Practical Deep learning for Computer Vision

Dr Kien Nguyen
Email: k.nguyenthanh@qut.edu.au
Queensland University of Technology, Australia

Warning

Deep learning is new SEXY (for a reason!)

Week 3: Object Recognition

- Resnet from scratch
- Transfer Learning
- Google Colab

Week 4: Object Detection

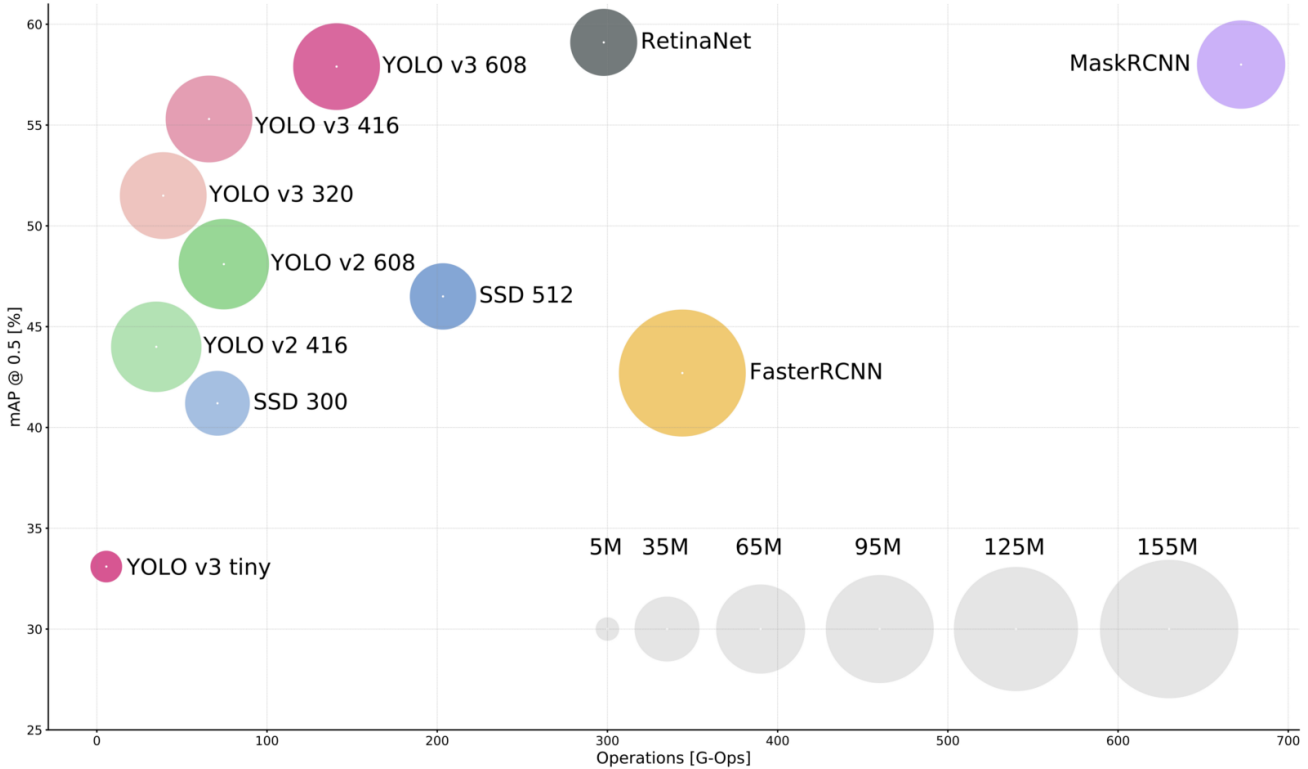
- DarkNet & YOLO
- Transfer Learning
- AWS

