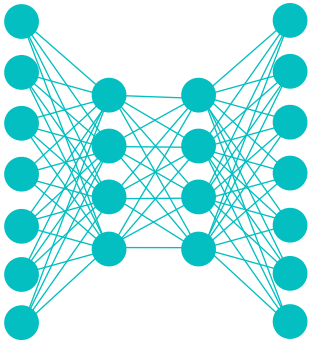
Lecture Notes for

Neural Networks and Machine Learning

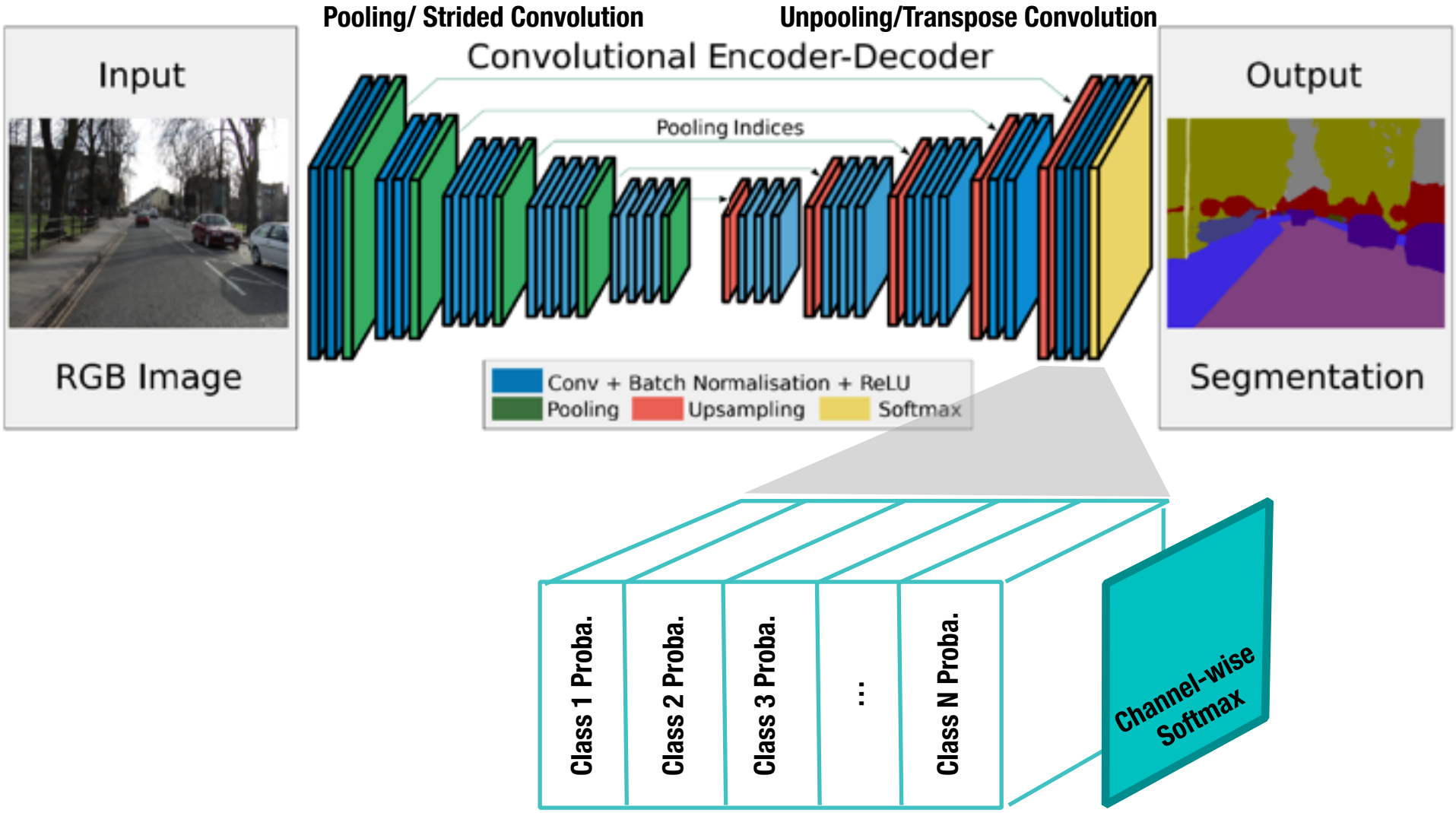
Fully Convolutional Learning II: Object Detection



Logistics and Agenda

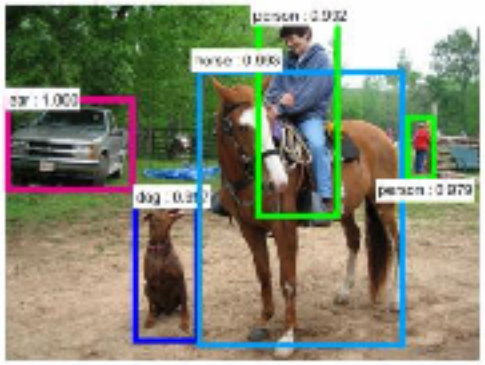
- Logistics
  - Next lab due soon
  - Lab grading updates
- Agenda
  - Full Convolutional Architectures
    - Semantic Segmentation Basics (last time)
    - Object Detection (this time):
      - RCNN, YOLO
    - Instance Segmentation (next time, probably):
      - Mask-RCNN, YOLACT

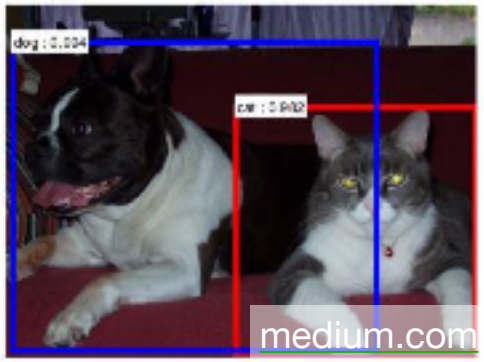
Last Time


Pooling/ Strided Convolution
Unpooling/Transpose Convolution
Input
Convolutional Encoder-Decoder
Output
Pooling Indices
RGB Image
Conv + Batch Normalisation + ReLU
Pooling
Upsampling
Softmax
Segmentation
Class 1 Proba.
Class 2 Proba.
Class 3 Proba.
...
Class N Proba.
Channel-wise Softmax

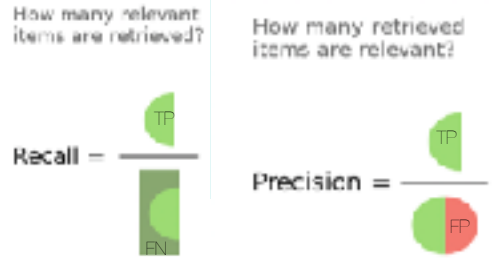
This time… Object Detection Methods

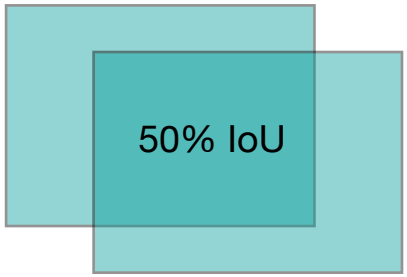
- Semantic segmentation is not great in terms of mIoU (some classes are at 50%)
- How to adapt these techniques to get bounding boxes, not semantic segmentations?
  - Could this be easier? More stable?
  - More consistent labeling?
  - Suitable for “higher risk” tracking applications?

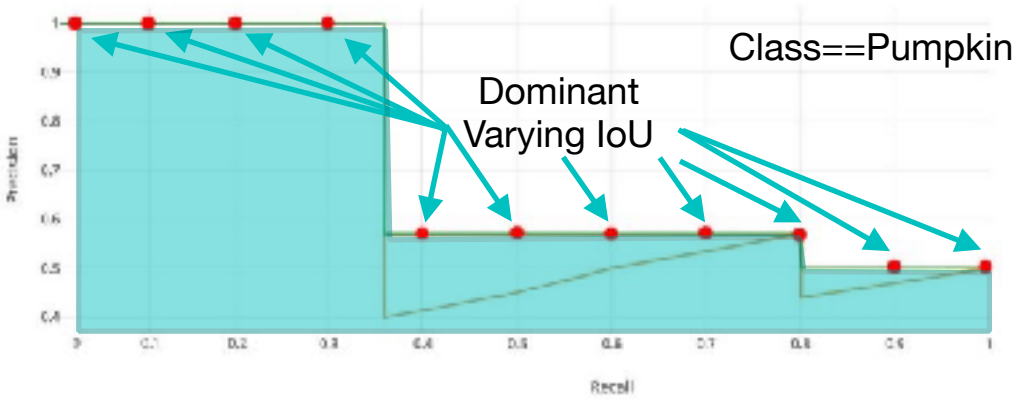




First: Measuring Performance







- mAP(IoU=x%)
  - if IoU > X%, check if correct
    - else not correct
  - Usually~50%, 75%, 90%
  - Define precision for each class, take average
- mAP(%), *sometimes just AP*
  - Formulate precision/recall curve for a class at varying levels of IoU (50%-95%)
  - Calculate dominating points
  - Take area under curve
  - Take average area over all classes (macro or micro averaging can be done, usually macro)

