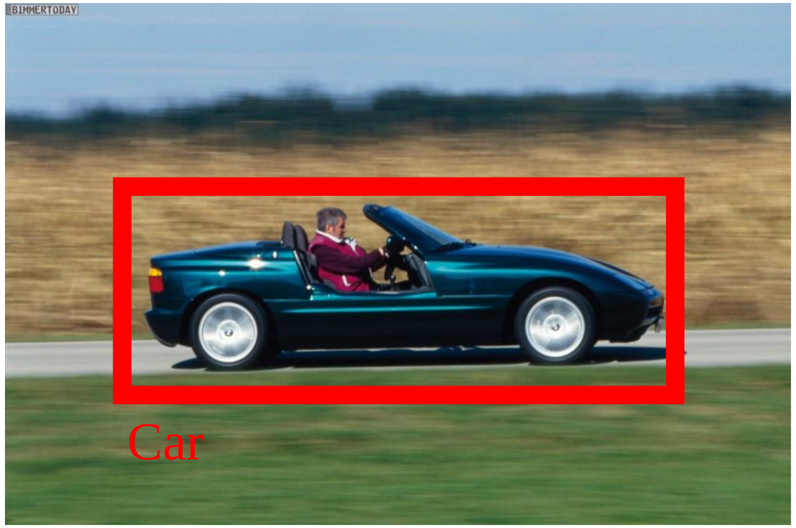
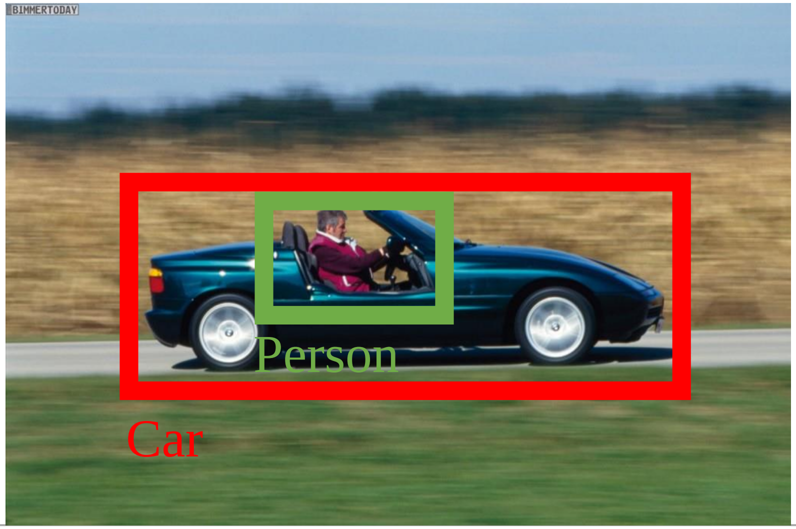
Detection as Regression