Week 5: GAN

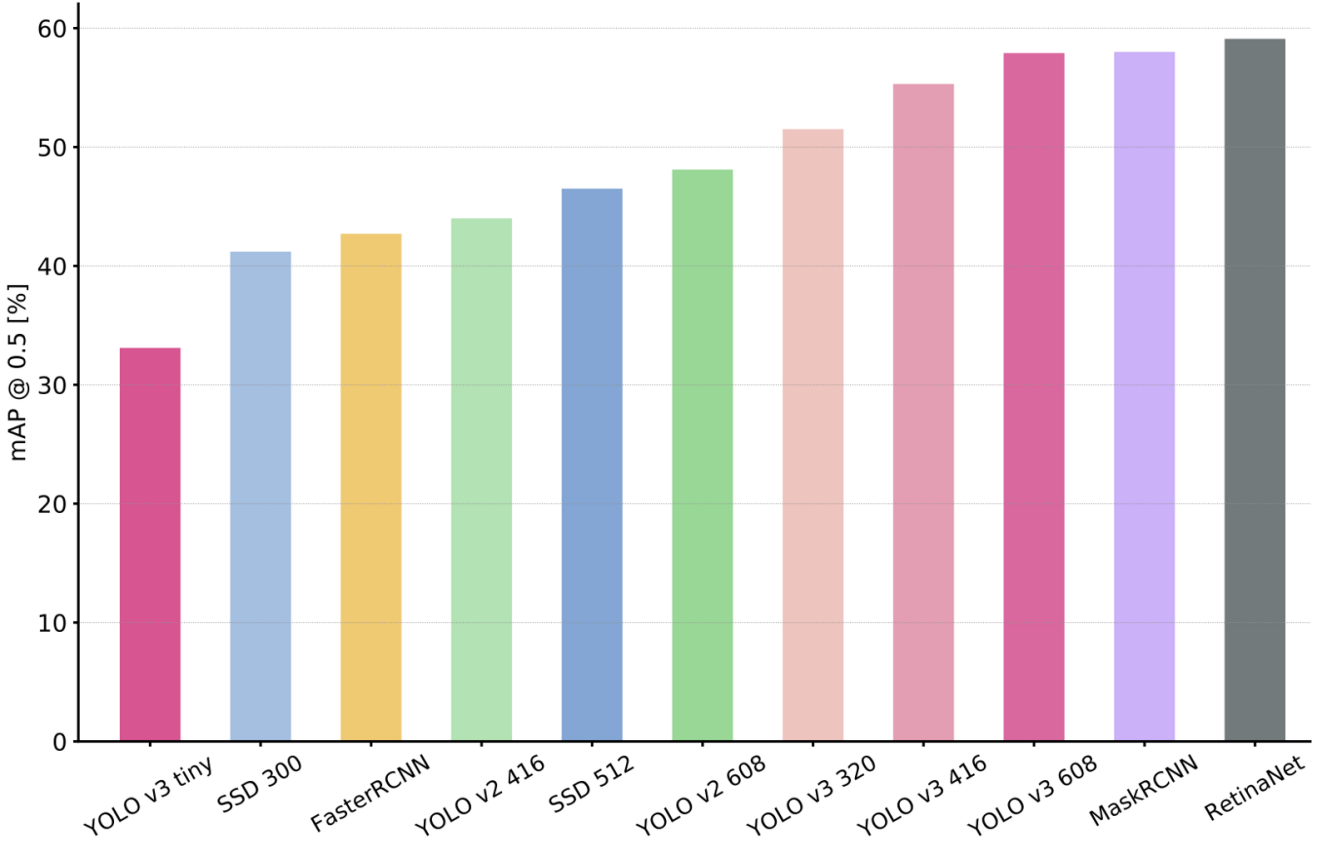
- SSD
- GAN

Week 4. Darknet & YOLO