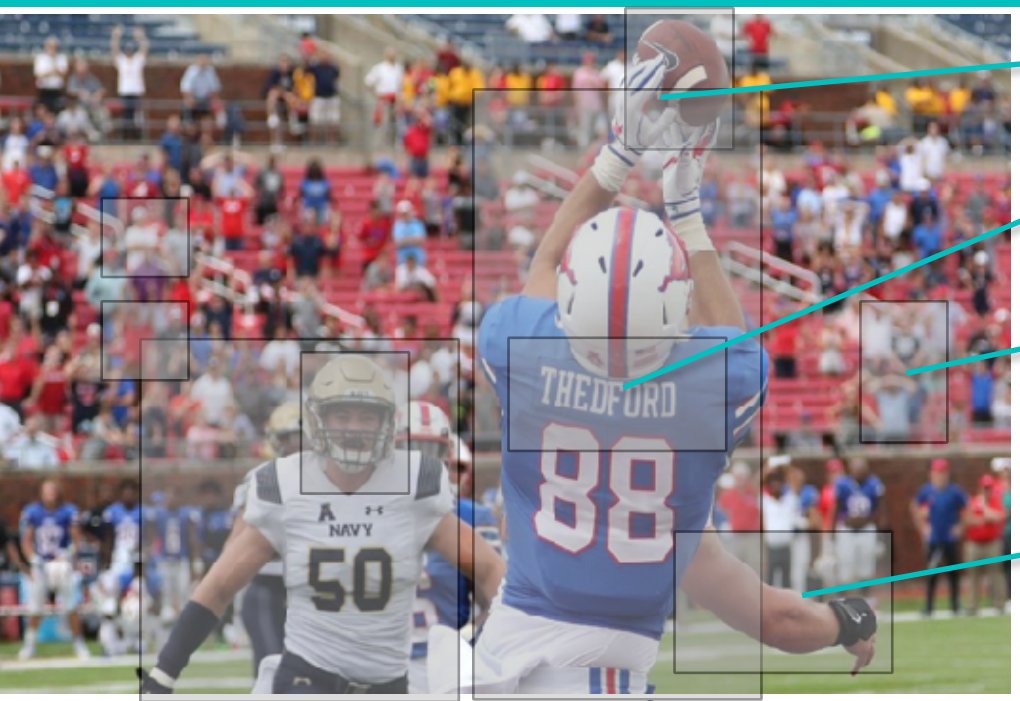
https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173

Object Detection with RCNN

A history in naming one network five different times
with five different papers
each time changing one thing about the architecture

Research!

2014: R-CNN

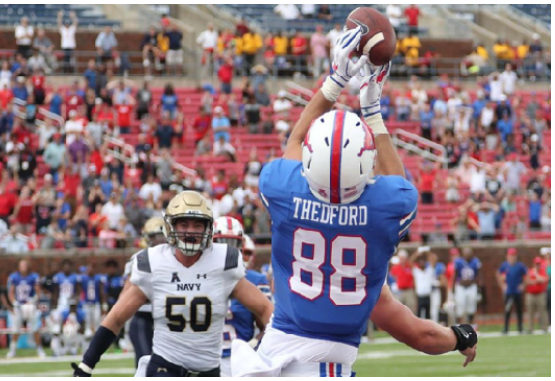


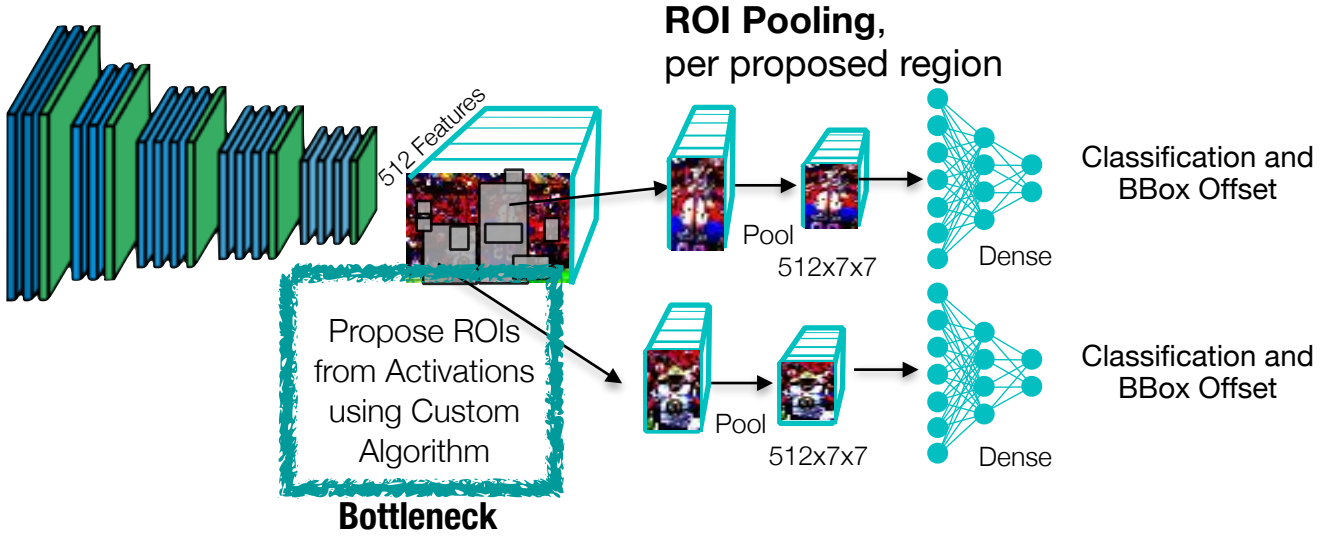
Propose BBox ROIs from Image

- Too Slow to Be Useful
- SVM and BBox Regression Trained Separately
- Fine Tuned Existing ConvNet (for Warped Images)
- ~50 Seconds per Image when Deployed

