Machine Learning Tokyo __init__

SSD

Charles Melby-Thompson

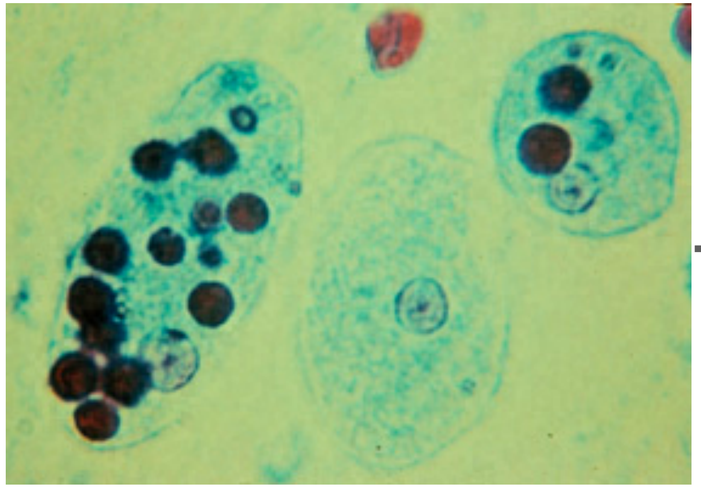
Machine Learning Tokyo __init__

Single Shot Multibox Detector

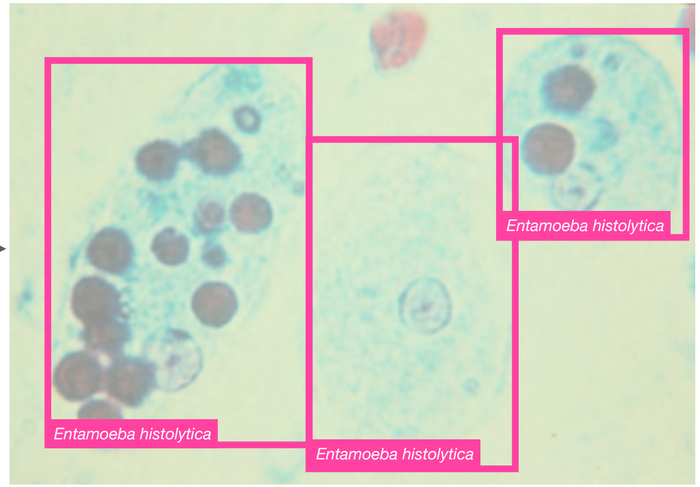
Charles Melby-Thompson

Object Detector

Input: Image



Output: Boxes and classes

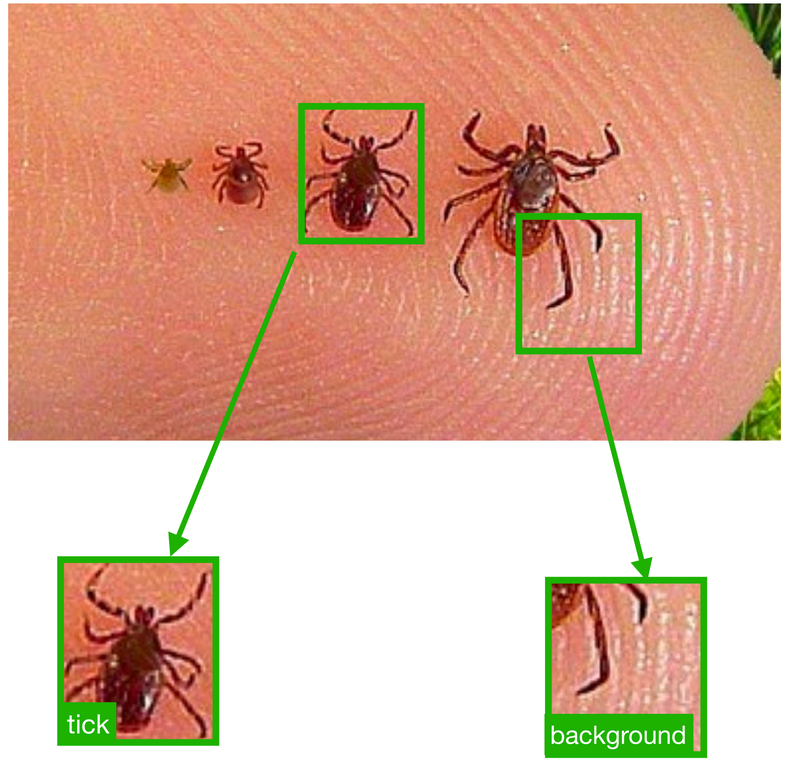


Object Detector

Approach #1

Approach as classification problem:

- sample boxes from the image
- apply image classifier → `background`, `tick`, ...
- non-background boxes are detections



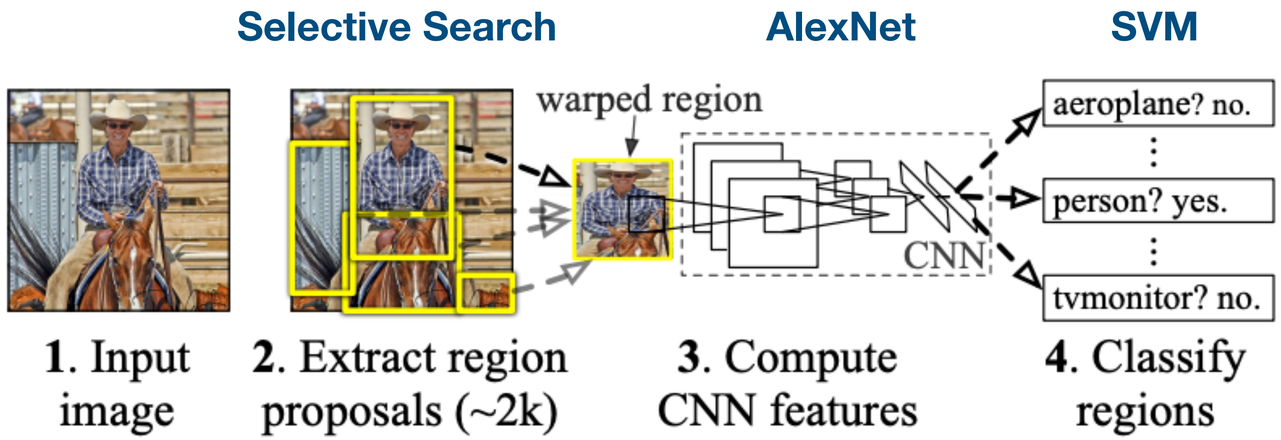
How to sample boxes?

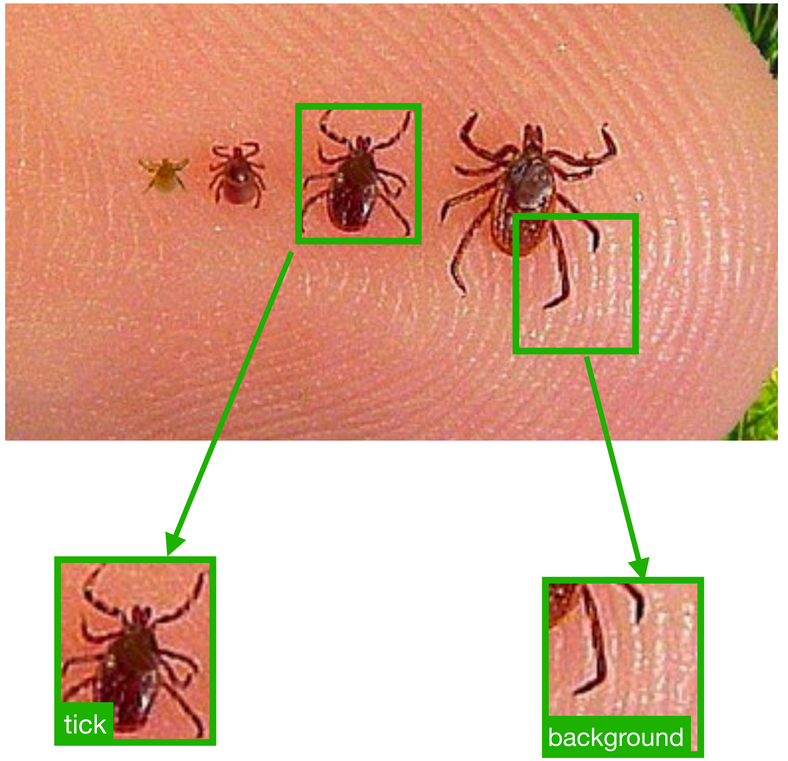
1. sliding window — expensive!
2. region proposal

Object Detector

Approach #1: R-CNN

R-CNN (arXiv: 1311.2524)





Key disadvantages:

1. 3 independently trained components (not end-to-end)
2. Slow!