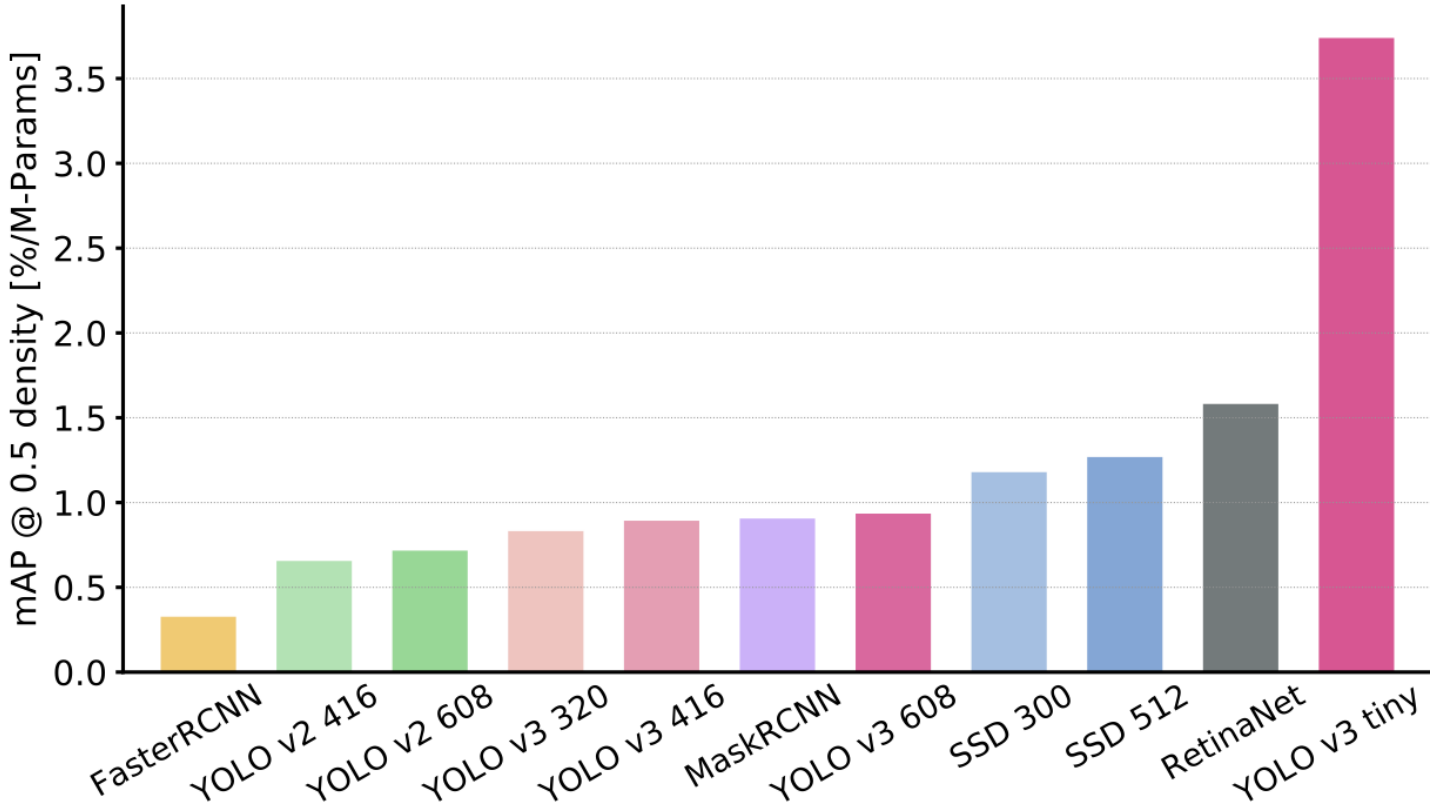
Modern object detector landscape @ MSCOCO 2017 validation set