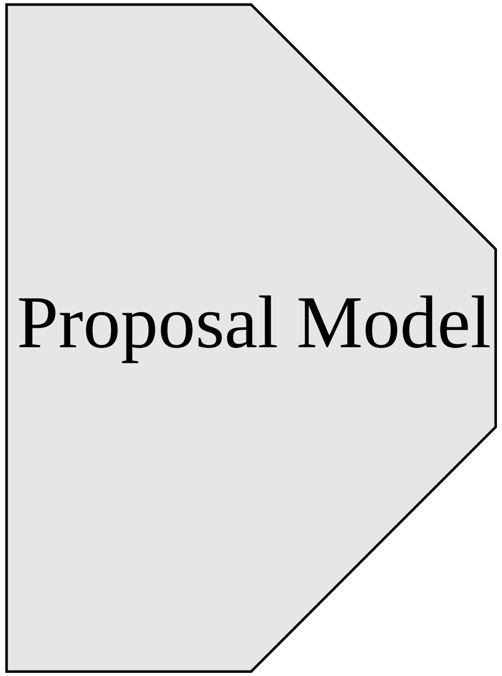
wellyzhangc@gmail.com

1 Detection Problem

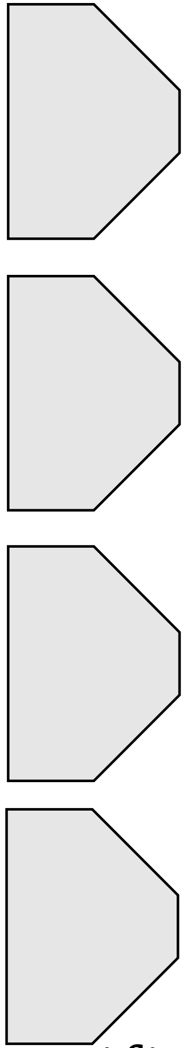




2 Previously: Proposals + Classification



2 Previously: Proposals + Classification



Background

Dog

Table

Person

Classifier

wellyzhangc@gmail.com

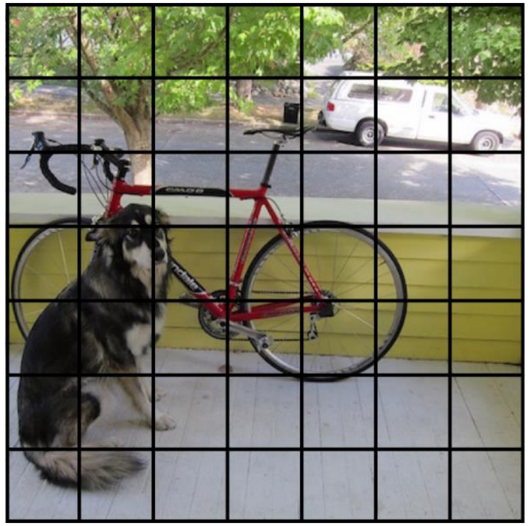
2.1 Problems

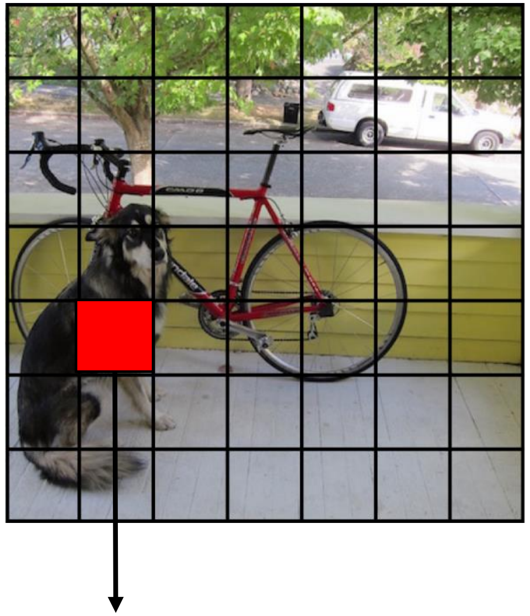
- People use large-scale neural networks for both the proposal model and classifier.
- Two passes of feed-forward.
- High accuracy but far from real-time.
- BING is fast for object proposal but people brag about end-to-end. BING's detection performance on VOC not found.

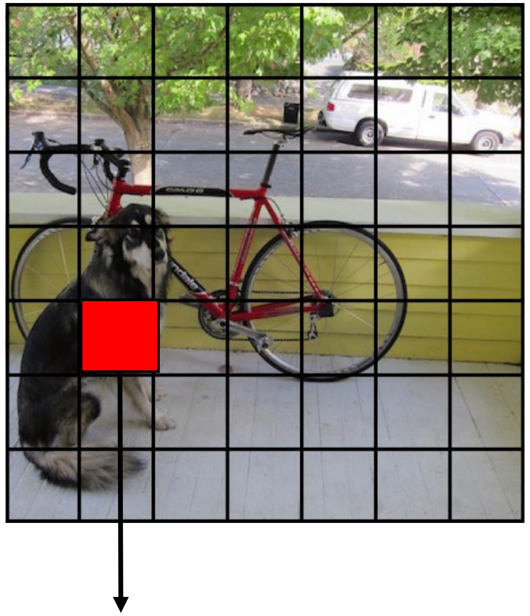
3 Detection as Regression

- Can we only use one pass of the network?
- What about regressing out the object bounding boxes together with its class distribution?
- Typical models: YOLO and SSD.