Object Detector

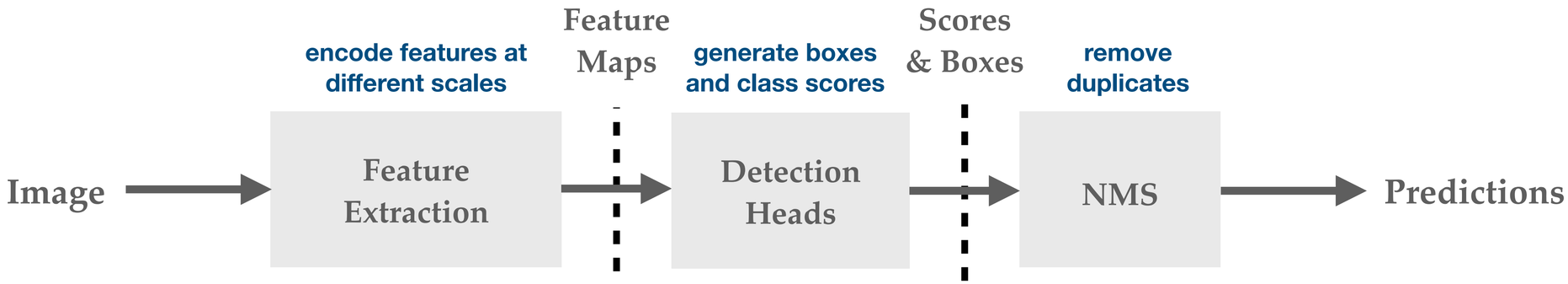
Alternatives

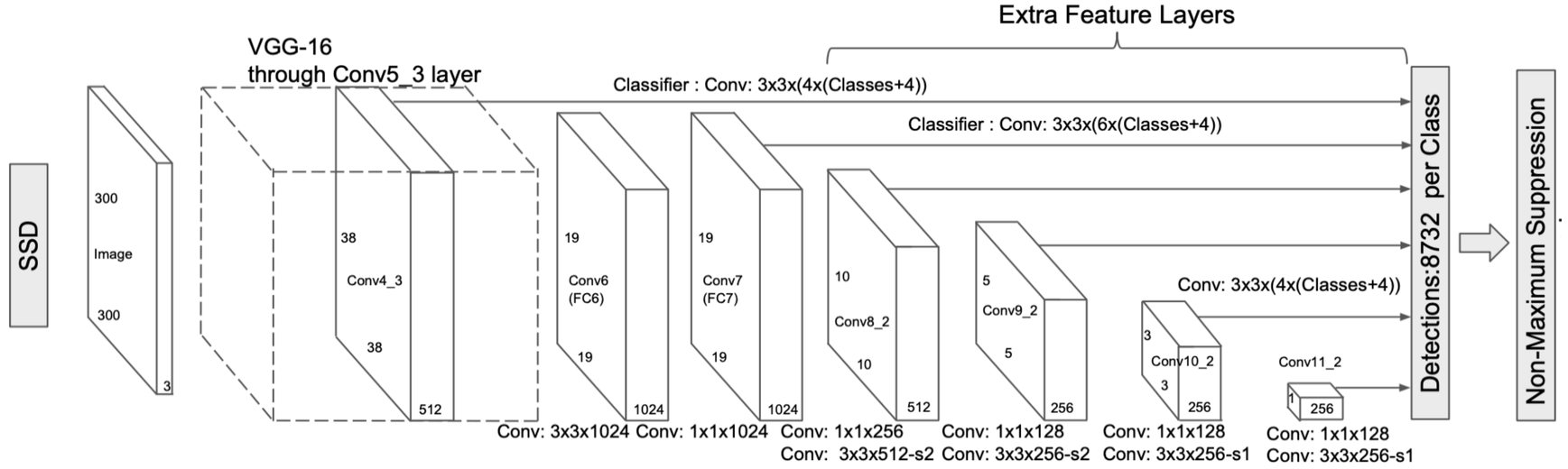
- Some alternatives:
  - Fast(er) R-CNN — end-to-end version of R-CNN
  - YOLO
  - Single Shot Multibox Detector

SSD Architecture

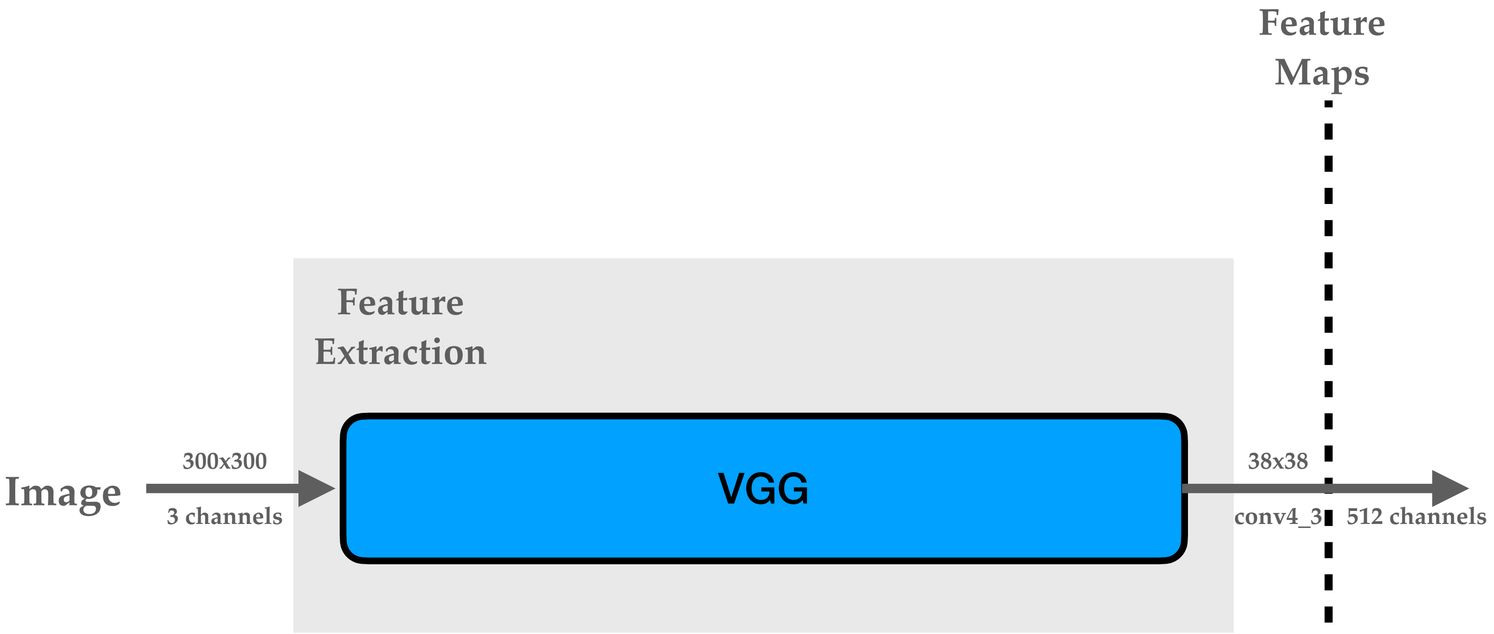
Key ideas:

1. multiple layers ⇒ handle different scales
2. different filters predict boxes of different shapes/sizes





