
| **YOLOv3-416** | 55.3 | 29 |
| **YOLOv3-608** | 57.9 | 51 |

# Training for COCO – Good for small object

- YOLOv3 has relatively high $AP_S$ performance (for small object) now.
- What is the cause of relatively worse $AP_{M, L}$ performance ?

| | backbone | AP | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ |
|---|---|---|---|---|---|---|---|
| *Two-stage methods* | | | | | | | |
| Faster R-CNN+++ [5] | ResNet-101-C4 | 34.9 | 55.7 | 37.4 | 15.6 | 38.7 | 50.9 |
| Faster R-CNN w FPN [8] | ResNet-101-FPN | 36.2 | 59.1 | 39.0 | 18.2 | 39.0 | 48.2 |
| Faster R-CNN by G-RMI [6] | Inception-ResNet-v2 [21] | 34.7 | 55.5 | 36.7 | 13.5 | 38.1 | 52.0 |
| Faster R-CNN w TDM [20] | Inception-ResNet-v2-TDM | 36.8 | 57.7 | 39.2 | 16.2 | 39.8 | **52.1** |
| *One-stage methods* | | | | | | | |
| YOLOv2 [15] | DarkNet-19 [15] | 21.6 | 44.0 | 19.2 | 5.0 | 22.4 | 35.5 |
| SSD513 [11, 3] | ResNet-101-SSD | 31.2 | 50.4 | 33.3 | 10.2 | 34.5 | 49.8 |
| DSSD513 [3] | ResNet-101-DSSD | 33.2 | 53.3 | 35.2 | 13.0 | 35.4 | 51.1 |
| RetinaNet [9] | ResNet-101-FPN | 39.1 | 59.1 | 42.3 | 21.8 | 42.7 | 50.2 |
| RetinaNet [9] | ResNeXt-101-FPN | **40.8** | **61.1** | **44.1** | **24.1** | **44.2** | 51.2 |
| YOLOv3 608 × 608 | Darknet-53 | 33.0 | 57.9 | 34.4 | 18.3 | 35.4 | 41.9 |

# What Didn’t Work

- Anchor box x, y offset predictions
  - Predicting the x,y offset as a multiple of the box width or height using a linear activation
- Linear x, y predictions instead of logistic

- Focal loss
  - YOLOv3 may already be robust to the problem focal loss is trying to solve
    because it has separate objectness predictions and conditional class predictions.
    Thus for most examples there is no loss from the class predictions?
    Or something? We aren’t totally sure.

- Dual IOU thresholds and truth assignment (refer to Faster R-CNN)
  - IOU > .7 as positive example
  - IOU [.3 - .7] is ignored
  - IOU < .3 as negative example

# More Incremental Improvements

**Joe Redmon** @pjreddie · 8月1日

Added spatial pyramid pooling to YOLOv3, get 60.6 mAP@.5 up from 57.9, yay for more incremental improvements! Also, 33 -> 36.4 COCO mAP.