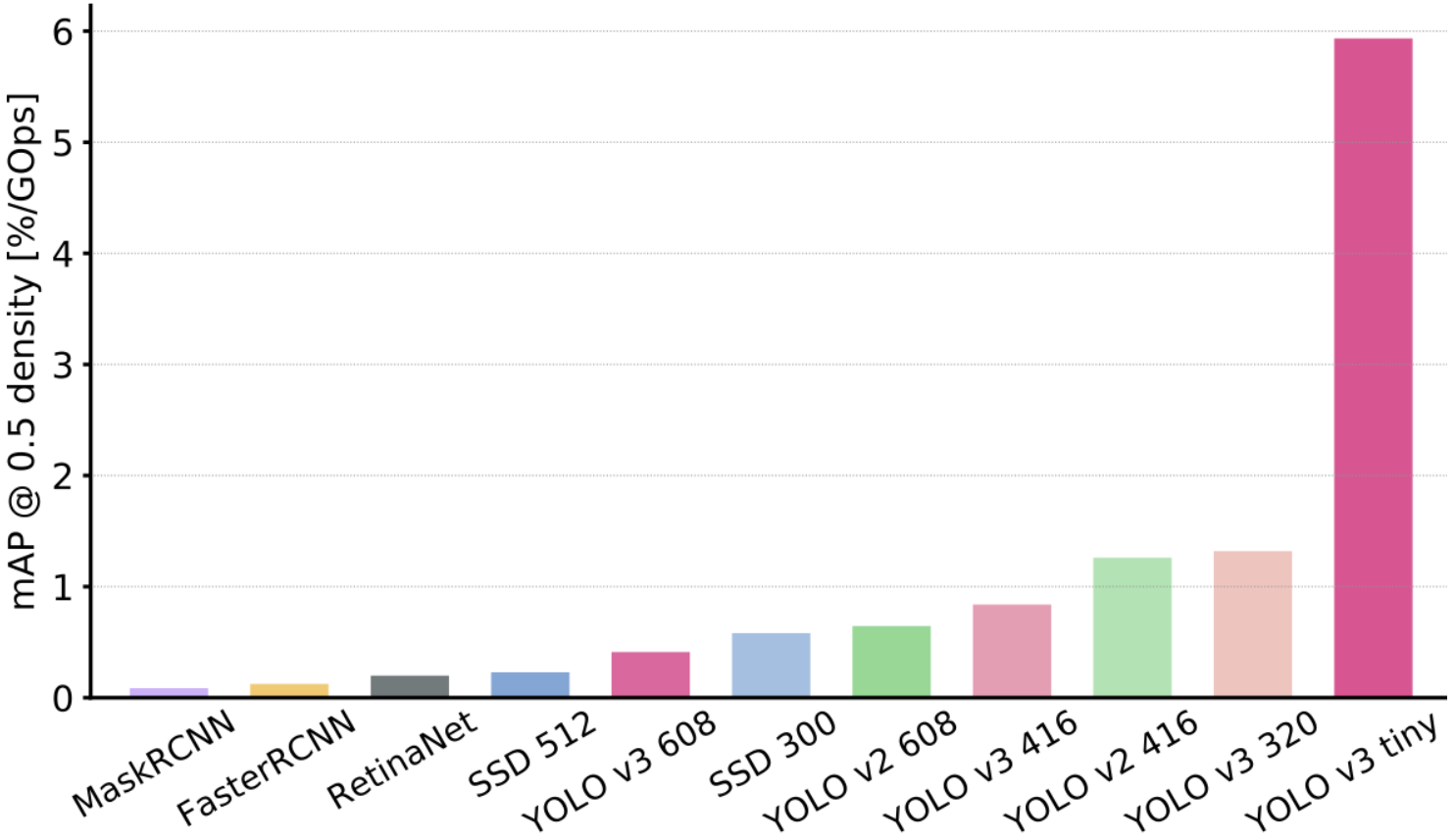
mAP@0.5 per Million Parameters vs GOps, MParam