SSD Architecture



VGG

conv3-64
conv3-64

conv3-128
conv3-128

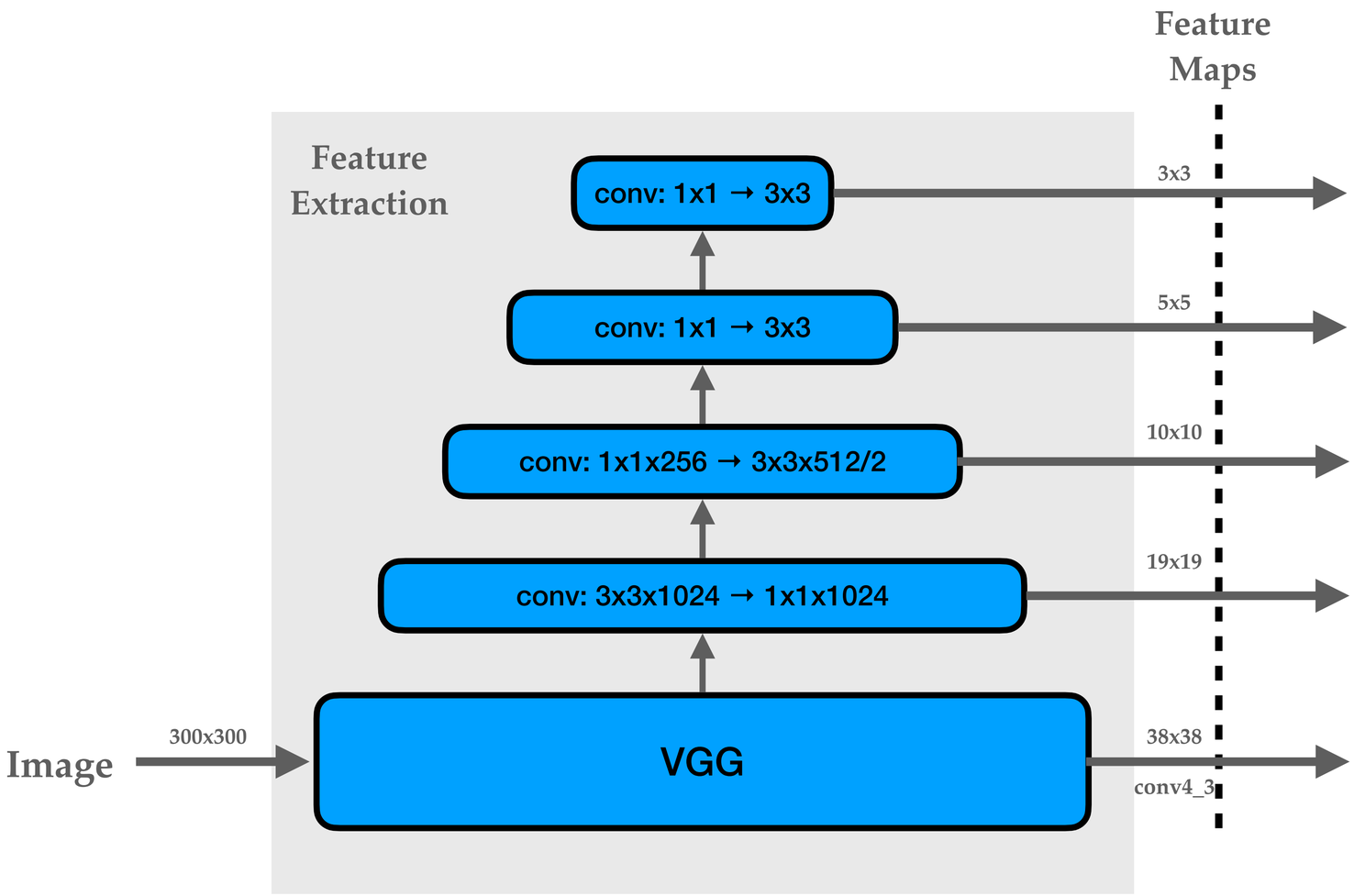
conv3-256
conv3-256
conv3-256
conv3-256

conv3-512
conv3-512
conv3-512
conv3-512

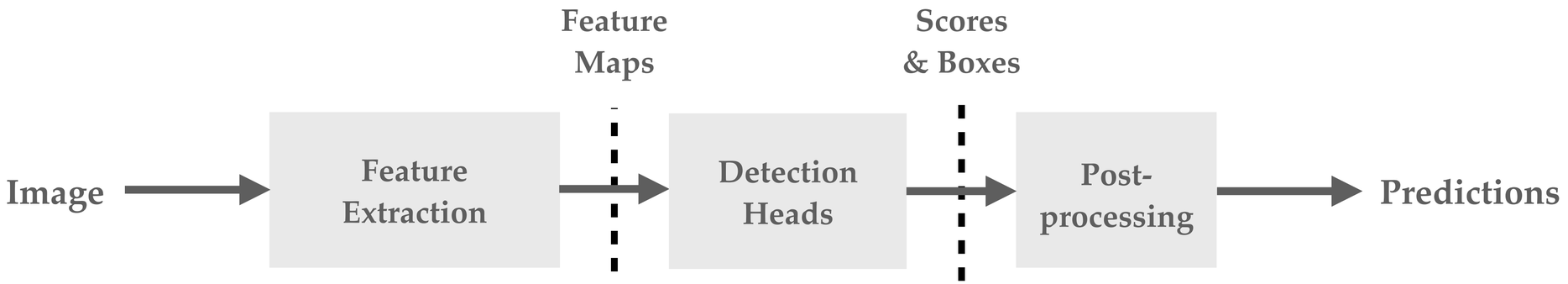
conv3-512
conv3-512
conv3-512
conv3-512

~~maxpool~~
~~FC-4096~~
~~FC-4096~~
~~FC-1000~~
~~soft-max~~

SSD Architecture

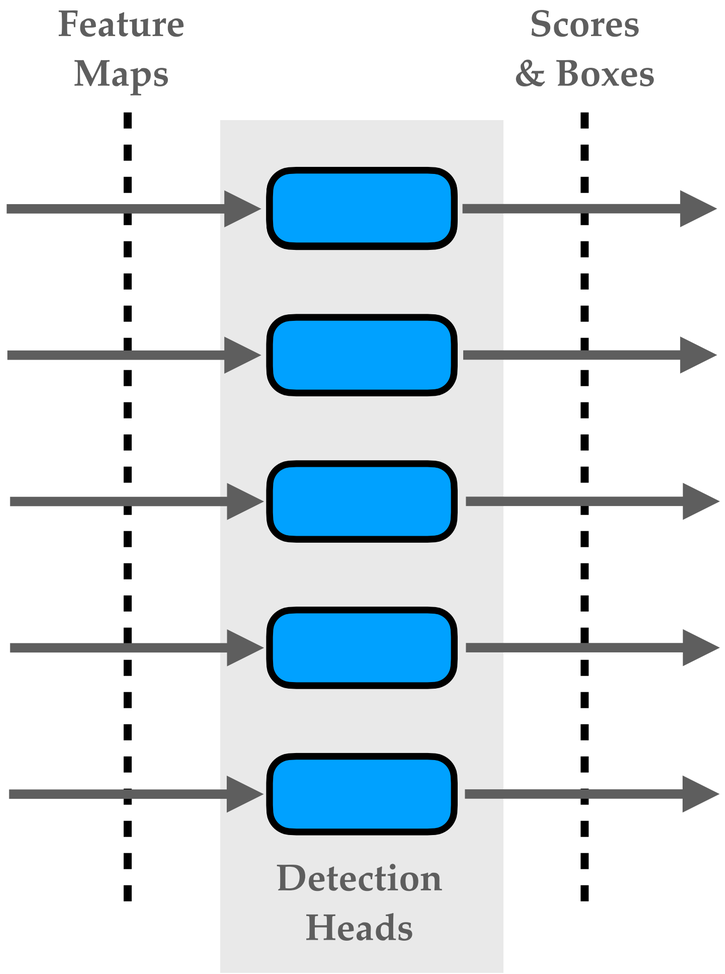


SSD Architecture



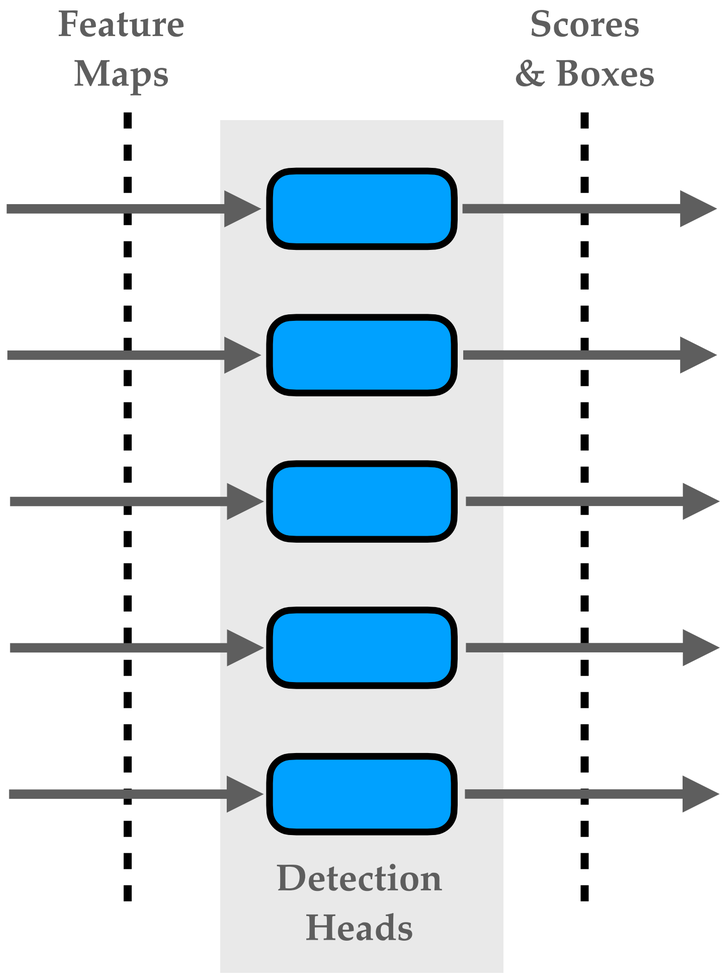
SSD Architecture

Multibox detector



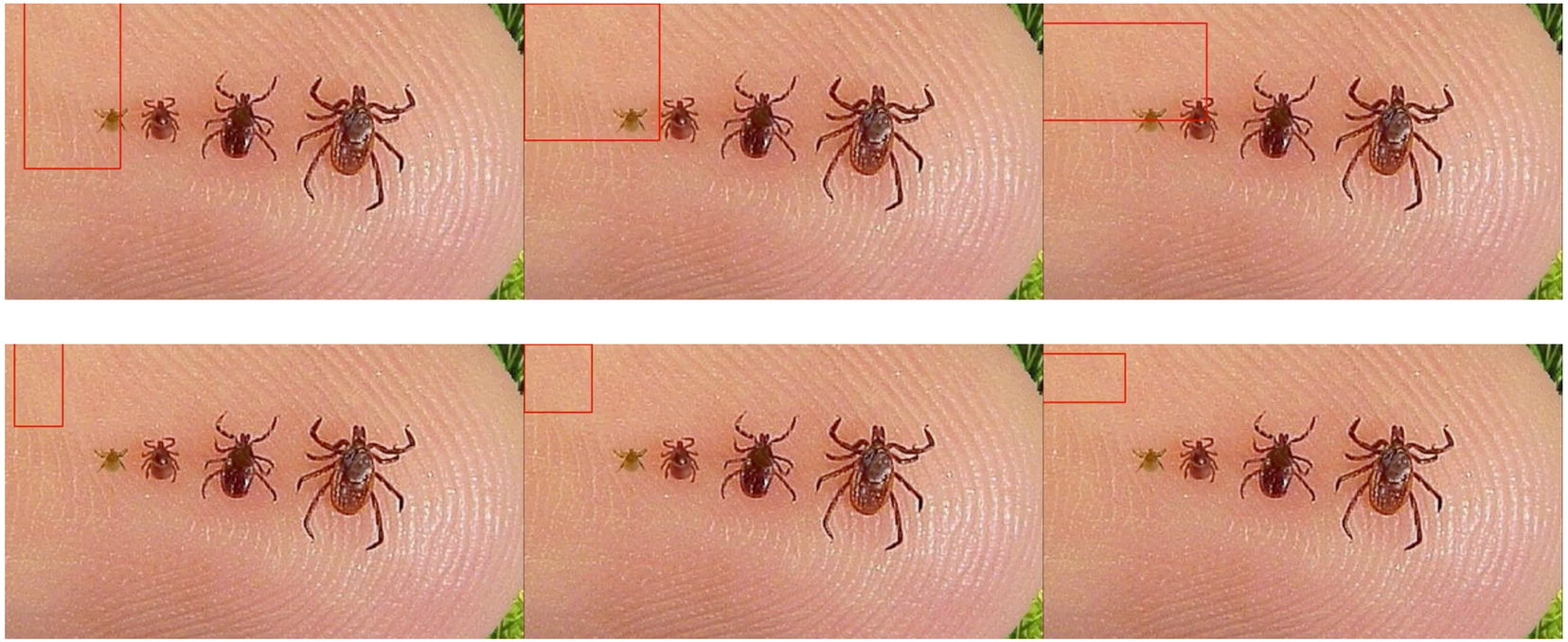
SSD Architecture

Multibox detector



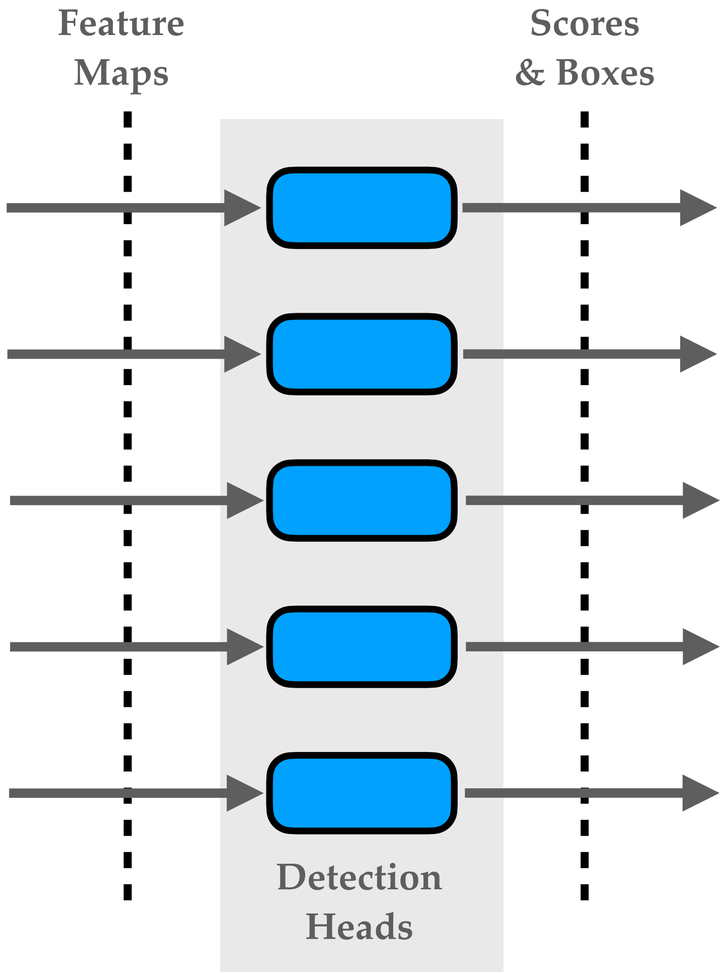