Try it out (you probably won't notice a difference) pjreddie.com/darknet/yolo/

翻譯推文

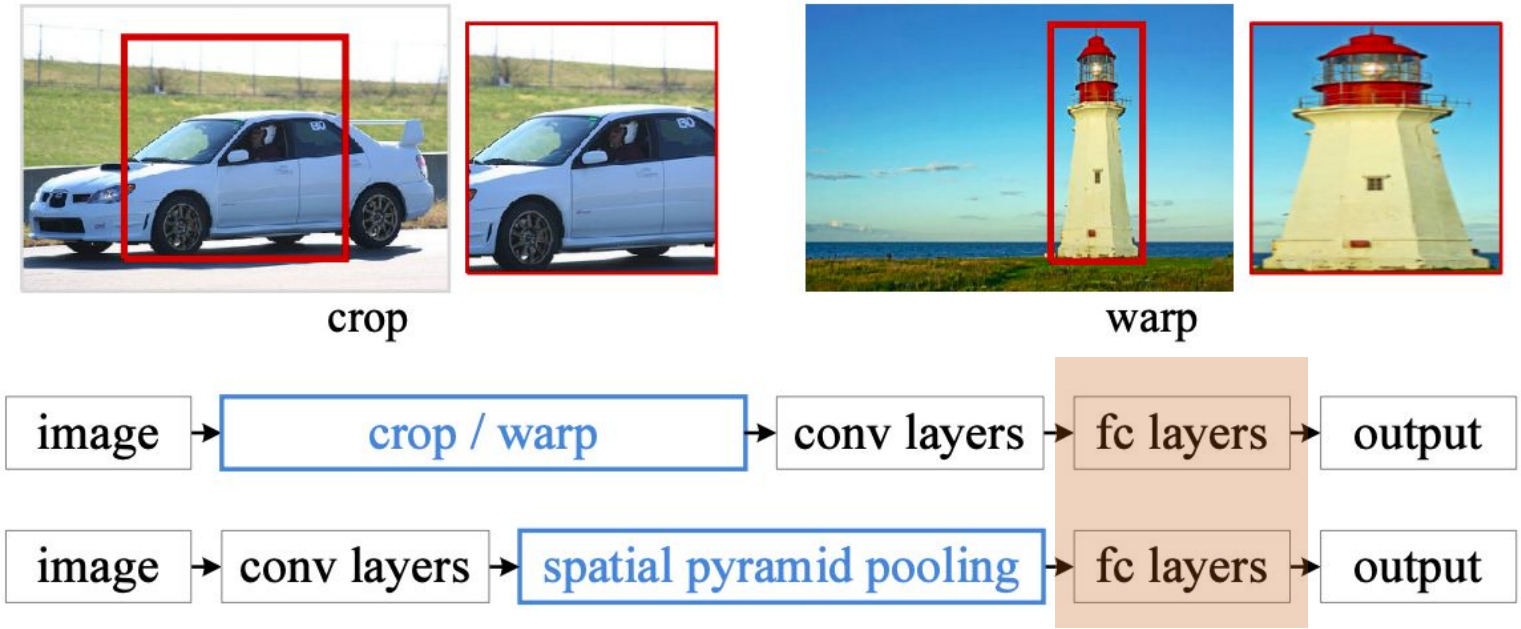
1 11 75

| | backbone | AP | $AP_{50}$ |
|---|---|---|---|
| *Two-stage methods* | | | |
| Faster R-CNN+++ [5] | ResNet-101-C4 | 34.9 | 55.7 |
| Faster R-CNN w FPN [8] | ResNet-101-FPN | 36.2 | 59.1 |
| Faster R-CNN by G-RMI [6] | Inception-ResNet-v2 [21] | 34.7 | 55.5 |
| Faster R-CNN w TDM [20] | Inception-ResNet-v2-TDM | 36.8 | 57.7 |
| *One-stage methods* | | | |
| YOLOv2 [15] | DarkNet-19 [15] | 21.6 | 44.0 |
| SSD513 [11, 3] | ResNet-101-SSD | 31.2 | 50.4 |
| DSSD513 [3] | ResNet-101-DSSD | 33.2 | 53.3 |
| RetinaNet [9] | ResNet-101-FPN | 39.1 | 59.1 |
| RetinaNet [9] | ResNeXt-101-FPN | **40.8** | **61.1** |
| YOLOv3 608 × 608 | Darknet-53 | 33.0 | 57.9 |
| YOLOv3 SPP | | 36.4 | 60.6 |

- Reference: https://twitter.com/pjreddiehttps://twitter.com/pjreddie/status/1024795604428308480

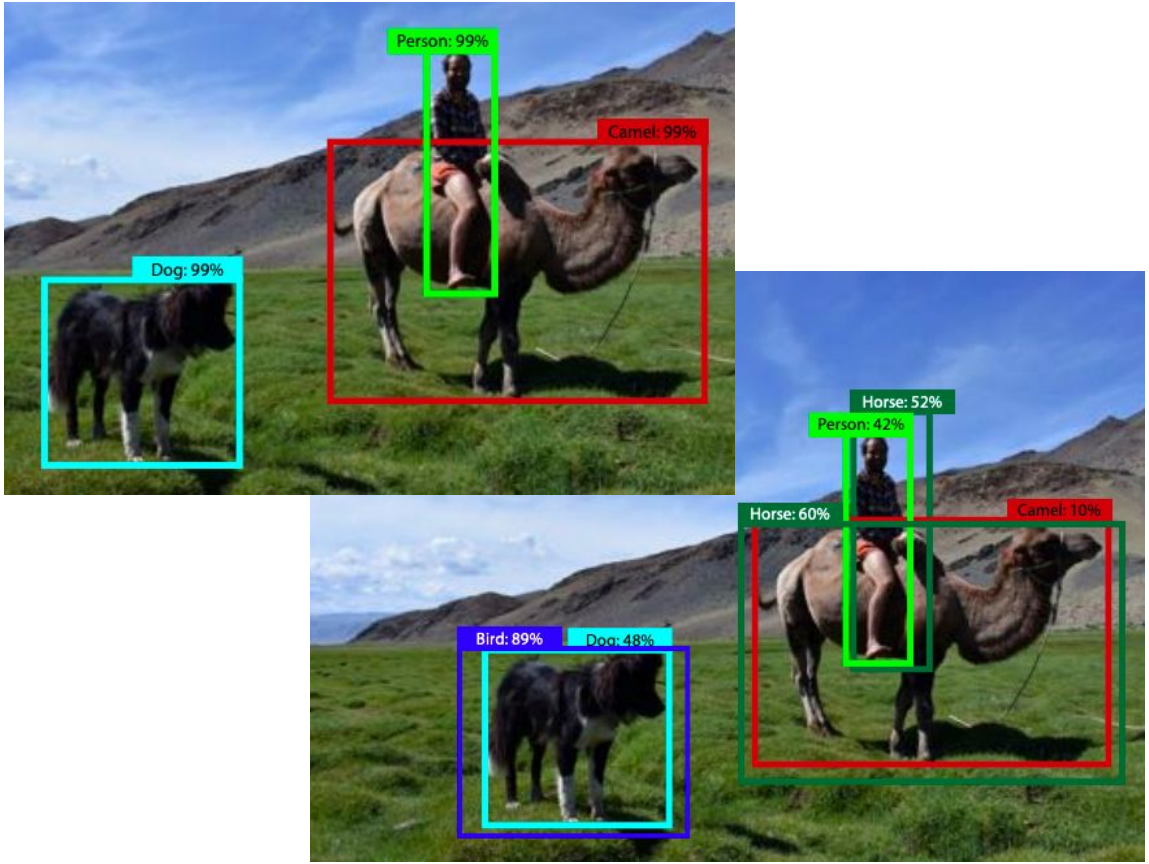
# Spatial Pyramid Pooling (SPP)

Reference: Spatial Pyramid Pooling in Deep Convolutional Networks for Visual ...



# Rebuttal

- Class vs. Localization
- VOC mAP
  = COCO (m)$AP_{50}$
- COCO metric (primary challenge metric) lack of justificaiton.
- New metrics should focus on discrepancies between **what people in the "real world" would care about** and our current metrics.



# Summary

- YOLOv3 is a good detector especially on the old detection metric of **.5 IOU**
- **Using Computer Vision for happy, good stuff.**

# Thank you !