3.1 YOLO







{P(object}, x, y, w, h}xB

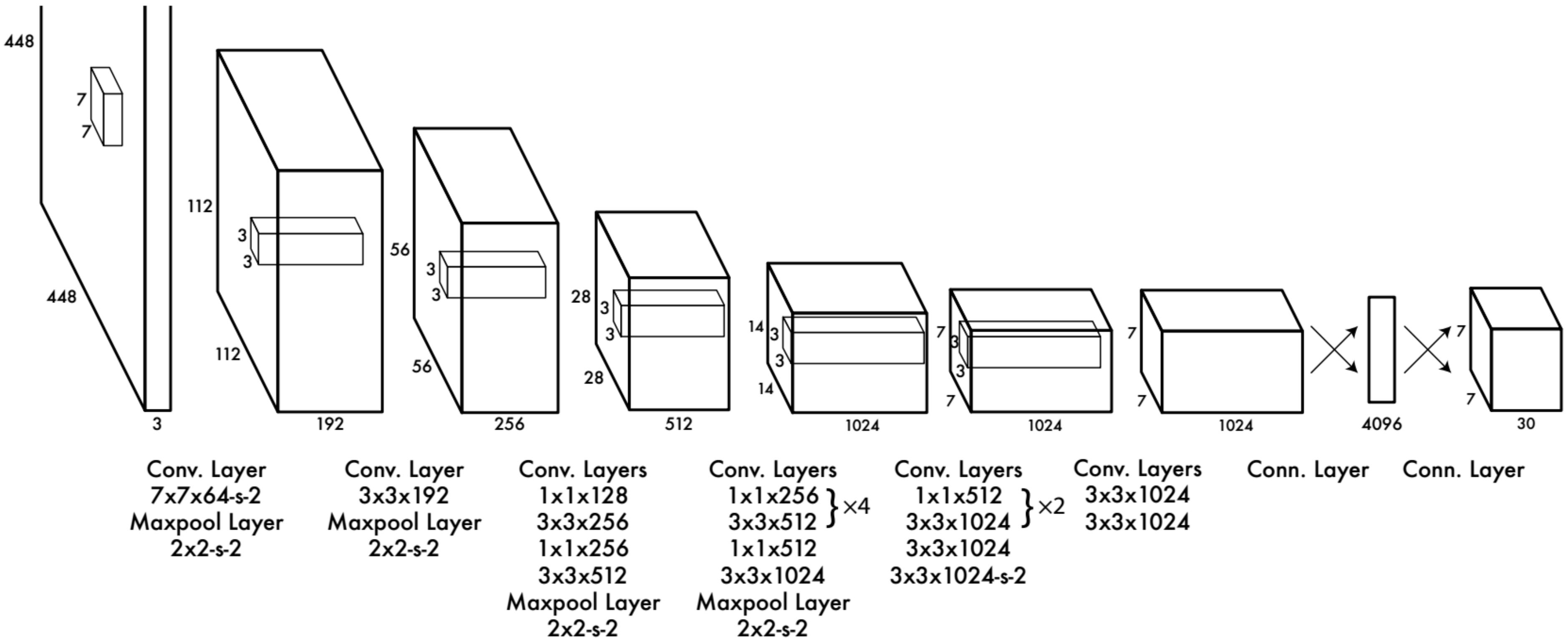
{P(car|object}, P(dog|object)...}

Split the image into an SxS grid

Each cell predicts B boxes, each with a confidence score and 4 coordinates

Each cell also predicts a conditional class distribution

3.1 YOLO



Network architecture: SxSx(classes + Bx5) outputs, use LeakyReLU

3.1 YOLO

$$
\begin{aligned}
&\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \\
&+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2 \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i\right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i\right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2
\end{aligned}
$$

Loss function: square root to balance effects for large and small bounding boxes

3.1 YOLO

- Ground truth coordinates computed by normalizing pictures to have unit width and height; cells that the centers of objects' bounding boxes fall into are responsible for detection.
- During inference, use Non-Maximum Suppression and threshold detections.

3.1 YOLO

- Fast enough but less accurate.
- Does not handle objects well that are either small or fall into the same cell.