mAP@0.5 vs Network



mAP@0.5 per Million Parameters vs Network

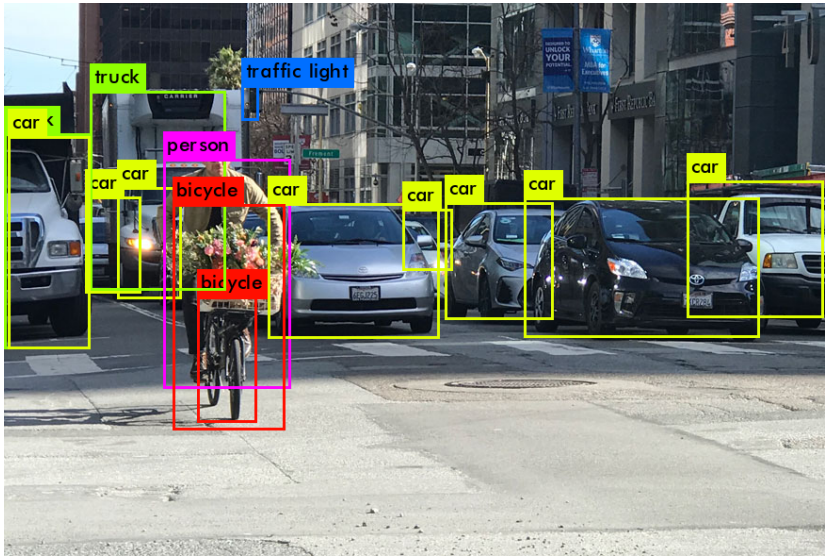


mAP@0.5 per GOps vs Network

Modern object detectors

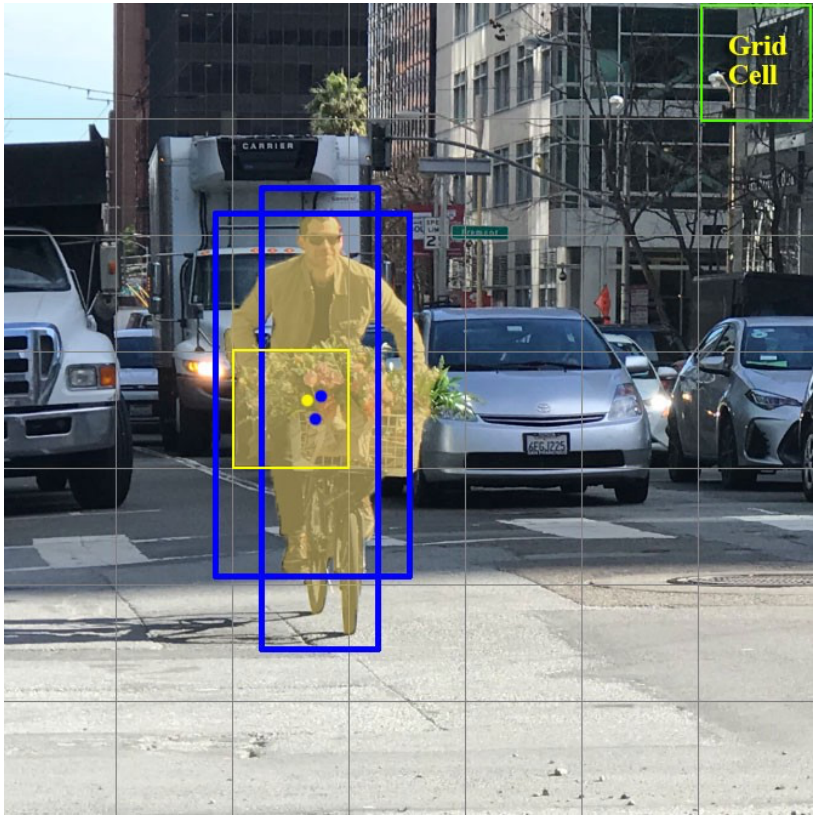
- https://zsc.github.io/megvii-pku-dl-course/slides/Lecture6(Object%20Detection).pdf