Girshick et al. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation, 2014

2015: Fast R-CNN

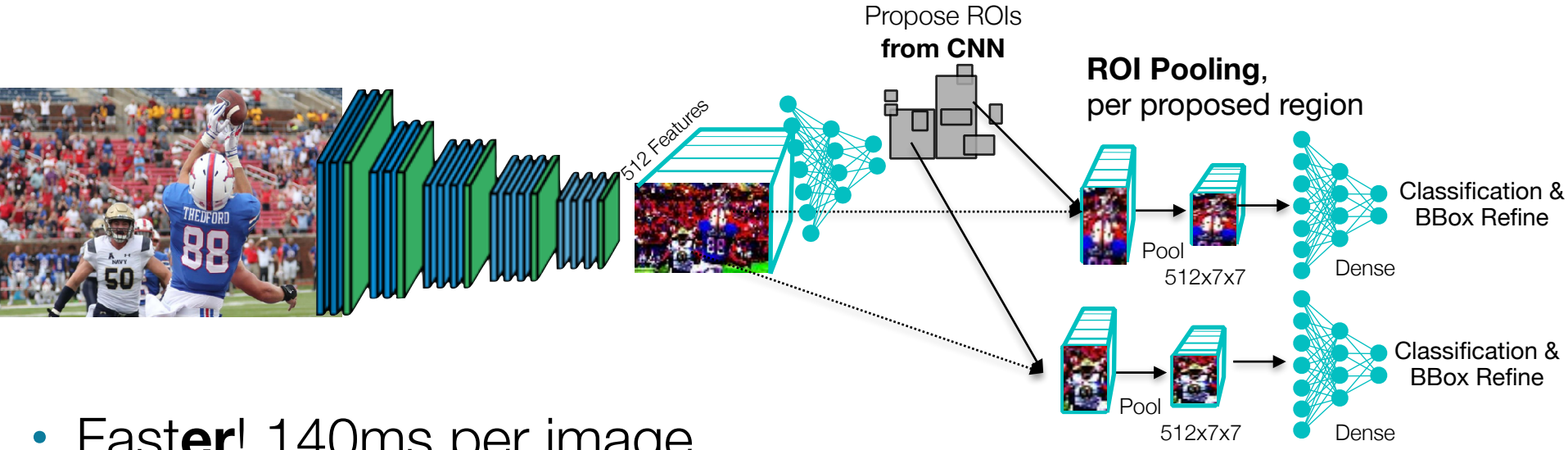




- Fast! 2.3 seconds per image (not ~50)
- But still not real time…

Girshick et al. Fast R-CNN, 2015, February

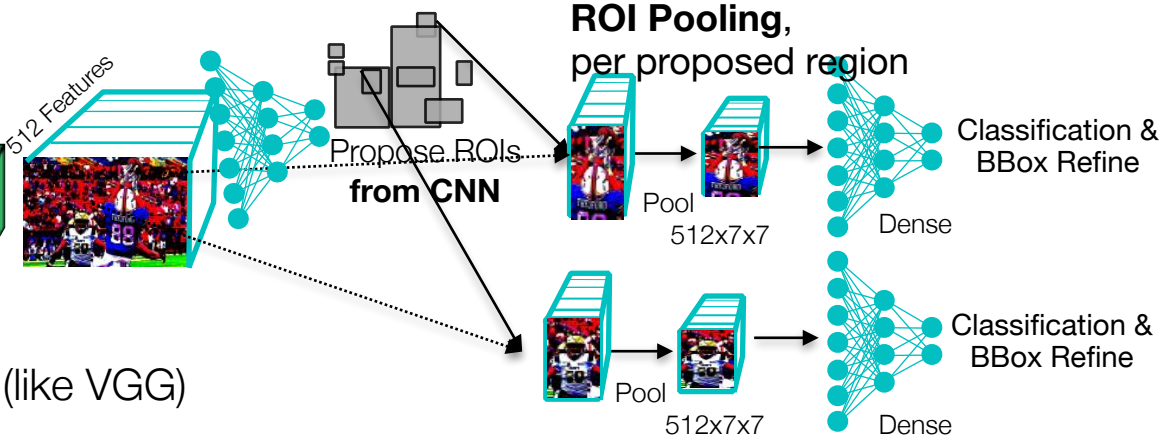
2015: Faster R-CNN



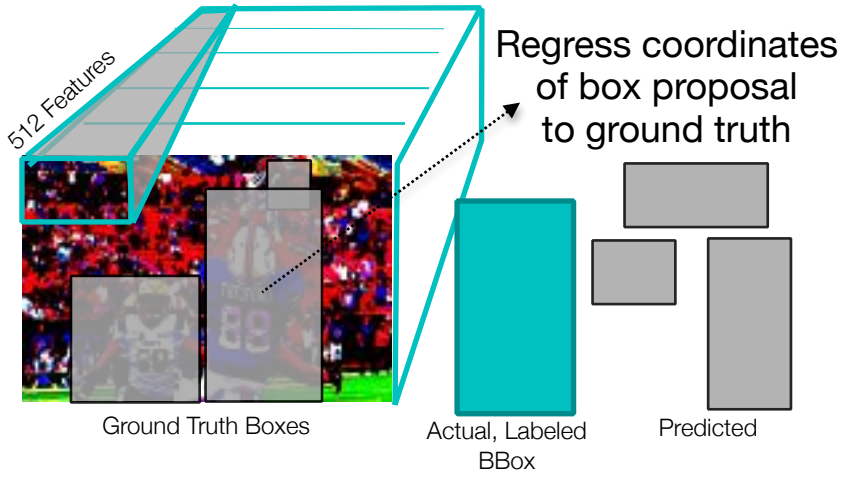
- Fast**er**! 140ms per image (7 FPS)
- Highly Accurate

Ren et al. Faster R-CNN, 2015, November

2015: Faster R-CNN, Training



- Pre-train image classifier backbone (like VGG)
- Train region proposal network
  - Sliding window (1x1 conv) across CNN activations from backbone classifier
  - Regress multiple bounding boxes (usually k proposals)
  - Regress "object-ness" of each box
- Train Fast R-CNN on generated ROI proposals
- Fix weights of classifier pipeline, fine tune RPN
- Fix RPN and fine tune classifier
- Rinse, repeat fine tuning



$$l_{box} = \sum_i \hat{p}_i \left[(x - \hat{x}_i)^2 + (y - \hat{y}_i)^2 + (\log w - \log \hat{w}_i)^2 + (\log h - \log \hat{h}_i)^2 \right]$$

$$l_{class} = \sum_c CE(c, \hat{c}) \qquad l_{obj} = \sum_i CE(p_i, \hat{p}_i)$$

Ren et al. Faster R-CNN, 2015, November

Object Detection with YOLO

- e researchers started working on a related, but different approach

Don’t want to listen to me explain it?

- Check out Joseph Redmon’s Talk at CVPR 2016:
  - https://www.youtube.com/watch?v=NM6lrxy0bxs
  - This is how you give a Technical presentation, plus he is from the school where I did graduate school… so he is clearly superior by grace of proximity…

2018: Everybody has a Gimmick

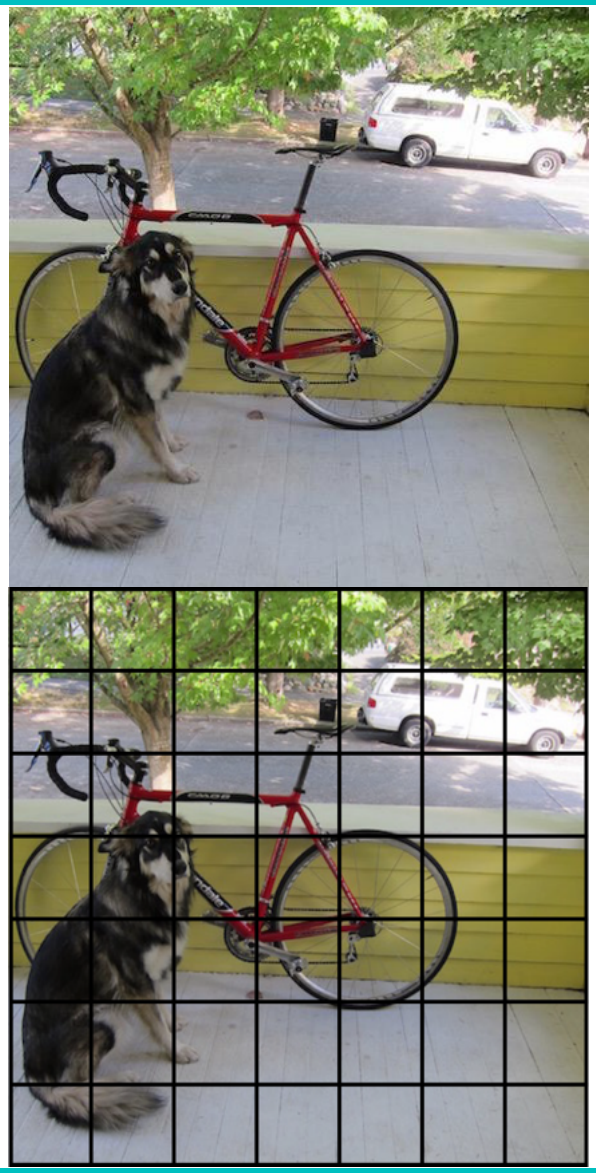
YOLO9000:
Better, Faster, Stronger

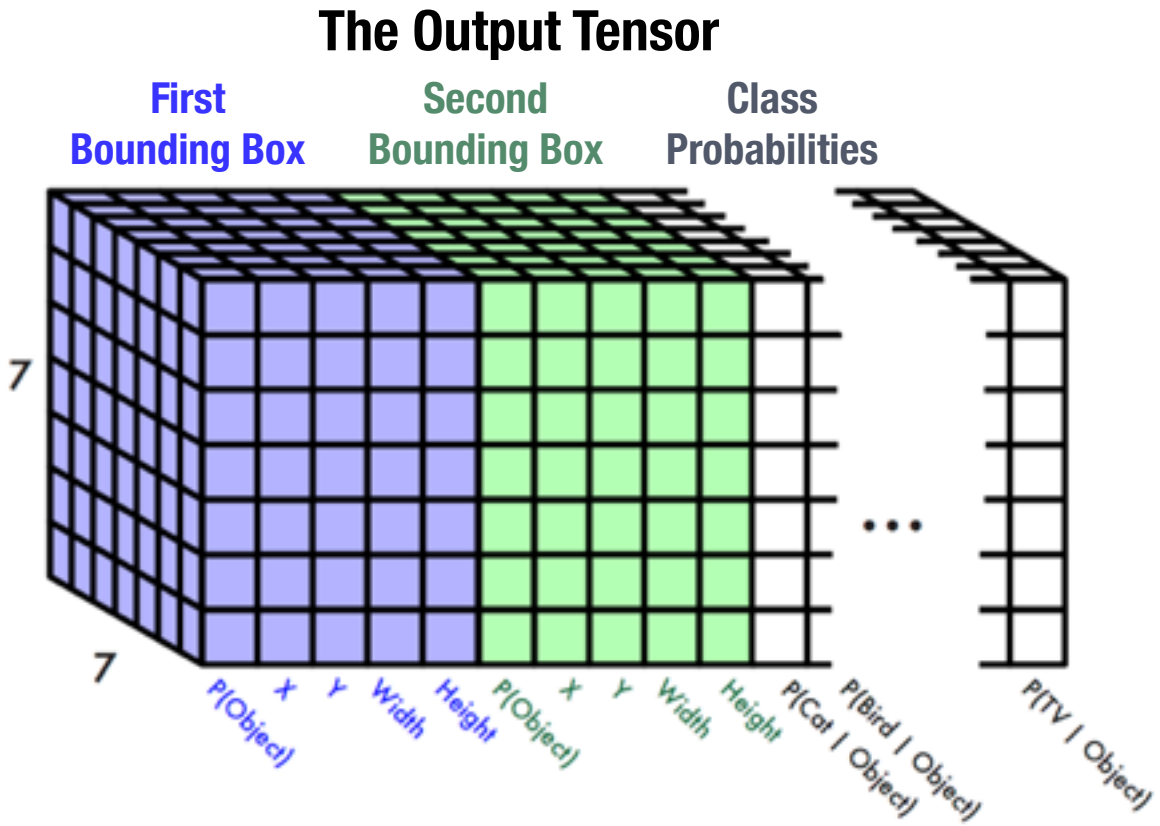
Joseph Redmon*†, Ali Farhadi*†
University of Washington*, Allen Institute for AI†
http://pjreddie.com/yolo9000/