Real-Time Detectors	Train	mAP	FPS
100Hz DPM [30]	2007	16.0	100
30Hz DPM [30]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [37]	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[27]	2007+2012	73.2	7
Faster R-CNN ZF [27]	2007+2012	62.1	18

3.1.1 Experience with YOLO

- Licence plate detection





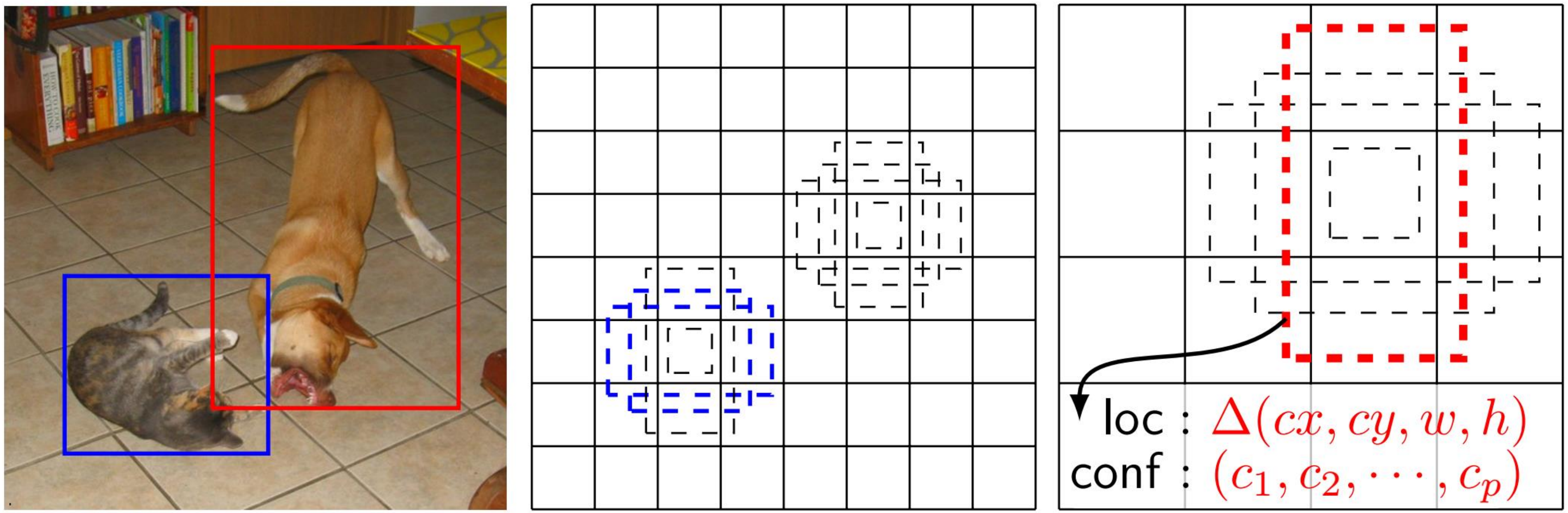

Model	mAP@0.5	No.Conv	Test Time	Init
YOLO	97.7	9	0.022s	No
Faster R-CNN	45.0	5	0.077s	Yes

wellyzhangc@gmail.com

3.2 SSD

- Also treat detection as regression
- Leverage multi-scale feature maps
- Borrow from YOLO the idea of image grid and anchor box and parameterization method from Faster R-CNN