YOLO - How it works



You Only Look Once: Unified, Real-Time Object Detection – CVPR 2016

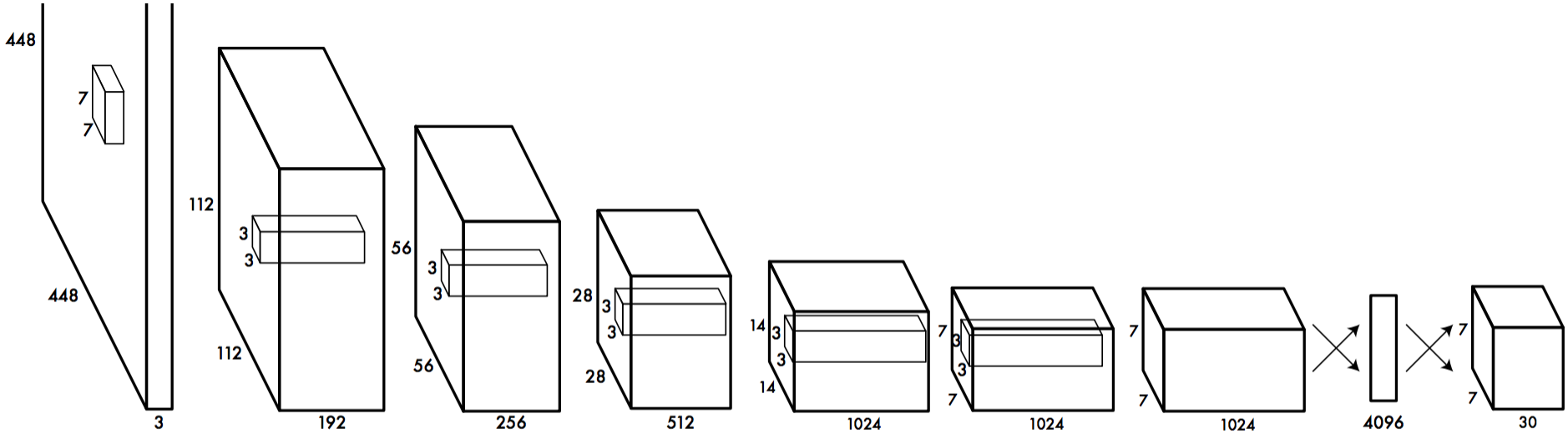
YOLO - How it works



- Divide image into grid: 7 x 7
- Within each grid cell:
  - Predict **one** object, with potential two different base boxes: B = 2
  - Regress from each of the B base boxes to a final box with 5 numbers:
    (x, y, h, w, box confidence score)
  - Predict scores for each of C = 20 classes
- Output: 7 x 7 x (5 * B + C)

$$7x7x(5*2 + 20) = 7x7x30$$

DarkNet-19



Conv. Layer
7x7x64-s-2
Maxpool Layer
2x2-s-2

Conv. Layer
3x3x192
Maxpool Layer
2x2-s-2

Conv. Layers
1x1x128
3x3x256
1x1x256
3x3x512
Maxpool Layer
2x2-s-2

Conv. Layers
1x1x256
3x3x512
} ×4
1x1x512
3x3x1024
Maxpool Layer
2x2-s-2

Conv. Layers
1x1x512
3x3x1024
} ×2
3x3x1024
3x3x1024-s-2

Conv. Layers
3x3x1024
3x3x1024

Conn. Layer

Conn. Layer

Loss function

- YOLO uses sum-squared error between the predictions and the ground truth to calculate loss. The loss function composes of:
  - the **classification loss**.
  - the **localization loss** (errors between the predicted boundary box and the ground truth).
  - the **confidence loss** (the objectness of the box).