- YOLO ~40-60 FPS
- Slightly more Accurate than **Faster R-CNN**

Redmon and Farhadi, YOLO9000: Better, Faster, Stronger, 2016, December 25 — *Merry Christmas?*

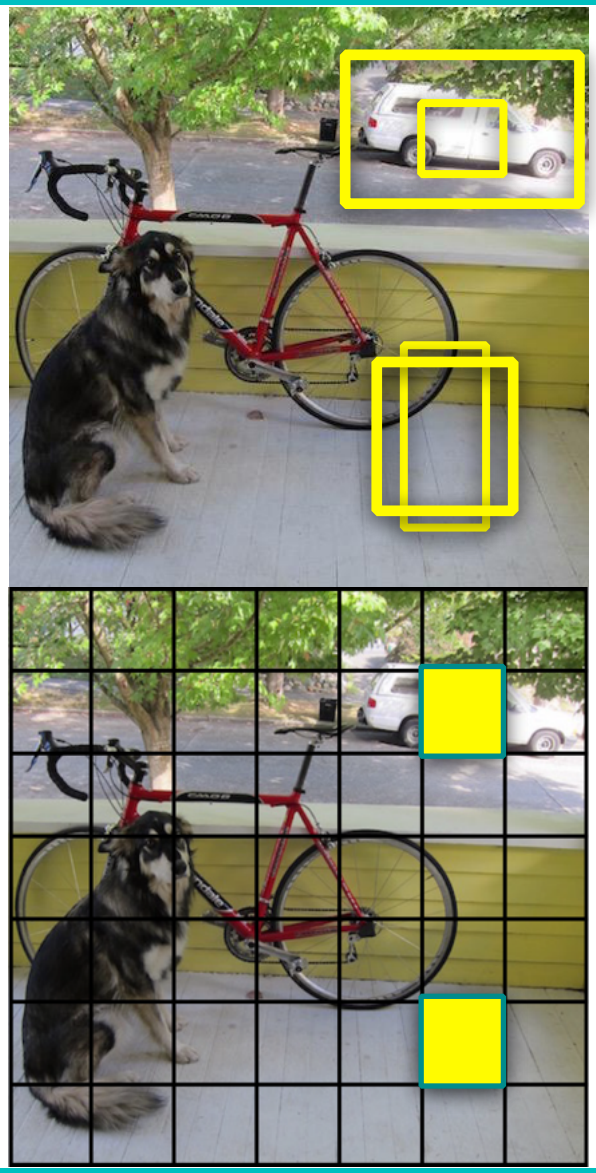
The YOLO Output Tensor

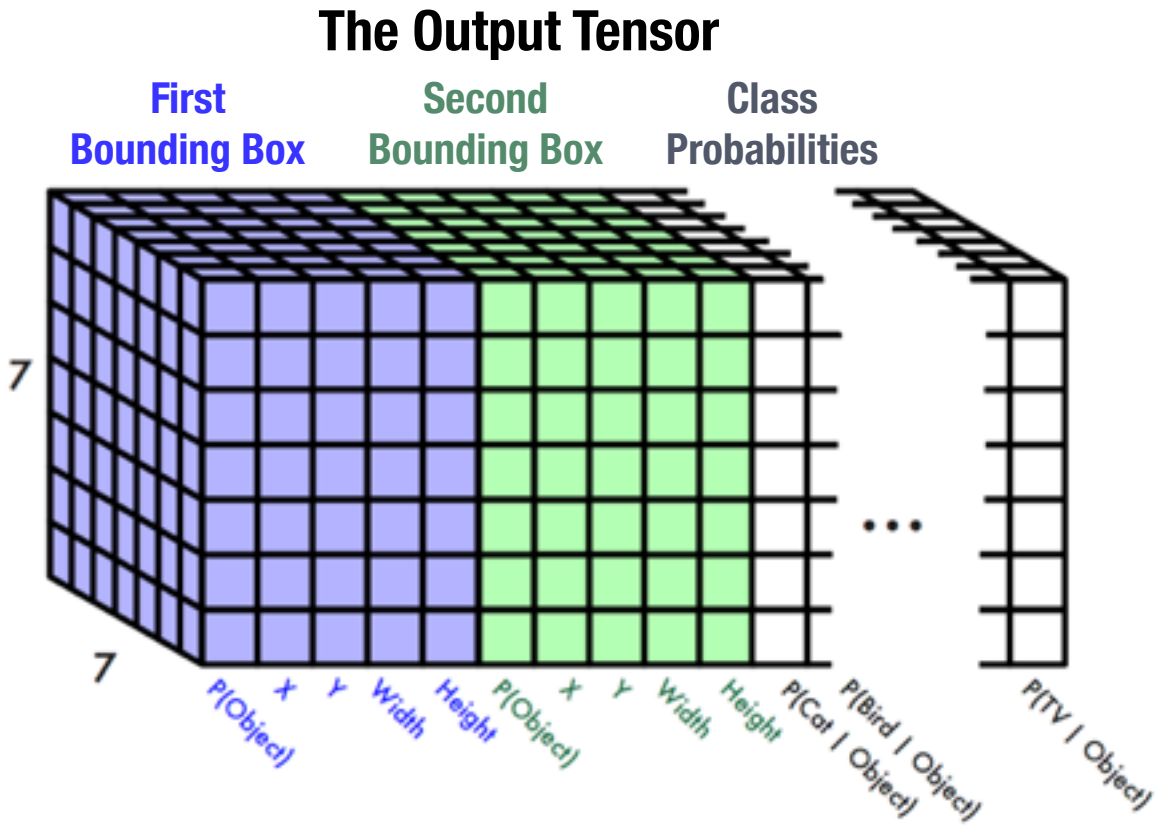




Redmon and Farhadi, YOLO9000: Better, Faster, Stronger, 2016, December 25 — *Merry Christmas?*