conv $3 \times 3 \times (n_A \cdot n_C)$ confidence scores

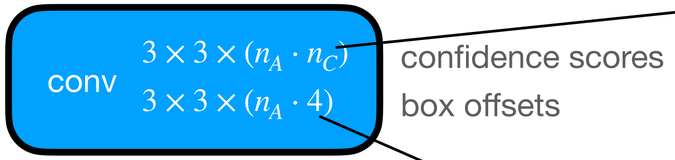
$3 \times 3 \times (n_A \cdot 4)$ box offsets



SSD Architecture

Multibox detector



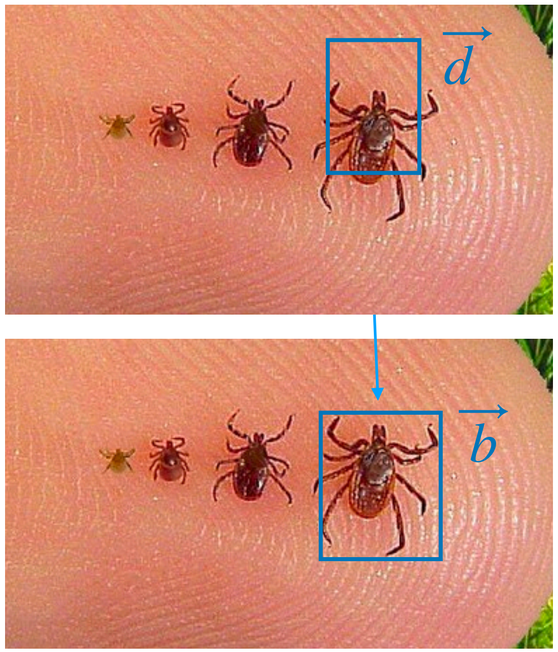


Confidence scores

background: 0.5
tick: 0.7
...

Box predictions

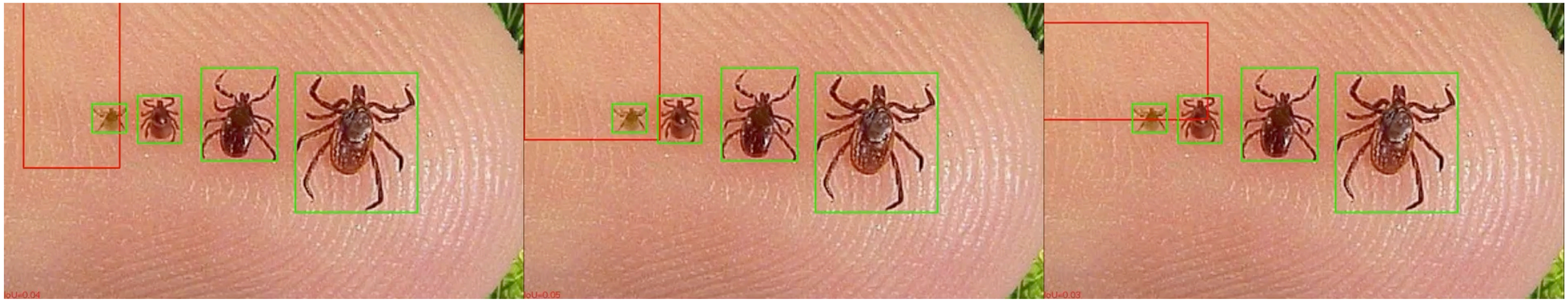
NN Output: $(\beta_x, \beta_y, \beta_w, \beta_h)$
Default box: (d_x, d_y, d_w, d_h)
Prediction: (b_x, b_y, b_w, b_h)



$$b_x = d_x + d_w \beta_x \qquad b_w = d_w e^{\beta_w}$$
$$b_y = d_y + d_h \beta_y \qquad b_h = d_h e^{\beta_h}$$