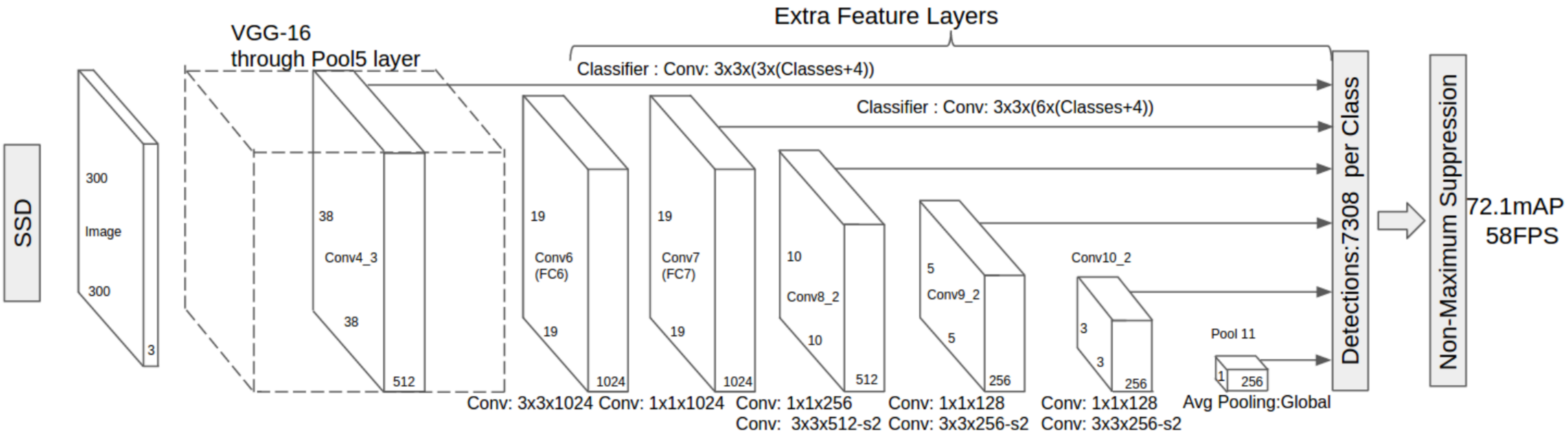
3.2 SSD



(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

Each element in the feature map predicts for each anchor box, the 4 offsets and the class distribution in a convolutional manner; anchor box acts like prior

3.2 SSD



Network architecture: fully convolutional

3.2 SSD

$$L(x, c, l, g) = \frac{1}{N}(L_{conf}(x, c) + \alpha L_{loc}(x, l, g))$$

Loss function: x is the image, c the class, l the anchor box and g the ground truth

wellyzhangc@gmail.com

3.2 SSD

$$\hat{g}_j^{cx} = (g_j^{cx} - d_i^{cx})/d_i^w \qquad \hat{g}_j^{cy} = (g_j^{cy} - d_i^{cy})/d_i^h$$

$$\hat{g}_j^w = \log\left(\frac{g_j^w}{d_i^w}\right) \qquad \hat{g}_j^h = \log\left(\frac{g_j^h}{d_i^h}\right)$$

Parameterization method: g for ground truth and d for anchor box

3.2 SSD

- Match the anchor box and the ground truth by IoU
- Do NMS to select detections in the end
- Use a selection method similar to Faster R-CNN during inference

Method	mAP	FPS	batch size
Faster R-CNN (VGG16)	73.2	7	1
Fast YOLO	52.7	155	1
YOLO (VGG16)	66.4	21	1
SSD300	74.3	46	1
SSD512	76.8	19	1
SSD300	74.3	59	8
SSD512	76.8	22	8

3.2 SSD

- Multi-scale feature maps help

Prediction source layers from:						mAP use boundary boxes?		# Boxes
conv4_3	conv7	conv8_2	conv9_2	conv10_2	conv11_2	Yes	No	
✔	✔	✔	✔	✔	✔	74.3	63.4	8732
✔	✔	✔	✔	✔		**74.6**	63.1	8764
✔	✔	✔	✔			73.8	68.4	8942
✔	✔	✔				70.7	69.2	9864
✔	✔					64.2	64.4	9025
	✔					62.4	64.0	8664

3.2 SSD

- Fun fact: performance boost comes from data augmentation

	SSD300				
more data augmentation?		✔	✔	✔	✔
include $\{\frac{1}{2}, 2\}$ box?	✔		✔	✔	✔
include $\{\frac{1}{3}, 3\}$ box?	✔			✔	✔
use atrous?	✔	✔	✔		✔
VOC2007 `test` mAP	65.5	71.6	73.7	74.2	**74.3**

3.3 YOLOv2

- Fully-convolutional model
- Adjust priors on bounding boxes instead of predicting the width and height outright; still predict coordinates
- Code released but paper not yet
- Project page: http://pjreddie.com/darknet/yolo/
- Code: https://github.com/pjreddie/darknet

3.3 YOLOv2

Model	Train	Test	mAP	FLOPS	FPS
Old YOLO	VOC 2007+2012	2007	63.4	40.19 Bn	45
SSD300	VOC 2007+2012	2007	74.3	-	46
SSD500	VOC 2007+2012	2007	76.8	-	19
YOLOv2	VOC 2007+2012	2007	76.8	34.90 Bn	67
YOLOv2 544x544	VOC 2007+2012	2007	78.6	59.68 Bn	40
Tiny YOLO	VOC 2007+2012	2007	57.1	6.97 Bn	207
SSD300	COCO trainval	test-dev	41.2	-	46
SSD500	COCO trainval	test-dev	46.5	-	19
YOLOv2 544x544	COCO trainval	test-dev	44.0	59.68 Bn	40

Thx!

wellyzhangc@gmail.com