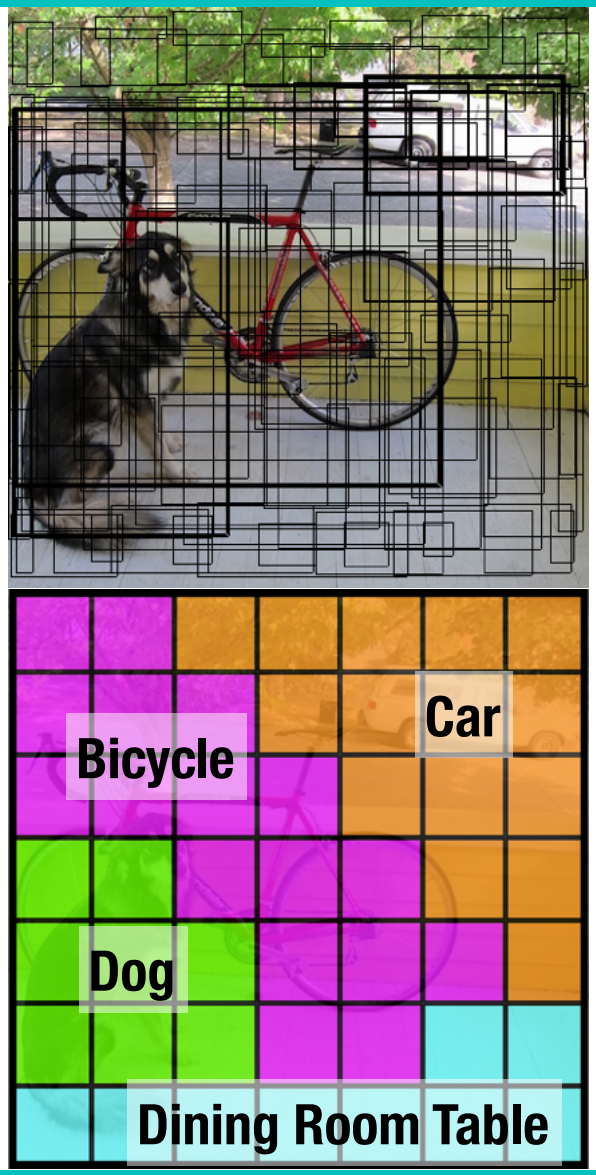
The YOLO Output Tensor



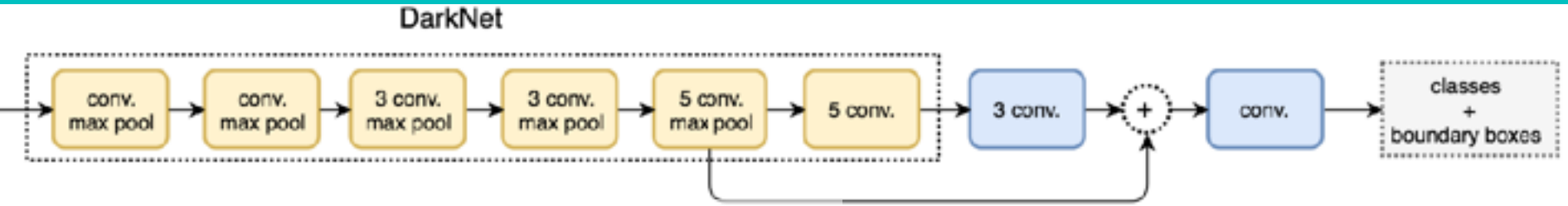


Redmon and Farhadi, YOLO9000: Better, Faster, Stronger, 2016, December 25 — *Merry Christmas?*

The YOLO Output Tensor

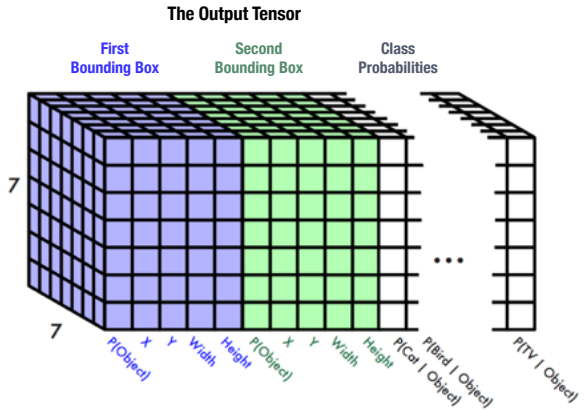


The YOLO Architecture



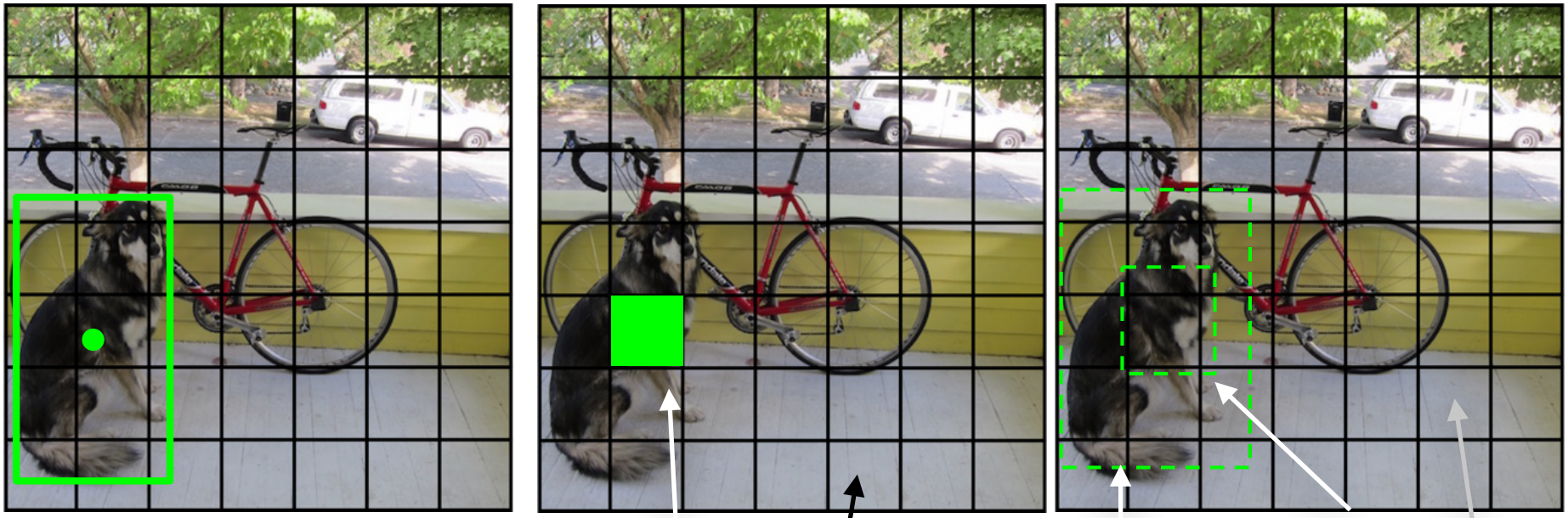
Trained from Traditional Image Dataset.
Architecture usually: DarkNet on ImageNet

Last layers:
Trained on images
with bounding boxes



https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088

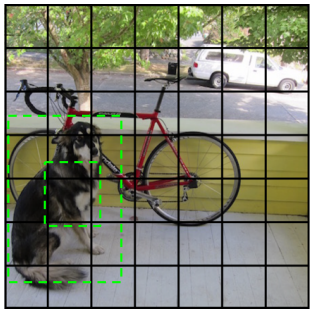
Training the YOLO Architecture



Redmon and Farhadi, YOLO9000: Better, Faster, Stronger, 2016, December 25 — *Merry Christmas?*

The YOLO Loss Function

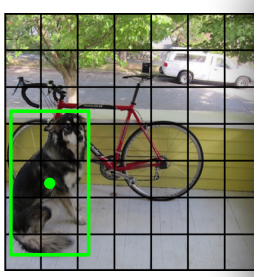
Update Bounding Box



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

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

Localization Loss



SxS cells, i^{th} cell

B boxes per cell, jth box

$\mathbb{1}^{\text{obj}}$ indicator function, from GT

$\hat{C}$ is confidence per box

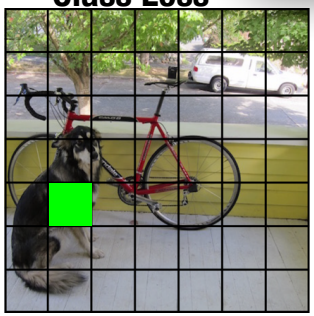
$\hat{p}(c)$ softmax output, per class

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

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

Object Detection Loss

Class Loss



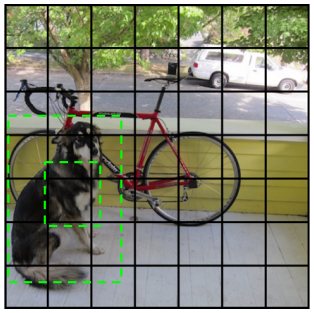
$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2$$

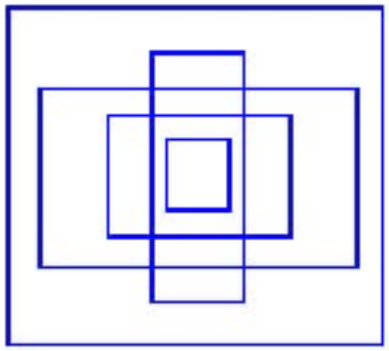
Classification Loss

https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088

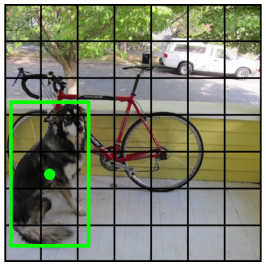
Updated YOLO Localization (v4)