SSD Architecture

Labels

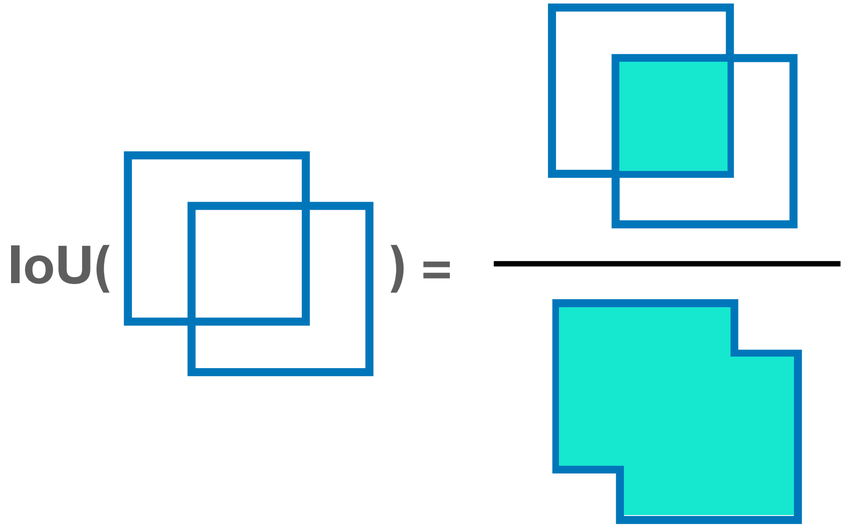


~ Pos: $(\hat{c}_d, \hat{\beta}_d)$ ~ (class target, offsets target) for default box d

Neg: d ~ default box containing no object

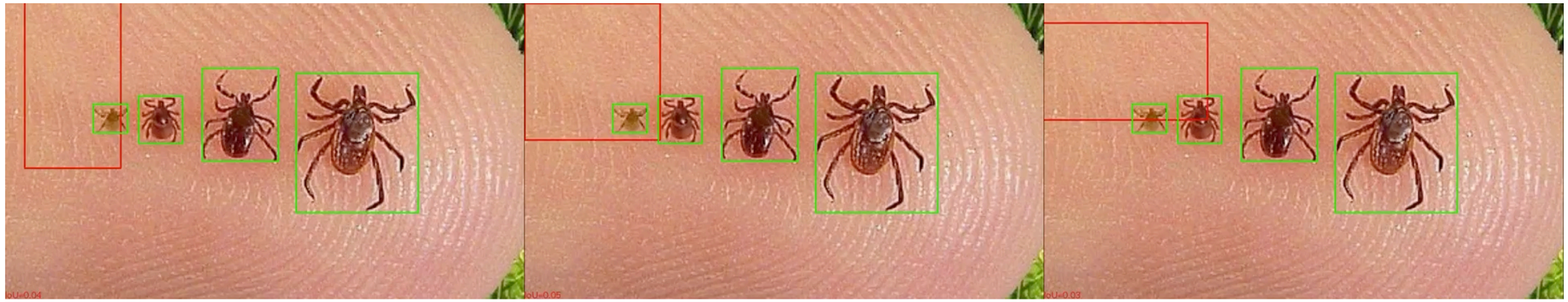
} **labels**

positive ⇔ intersection over union > threshold (0.5)



SSD Architecture

Loss



~ Pos: $(\hat{c}_d, \hat{\beta}_d)$ ~ (class target, offsets target) for default box
Neg: d ~ default boxes containing no object
} **labels**

Network output: $(\sigma_{d,c}, \beta_d)$ ~ (logit for class c, offsets target) for default box d

$$L_{loc} = \sum_{(\hat{c}_d, \hat{\beta}_d) \in Pos} \ell_{loc}(\beta_d - \hat{\beta}_d)$$

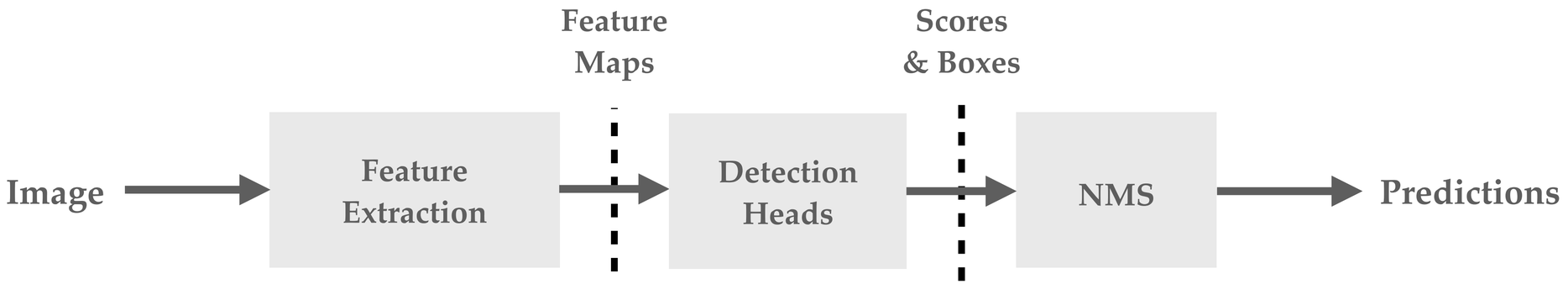
Typically: ℓ_{loc} = smooth ℓ_1

$$L_{conf} = -\sum_{(\hat{c}_d, \hat{\beta}_d) \in Pos} \log(\gamma_{d,\hat{c}_d}) - \sum_{d \in Neg} \log(\gamma_{d,bg})$$

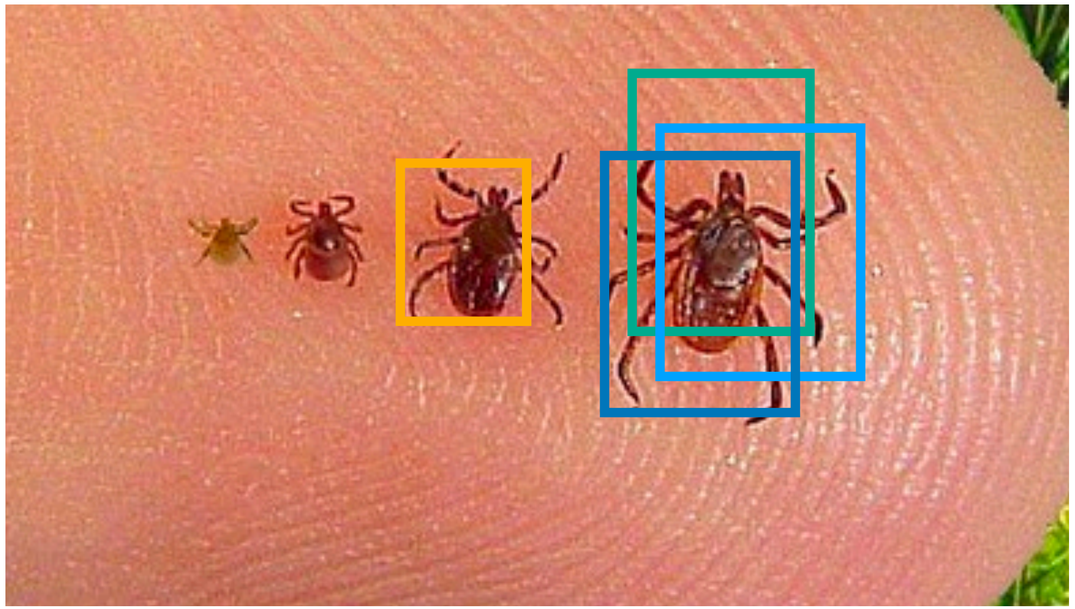
$$\gamma_{d,c} = \frac{\exp(\sigma_{c,d})}{\sum_{c'} \exp(\sigma_{c',d})}$$ (probability for class c)

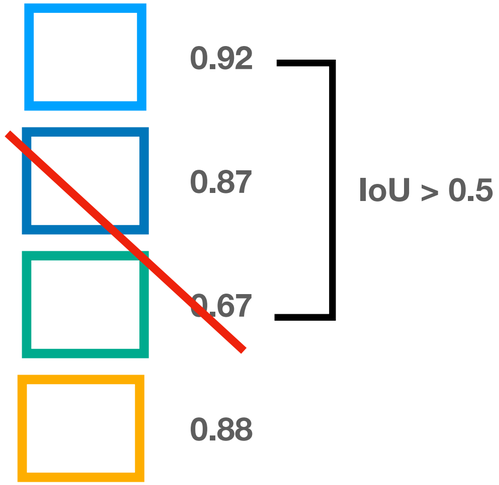
Imbalance problem: hard negative mining, focal loss, ...