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

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

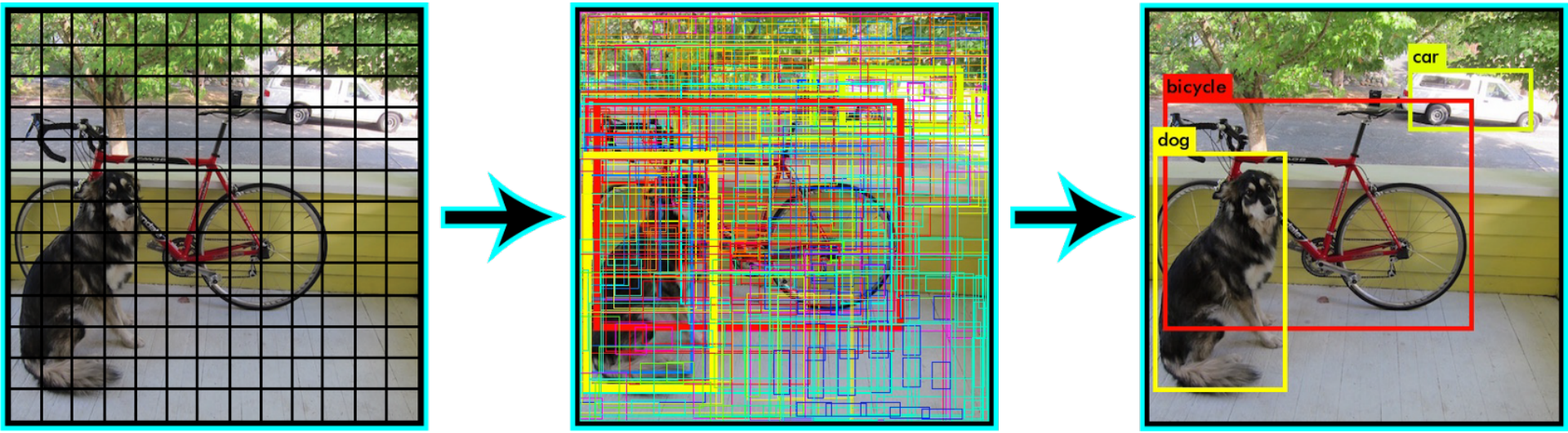
$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

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

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

YOLO - How it works

- To make a final prediction, we keep those with high box confidence scores (greater than 0.25) as our final predictions (the right picture).



class confidence score = box confidence score × conditional class probability

YOLO advantages

- Fast. Good for real-time processing.
- Predictions (object locations and classes) are made from one single network. Can be trained end-to-end to improve accuracy.
- YOLO is more generalized. It outperforms other methods when generalizing from natural images to other domains like artwork

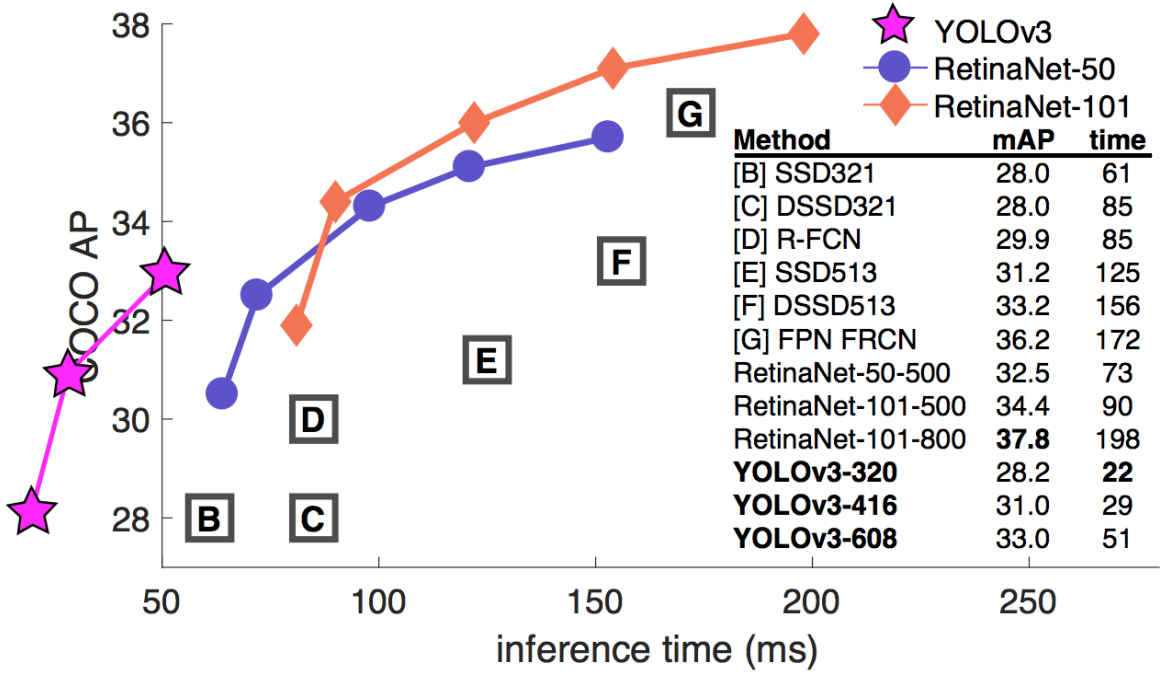
YOLO9000 (v2)