Update Bounding Box

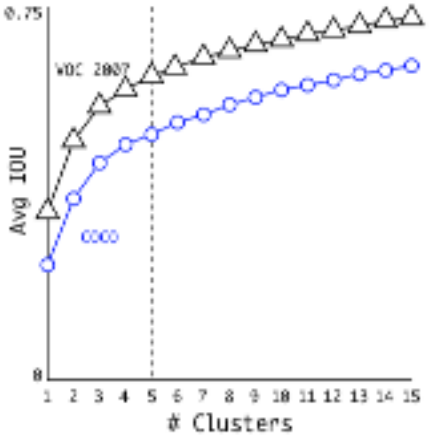


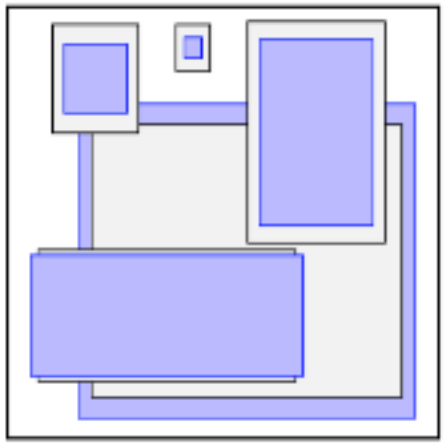


- Define 5 pre-defined box shapes (based on data)
- Regress x, y offset from cell center
- Bound x and y to the bounds of the cell
- And w and h scaling of predefined shape

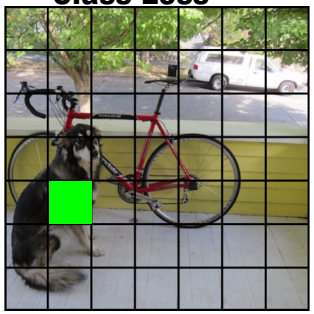


"Good" Shape Priors Found via Clustering





Class Loss



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

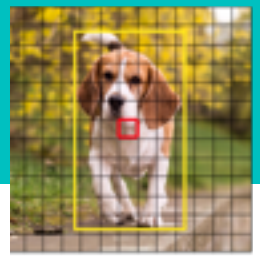
$$+\lambda_{\text{noobj}}\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{1}_{ij}^{\text{noobj}}\left(C_i-\hat{C}_{ij}\right)^2$$

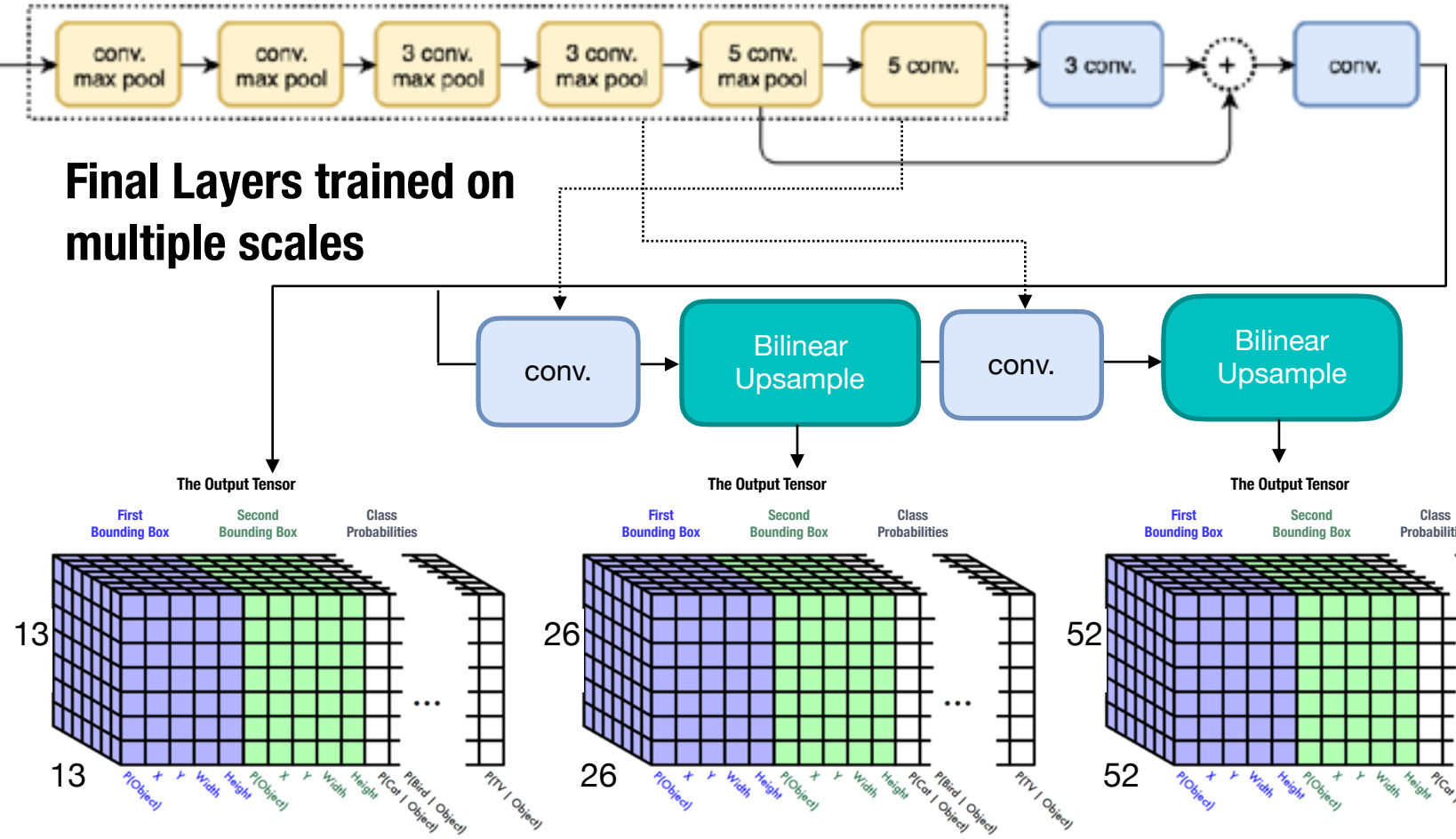
$$+\sum_{i=0}^{S^2}\mathbb{1}_{i}^{\text{obj}}\sum_{c\in\text{classes}}\left(p_i(c)-\hat{p}_i(c)\right)^2$$

Classification Loss

https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088

The YOLOv3 Architecture





All Losses Applied, 13x13 grid

All Losses Applied, 26x26 grid

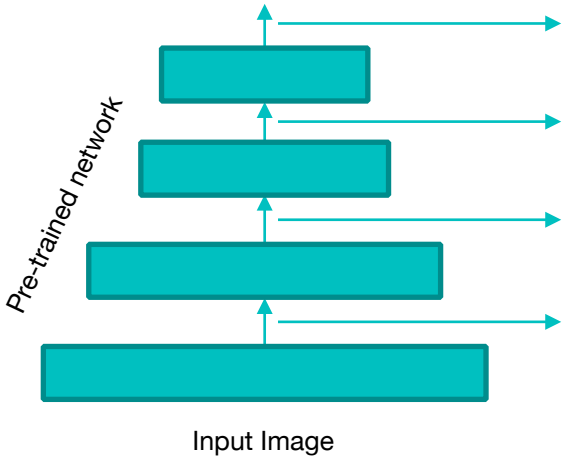
All Losses Applied, 52x52 grid

https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088

More YOLO Variants

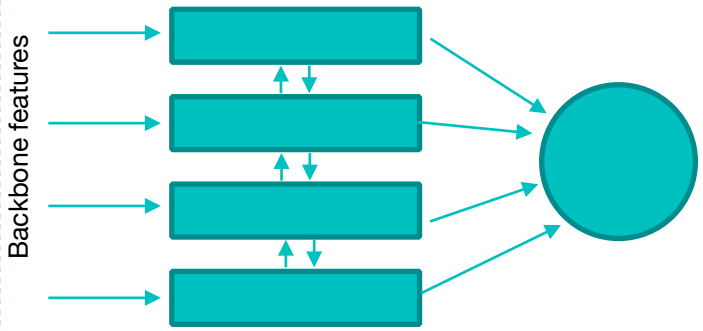
YOLO Structures, In General

Backbone

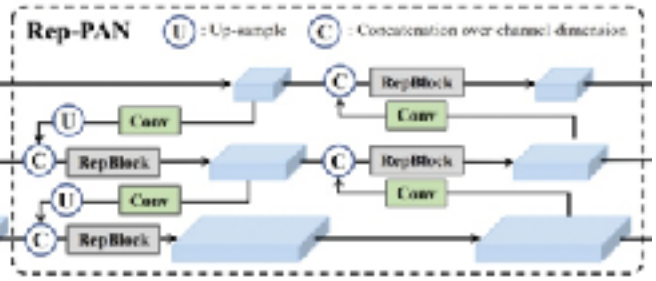


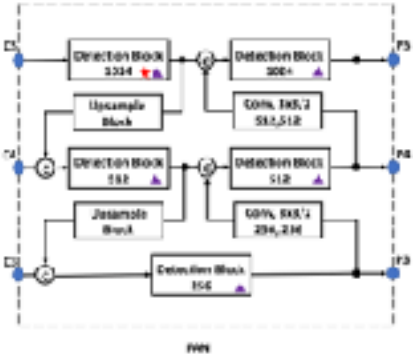
- YOLOv1-v4: DarkNet
- Yv5: EfficientNet
- Yv6: EfficientNet-L2
- PP-Yv1-v2: ResNet50-vd
- Yv6 v3.0: … 🙄 …