SSD Architecture

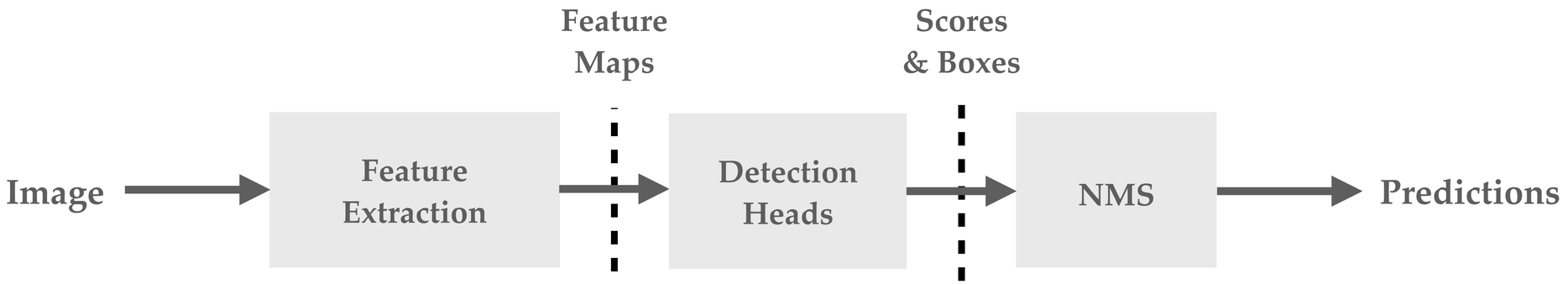


NMS

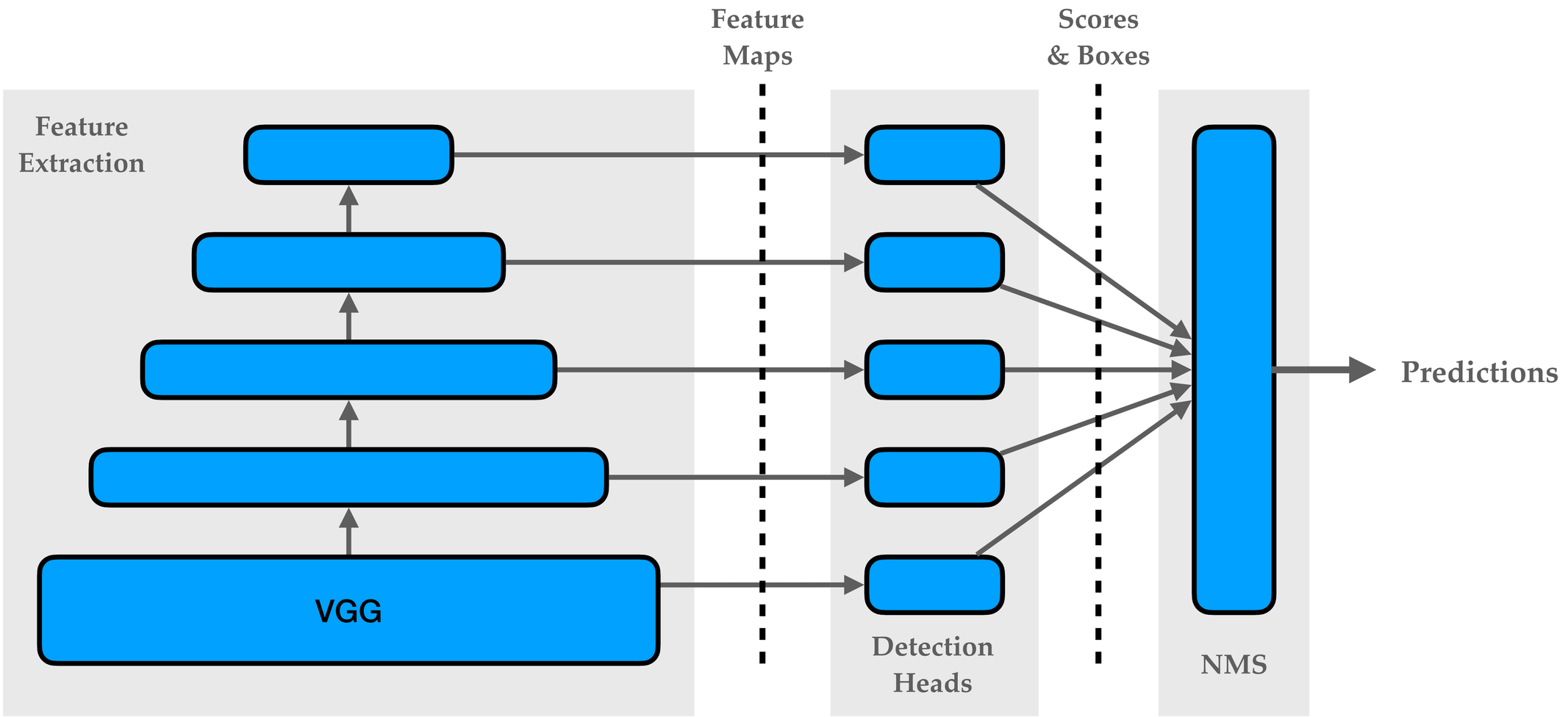




SSD Architecture



SSD Architecture



Performance

Method	data	Avg. Precision, IoU:			Avg. Precision, Area:			Avg. Recall, #Dets:			Avg. Recall, Area:		
		0.5:0.95	0.5	0.75	S	M	L	1	10	100	S	M	L
Fast [6]	train	19.7	35.9	-	-	-	-	-	-	-	-	-	-
Fast [24]	train	20.5	39.9	19.4	4.1	20.0	35.8	21.3	29.5	30.1	7.3	32.1	52.0
Faster [2]	trainval	21.9	42.7	-	-	-	-	-	-	-	-	-	-
ION [24]	train	23.6	43.2	23.6	6.4	24.1	38.3	23.2	32.7	33.5	10.1	37.7	53.6
Faster [25]	trainval	24.2	45.3	23.5	7.7	26.4	37.1	23.8	34.0	34.6	12.0	38.5	54.4
SSD300	trainval35k	23.2	41.2	23.4	5.3	23.2	39.6	22.5	33.2	35.3	9.6	37.6	56.5
SSD512	trainval35k	**26.8**	**46.5**	**27.8**	**9.0**	**28.9**	**41.9**	**24.8**	**37.5**	**39.8**	**14.0**	**43.5**	**59.0**

Table 5: **COCO `test-dev2015` detection results.**

This was great in 2015, but...