	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**

YOLO9000: Better, Faster, Stronger – CVPR 2017

YOLOv3

- New DarkNet-53
- Feature Pyramid Networks (FPN) like Feature Pyramid
- New loss function



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

YOLOv3: An Incremental Improvement - 2018

https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b

DarkNet-19

Type	Filters	Size/Stride	Output
Convolutional	32	3 × 3	224 × 224
Maxpool		2 × 2/2	112 × 112
Convolutional	64	3 × 3	112 × 112
Maxpool		2 × 2/2	56 × 56
Convolutional	128	3 × 3	56 × 56
Convolutional	64	1 × 1	56 × 56
Convolutional	128	3 × 3	56 × 56
Maxpool		2 × 2/2	28 × 28
Convolutional	256	3 × 3	28 × 28
Convolutional	128	1 × 1	28 × 28
Convolutional	256	3 × 3	28 × 28
Maxpool		2 × 2/2	14 × 14
Convolutional	512	3 × 3	14 × 14
Convolutional	256	1 × 1	14 × 14
Convolutional	512	3 × 3	14 × 14
Convolutional	256	1 × 1	14 × 14
Convolutional	512	3 × 3	14 × 14
Maxpool		2 × 2/2	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	512	1 × 1	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	512	1 × 1	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	1000	1 × 1	7 × 7
Avgpool		Global	1000
Softmax			

DarkNet-53

	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			

YOLO - How it works

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

- Implementations from scratch:
  - Tensorflow: https://itnext.io/implementing-yolo-v3-in-tensorflow-tf-slim-c3c55ff59dbe
  - Pytorch: https://www.kdnuggets.com/2018/05/implement-yolo-v3-object-detector-pytorch-part-1.html

Demo

- Detection Using A Pre-Trained Model with python
  - Detect objects with a pre-trained YOLOv3 on the COCO dataset (80classes)

- Detection Using A Pre-Trained Model with executable
  - https://pjreddie.com/darknet/yolo/

Transfer learning with YOLO – Detect your own objects

- Detect a single object YOLOv2
  - https://medium.com/@manivannan_data/how-to-train-yolov2-to-detect-custom-objects-9010df784f36
- Detect multiple objects YOLOv2
  - https://medium.com/@manivannan_data/how-to-train-multiple-objects-in-yolov2-using-your-own-dataset-2b4fee898f17
- Detect multiple objects with YOLOv3
  - https://medium.com/@manivannan_data/how-to-train-yolov3-to-detect-custom-objects-ccbcafeb13d2

Transfer learning with YOLO – Detect your own objects

- Detect a single object YOLOv2
  - https://medium.com/@manivannan_data/how-to-train-yolov2-to-detect-custom-objects-9010df784f36
- Detect multiple objects YOLOv2
  - https://medium.com/@manivannan_data/how-to-train-multiple-objects-in-yolov2-using-your-own-dataset-2b4fee898f17
- Detect multiple objects with YOLOv3
  - https://medium.com/@manivannan_data/how-to-train-yolov3-to-detect-custom-objects-ccbcafeb13d2

YOLOv3 on Google Colab

- Detect objects with a pre-trained YOLOv3 on the COCO dataset (80classes)
- OpenCV

- All my tutorial codes are here:
    - https://colab.research.google.com/drive/1mpOqyC9bSM6Gup5M9lBkB16I1yo_V7Db