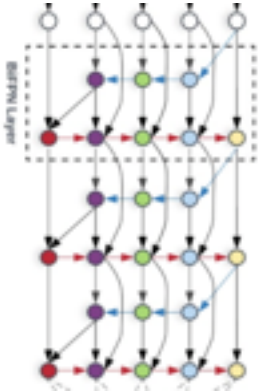
Neck



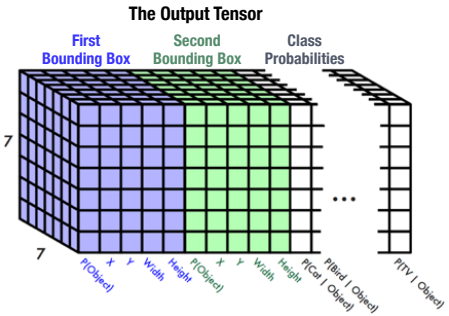
- Yv1: Transfer Layers
- Yv3: Upsampling layers
- Yv4: Cross Stage Partial layers (CSPNet). Use multiple layers from backbone with weighted concat
- PP-Y: Path Aggregation Network
- PP-Yv2: Multi-scale PAN



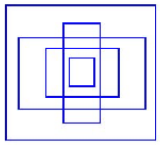




Head(s)

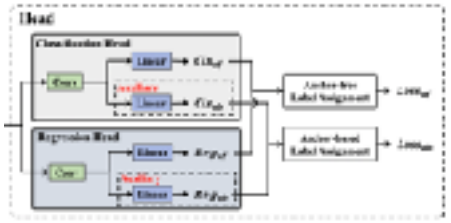


- Yv1: single head
- Yv3: Multi-resolution heads
- Yv4: Clustered anchor boxes and new gradient penalty
- PP-Yv2: IoU prediction loss
- Yv6 v3.0: Dense Anchor Boxes (anchors as aux. task)



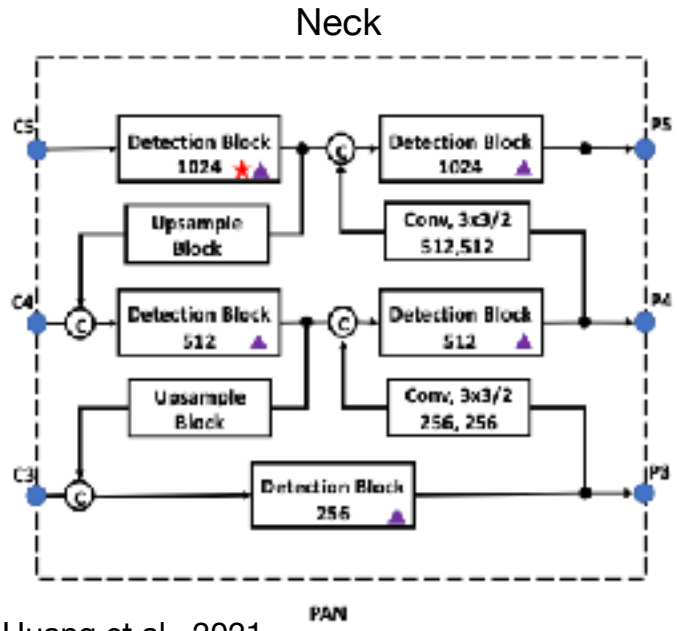
$$\mathcal{L}_{IoU} = \begin{matrix} -t_{IoU} \cdot \log(\sigma(p)) + \\ (t_{IoU} - 1) \cdot \log(1 - \sigma(p)) \end{matrix}$$

IoU Prediction

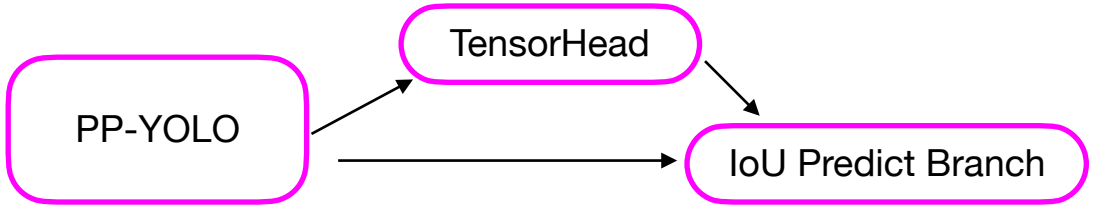


One Example: PP-YOLOv2

- Path Aggregation Network (PAN) for multi-scale processing
- Varied Input Sized Images and "Mish" activation
- "IoU Aware Branch" that uses the IoU prediction in the back propagation



Huang et al., 2021



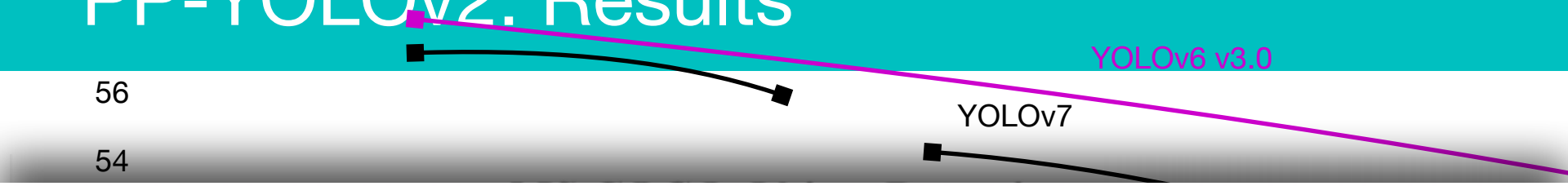
$$\mathcal{L}_{IoU} = -t_{IoU} \cdot \log(\sigma(p)) + (t_{IoU} - 1) \cdot \log(1 - \sigma(p))$$

IoU between output of YOLO BBox and GT

p=predicted IoU from branch

Main idea is to predict the IoU in a separate multi-task branch, which makes the network better at finding the correct bounding boxes

PP-YOLOv2: Results



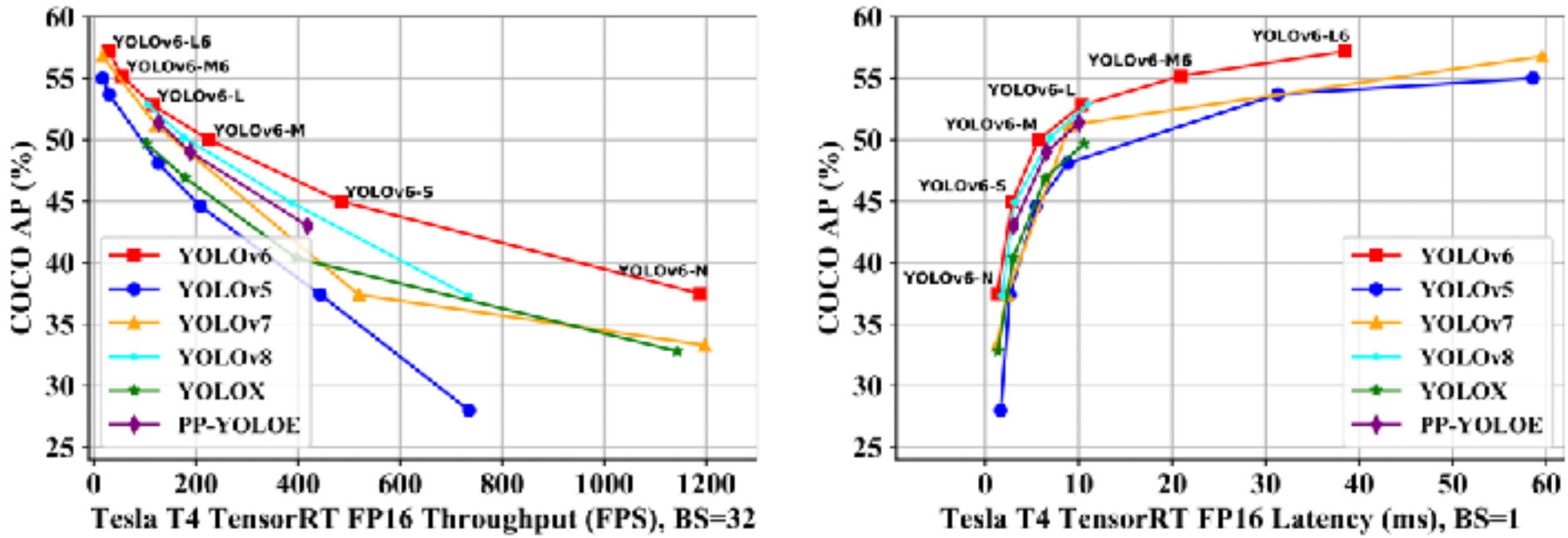
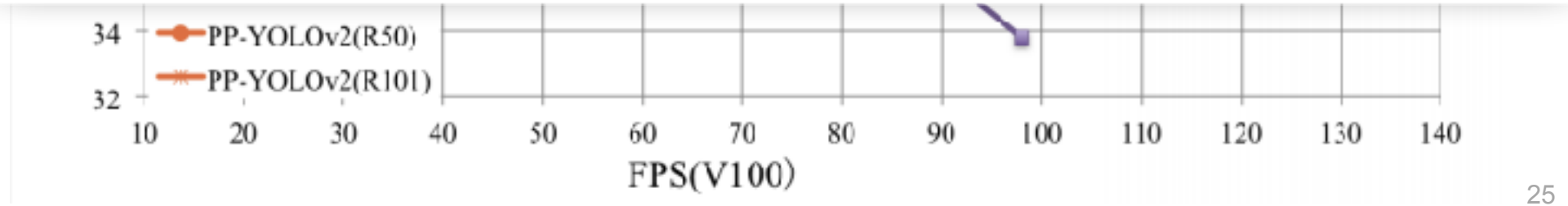