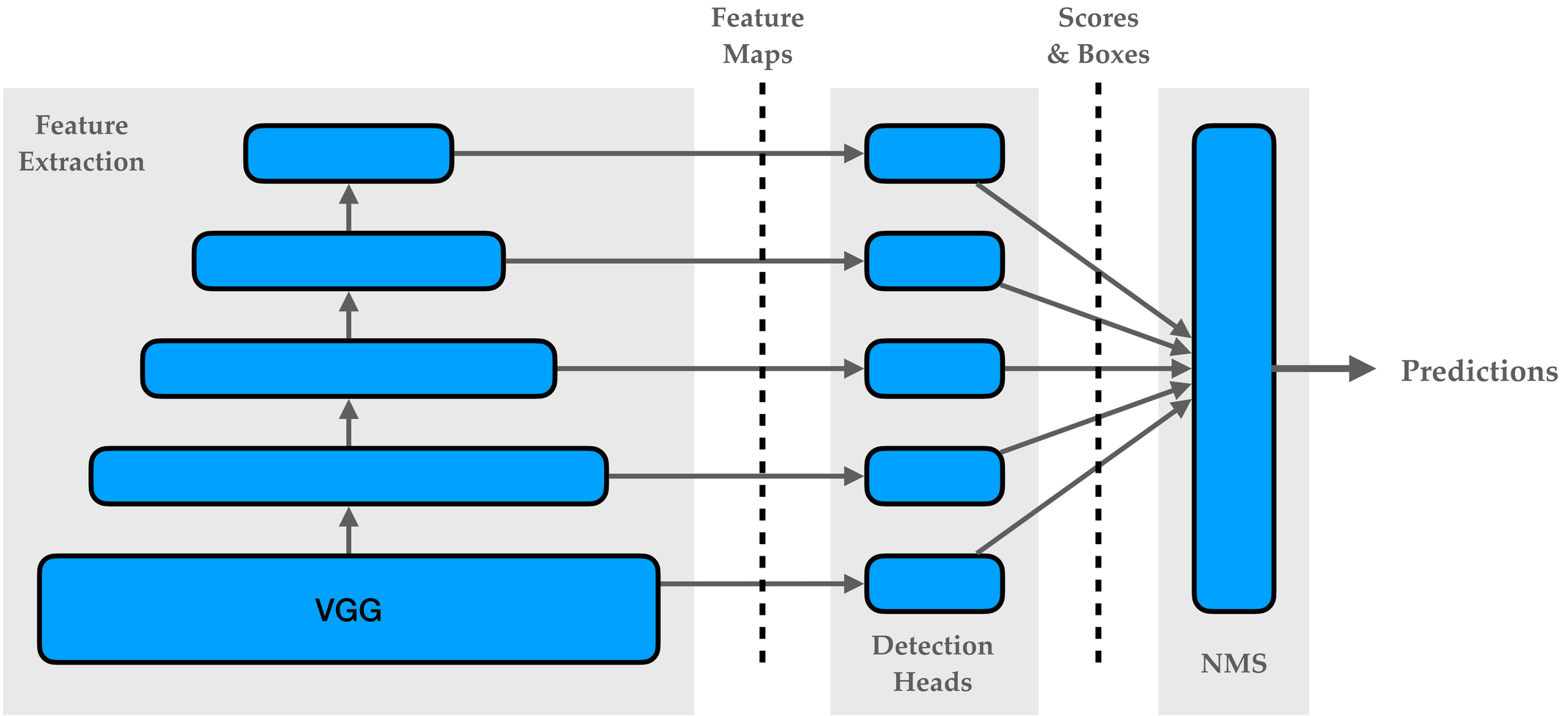
Source: **SSD**

Performance

Model	test-dev AP	test-dev AP_{50}	test-dev AP_{75}	val AP	Params	Ratio	FLOPs	Ratio	Latency (ms) TitianV	Latency (ms) V100
EfficientDet-D0 (512)	**34.6**	**53.0**	**37.1**	**34.3**	**3.9M**	**1x**	**2.5B**	**1x**	**12**	**10.2**
YOLOv3 [34]	33.0	57.9	34.4	-	-	-	71B	28x	-	-
EfficientDet-D1 (640)	**40.5**	**59.1**	**43.7**	**40.2**	**6.6M**	**1x**	**6.1B**	**1x**	**16**	**13.5**
RetinaNet-R50 (640) [24]	39.2	58.0	42.3	39.2	34M	6.7x	97B	16x	25	-
RetinaNet-R101 (640)[24]	39.9	58.5	43.0	39.8	53M	8.0x	127B	21x	32	-
EfficientDet-D2 (768)	**43.9**	**62.7**	**47.6**	**43.5**	**8.1M**	**1x**	**11B**	**1x**	**23**	**17.7**
Detectron2 Mask R-CNN R101-FPN [1]	-	-	-	42.9	63M	7.7x	164B	15x	-	56‡
Detectron2 Mask R-CNN X101-FPN [1]	-	-	-	44.3	107M	13x	277B	25x	-	103‡
EfficientDet-D3 (896)	**47.2**	**65.9**	**51.2**	**46.8**	**12M**	**1x**	**25B**	**1x**	**37**	**29.0**
ResNet-50 + NAS-FPN (1024) [10]	44.2	-	-	-	60M	5.1x	360B	15x	64	-
ResNet-50 + NAS-FPN (1280) [10]	44.8	-	-	-	60M	5.1x	563B	23x	99	-
ResNet-50 + NAS-FPN (1280@384)[10]	45.4	-	-	-	104M	8.7x	1043B	42x	150	-
EfficientDet-D4 (1024)	**49.7**	**68.4**	**53.9**	**49.3**	**21M**	**1x**	**55B**	**1x**	**65**	**42.8**
AmoebaNet+ NAS-FPN +AA(1280)[45]	-	-	-	48.6	185M	8.8x	1317B	24x	246	-
EfficientDet-D5 (1280)	**51.5**	**70.5**	**56.1**	**51.3**	**34M**	**1x**	**135B**	**1x**	**128**	**72.5**
Detectron2 Mask R-CNN X152 [1]	-	-	-	50.2	-	-	-	-	-	234‡
EfficientDet-D6 (1280)	**52.6**	**71.5**	**57.2**	**52.2**	**52M**	**1x**	**226B**	**1x**	**169**	**92.8**
AmoebaNet+ NAS-FPN +AA(1536)[45]	-	-	-	50.7	209M	4.0x	3045B	13x	489	-
EfficientDet-D7 (1536)	**53.7**	**72.4**	**58.4**	**53.4**	**52M**		**325B**		**232**	**122**
EfficientDet-D7x (1536)	**55.1**	**74.3**	**59.9**	**54.4**	**77M**		**410B**		**285**	**153**

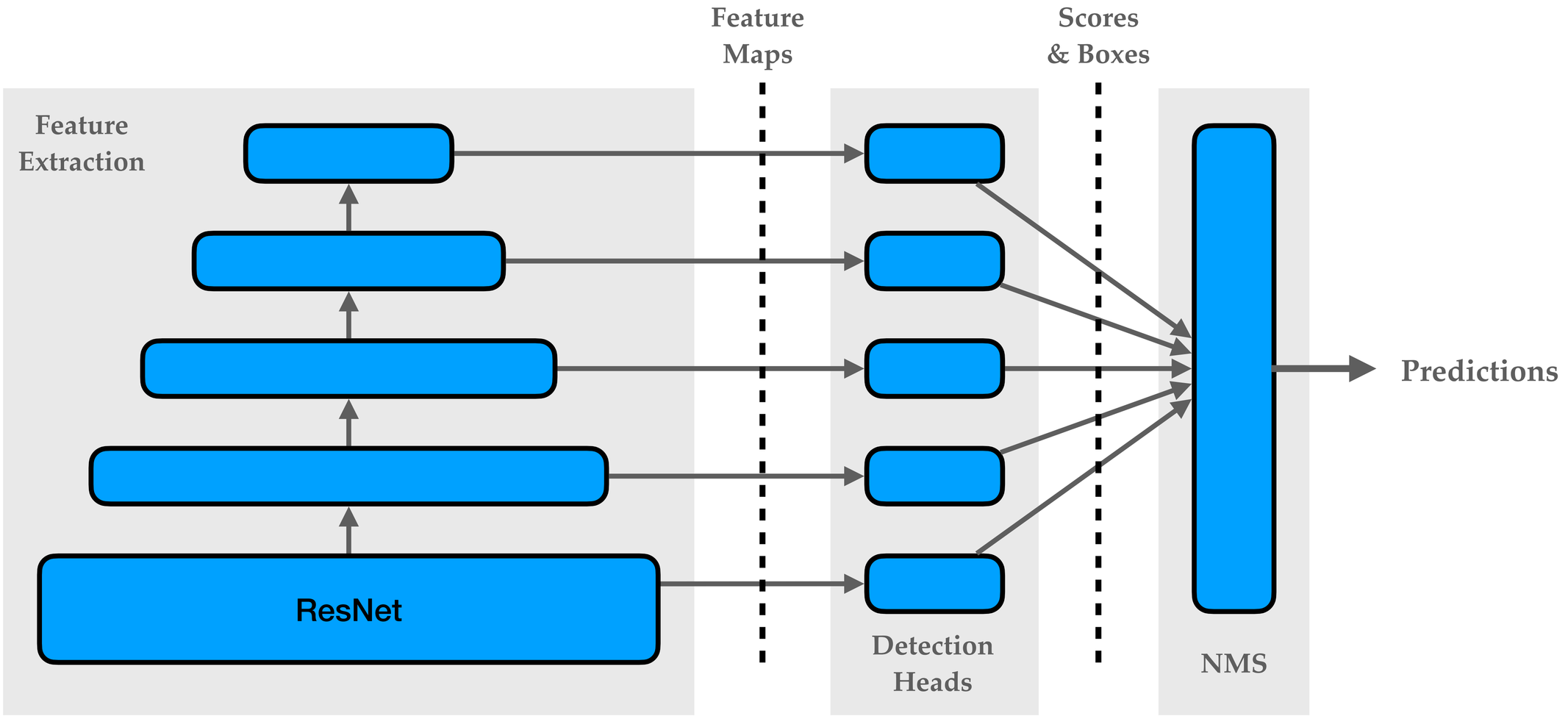
Source: **EfficientDet**

Modifications to SSD Architecture



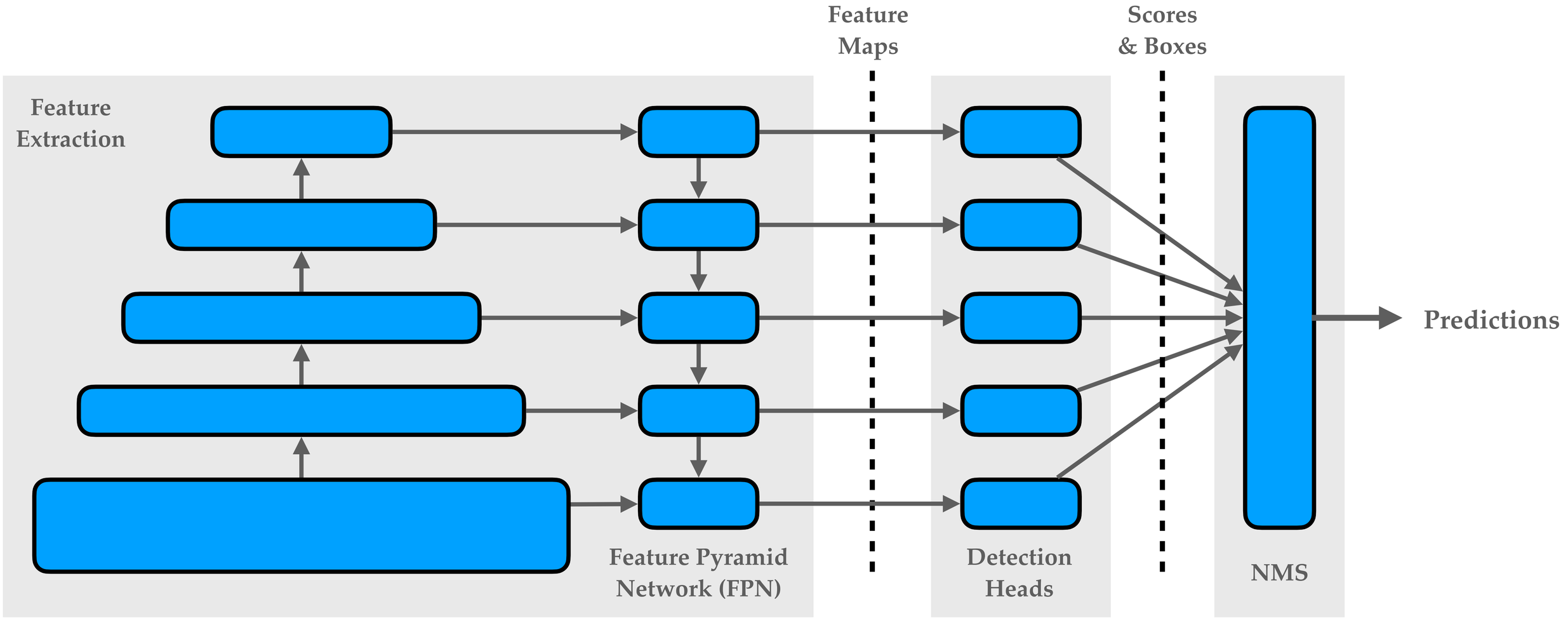
Modifications to SSD Architecture

Change the backbone



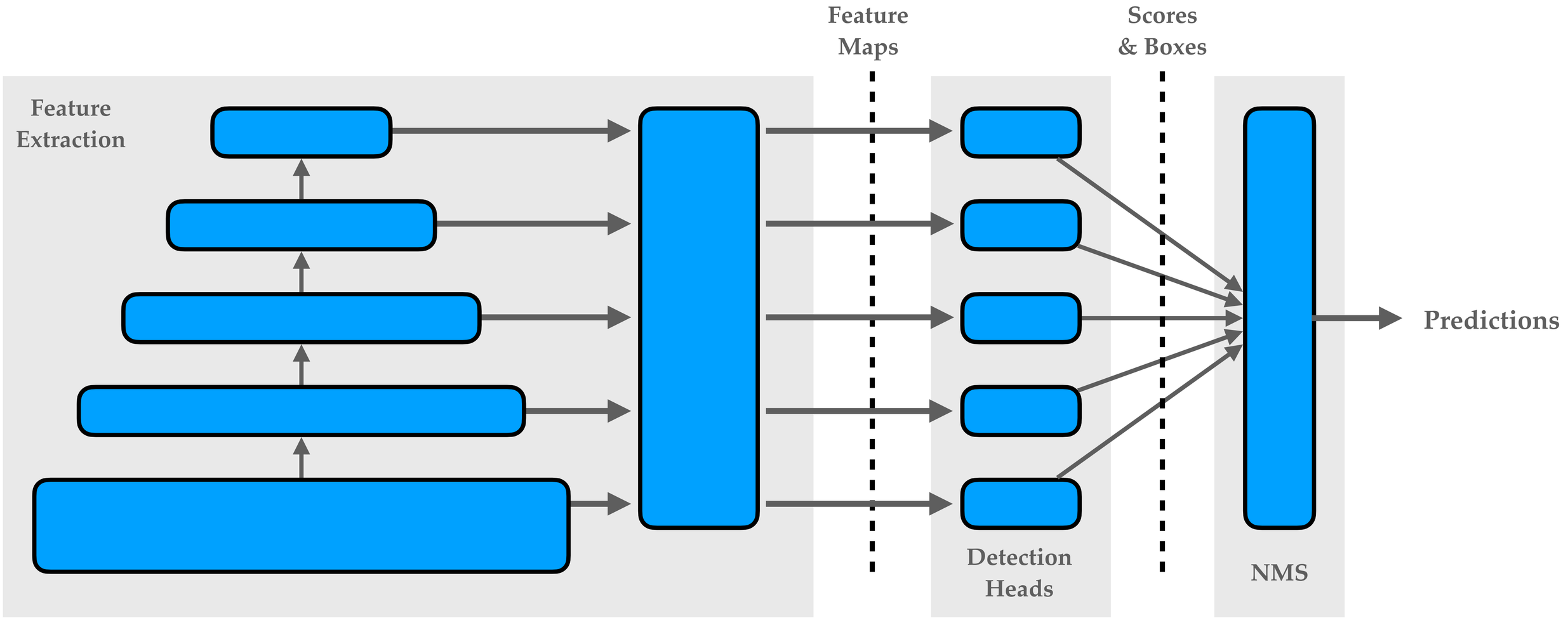
Modifications to SSD Architecture

Augment the feature extractor: FPN



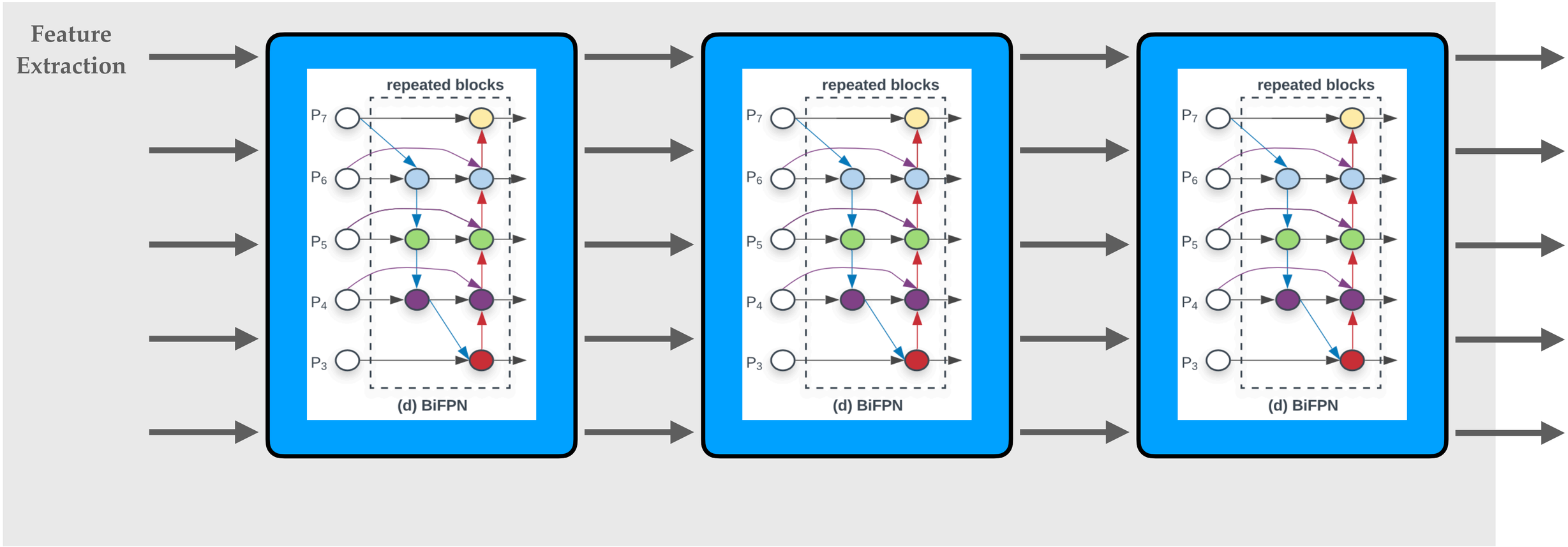
Modifications to SSD Architecture

Augment the feature extractor



Modifications to SSD Architecture

Augment the feature extractor: EfficientDet's BiFPN



Graphic source: **EfficientDet**

References

Papers

- **SSD:** Liu *et al.*, "SSD: Single Shot Multibox Detector" (arXiv: 1512.02325)
- **R-CNN:** Girschick, Donahue, Darrell, & Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation" (arXiv: 1311.2524)
- **FPN:** Lin *et al.*, "Feature Pyramid Networks for Object Detection" (arXiv: 1612.03144)
- **EfficientDet:** Tan, Pang, & Le, "EfficientDet: Scalable and Efficient Object Detection" (arXiv: 1911.09070)

Images

- Dysentery amoeba: https://commons.wikimedia.org/wiki/File:Trophozoites_of_Entamoeba_histolytica_with_ingested_erythrocytes.JPG
- Ticks: Fairfax County. https://www.flickr.com/photos/fairfaxcounty/7209178448