Figure 1: Comparison of state-of-the-art efficient object detectors. Both latency and throughput (at a batch size of 32) are given for a handy reference. All models are test with TensorRT 7.



A closing thought from YOLOv3 Report

Abstract

We present some updates to YOLO! We made a bunch of little design changes to make it better. We also trained this new network that's pretty swell. It's a little bigger than last time but more accurate. It's still fast though, don't worry. At 320 × 320 YOLOv3 runs in 22 ms at 28.2 mAP, as accurate as SSD but three times faster. When we look at the old .5 IOU mAP detection metric YOLOv3 is quite good. It achieves 57.9 AP_{50} in 51 ms on a Titan X, compared to 57.5 AP_{50} in 198 ms by RetinaNet, similar performance but 3.8× faster. As always, all the code is online at https://pjreddie.com/yolo/.

But maybe a better question is: "What are we going to do with these detectors now that we have them?" A lot of the people doing this research are at Google and Facebook. I guess at least we know the technology is in good hands and definitely won't be used to harvest your personal information and sell it to.... wait, you're saying that's exactly what it will be used for?? Oh.

Well the other people heavily funding vision research are the military and they've never done anything horrible like killing lots of people with new technology oh wait.....[1]

I have a lot of hope that most of the people using computer vision are just doing happy, good stuff with it, like counting the number of zebras in a national park [13], or tracking their cat as it wanders around their house [19]. But computer vision is already being put to questionable use and as researchers we have a responsibility to at least consider the harm our work might be doing and think of ways to mitigate it. We owe the world that much.

In closing, do not @ me. (Because I finally quit Twitter).

[1] The author is funded by the Office of Naval Research and Google.

Redmon and Farhadi, YOLOv3: An incremental Improvement

A closing thought from YOLOv3 Report

The Rebuttal I wish I could Write:

Reviewer #2 AKA Dan Grossman (lol blinding who does that) insists that I point out here that our graphs have not one but two non-zero origins. You're absolutely right Dan, that's because it looks way better than admitting to ourselves that we're all just here battling over 2-3% mAP. But here are the requested graphs. I threw in one with FPS too because we look just like super good when we plot on FPS.

Reviewer #4 AKA JudasAdventus on Reddit writes "Entertaining read but the arguments against the MSCOCO metrics seem a bit weak". Well, I always knew you would be the one to turn on me Judas. You know how when you work on a project and it only comes out alright so you have to figure out some way to justify how what you did actually was pretty cool? I was basically trying to do that and I lashed out at the COCO metrics a little bit. But now that I've staked out this hill I may as well die on it.

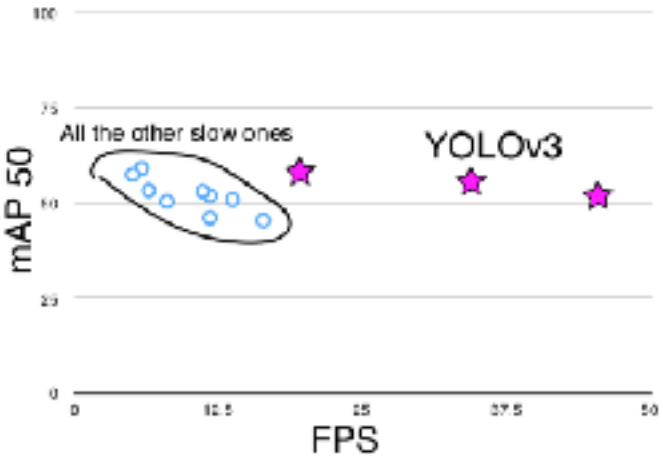


Figure 5. These two hypothetical detectors are perfect according to mAP over these two images. They are both perfect. Totally equal.

Now this is OBVIOUSLY an over-exaggeration of the problems with mAP but I guess my newly reteconned point is that there are such obvious discrepancies between what people in the "real world" would care about and our current metrics that I think if we're going to come up with new metrics we should focus on these discrepancies. Also, like, it's already mean average precision, what do we even call the COCO metric, average mean average precision?

Here's a proposal, what people actually care about is given an image and a detector, how well will the detector find and classify objects in the image. What about getting rid of the per-class AP and just doing a global average precision? Or doing an AP calcu-

Boxes are stupid anyway though, I'm probably a true believer in masks except I can't get YOLO to learn them.

YOLACT, one pass mask generation

YOLACT
Real-time Instance Segmentation

Daniel Bolya Chong Zhou Fanyi Xiao Yong Jae Lee
University of California, Davis
{dbolya, czhou, fyxiao, yongjaelee}@ucdavis.edu

Abstract

*We present a simple, fully-convolutional model for real-time instance segmentation that achieves 29.8 mAP on MS COCO at 33.5 fps evaluated on a single Titan Xp, which is significantly faster than any previous competitive approach. Moreover, we obtain this result after training on **only one GPU**. We accomplish this by breaking instance segmentation into two parallel subtasks: (1) generating a set of prototype masks and (2) predicting per-instance mask coefficients. Then we produce instance masks by linearly combining the prototypes with the mask coefficients. We find that because this process doesn't depend on repooling, this approach produces very high-quality masks and exhibits temporal stability for free. Furthermore, we analyze the emergent behavior of our prototypes and show they learn to localize instances on their own in a translation variant manner, despite being fully-convolutional. Finally, we also propose Fast NMS, a drop-in 12 ms faster replacement for standard NMS that only has a marginal performance penalty.*

1. Introduction

"Boxes are stupid anyway though, I'm probably a true believer in masks except I can't get YOLO to learn them."
– Joseph Redmon, YOLOv3 [36]

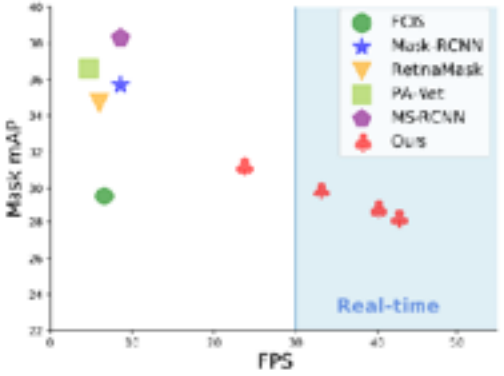


Figure 1: Speed-performance trade-off for various instance segmentation methods on COCO. To our knowledge, ours is the first *real-time* (above 30 FPS) approach with around 30 mask mAP on COCO `test-dev`.

However, instance segmentation is hard—much harder than object detection. One-stage object detectors like SSD and YOLO are able to speed up existing two-stage detectors like Faster R-CNN by simply removing the second stage and making up for the lost performance in other ways. The same approach is not easily extendable, however, to instance segmentation. State-of-the-art two-stage

Bolya, Daniel, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. "YOLACT: real-time instance segmentation." In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 9157-9166. October 2019.

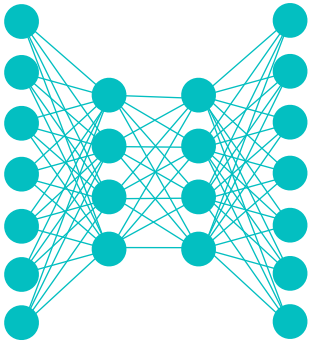
Lecture Notes for

Neural Networks and Machine Learning

FCN Learning: Detection

Next Time:
Instance Segmentation

Reading: None



Backup slides

Title Between Topics

Example Slide

Title

Subtitle

`Follow Along: Notebook Name`