深度学习-目标检测篇

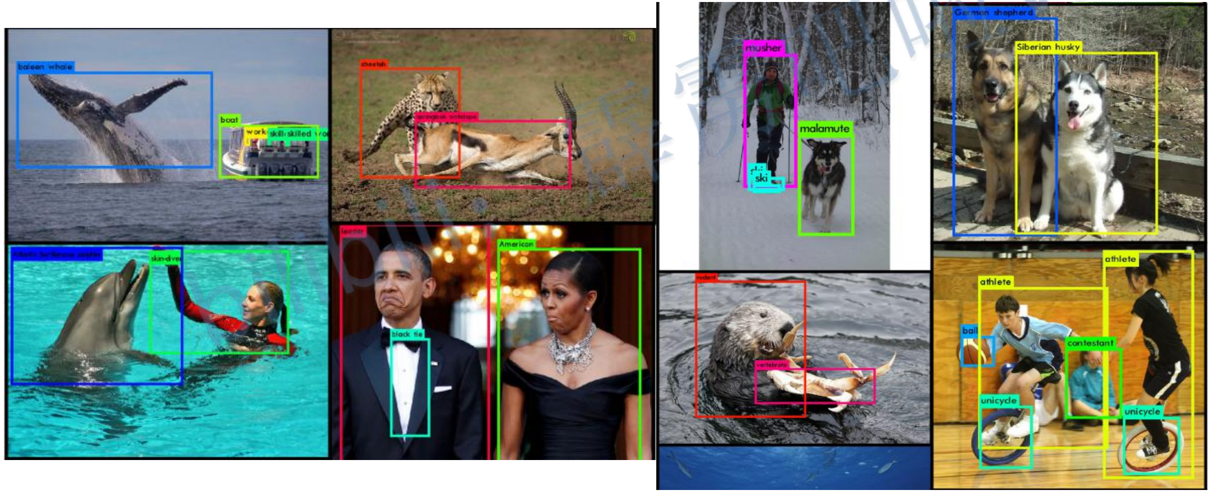
作者：神秘的wz

YOLO v2

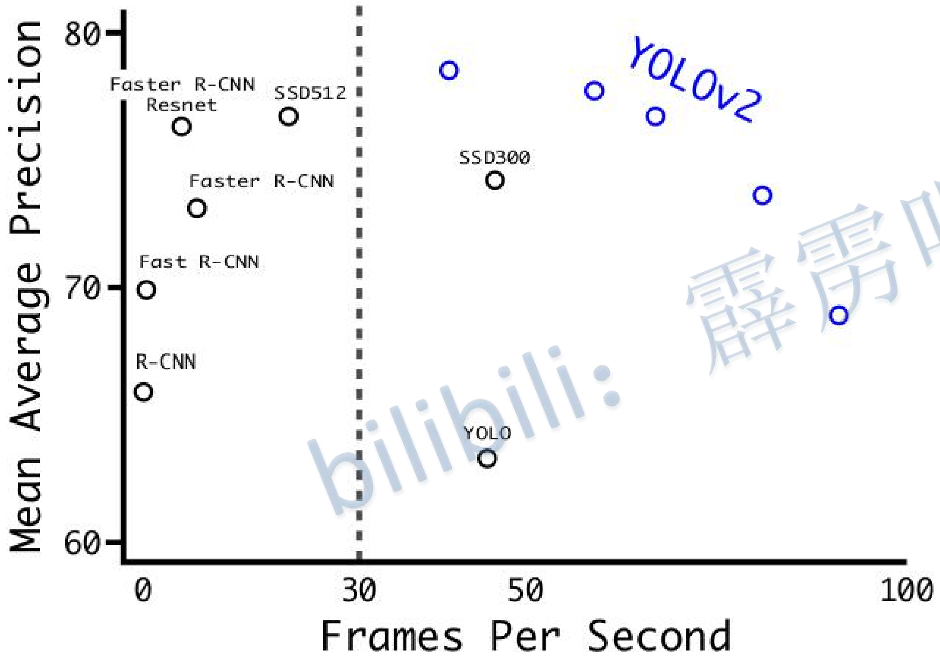
YOLO9000:
Better, Faster, Stronger

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

2017 CVPR



YOLO v2



Detection Frameworks	Train	mAP	FPS
Fast R-CNN [5]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[15]	2007+2012	73.2	7
Faster R-CNN ResNet[6]	2007+2012	76.4	5
YOLO [14]	2007+2012	63.4	45
SSD300 [11]	2007+2012	74.3	46
SSD500 [11]	2007+2012	76.8	19
YOLOv2 288 × 288	2007+2012	69.0	91
YOLOv2 352 × 352	2007+2012	73.7	81
YOLOv2 416 × 416	2007+2012	76.8	67
YOLOv2 480 × 480	2007+2012	77.8	59
YOLOv2 544 × 544	2007+2012	**78.6**	40

YOLO v2

YOLOv2中的各种尝试

Better章节

- Batch Normalization
- High Resolution Classifier
- Convolutional With Anchor Boxes
- Dimension Clusters
- Direct location prediction
- Fine-Grained Features
- Multi-Scale Training

YOLO v2

Batch Normalization

Batch Normalization. Batch normalization leads to significant improvements in convergence while eliminating the need for other forms of regularization [7]. By adding batch normalization on all of the convolutional layers in YOLO we get more than 2% improvement in mAP. Batch normalization also helps regularize the model. With batch normalization we can remove dropout from the model without overfitting.

YOLO v2

High Resolution Classifier

For YOLOv2 we first fine tune the classification network at the full 448×448 resolution for 10 epochs on ImageNet. This gives the network time to adjust its filters to work better on higher resolution input. We then fine tune the resulting network on detection. This high resolution classification network gives us an increase of almost 4% mAP.

YOLO v2

Convolutional With Anchor Boxes

Convolutional With Anchor Boxes. YOLO predicts the coordinates of bounding boxes directly using fully connected layers on top of the convolutional feature extractor.

map. Predicting offsets instead of coordinates simplifies the problem and makes it easier for the network to learn.

Using anchor boxes we get a small decrease in accuracy. YOLO only predicts 98 boxes per image but with anchor boxes our model predicts more than a thousand. Without anchor boxes our intermediate model gets 69.5 mAP with a recall of 81%. With anchor boxes our model gets 69.2 mAP with a recall of 88%. Even though the mAP decreases, the increase in recall means that our model has more room to improve.

YOLO v2

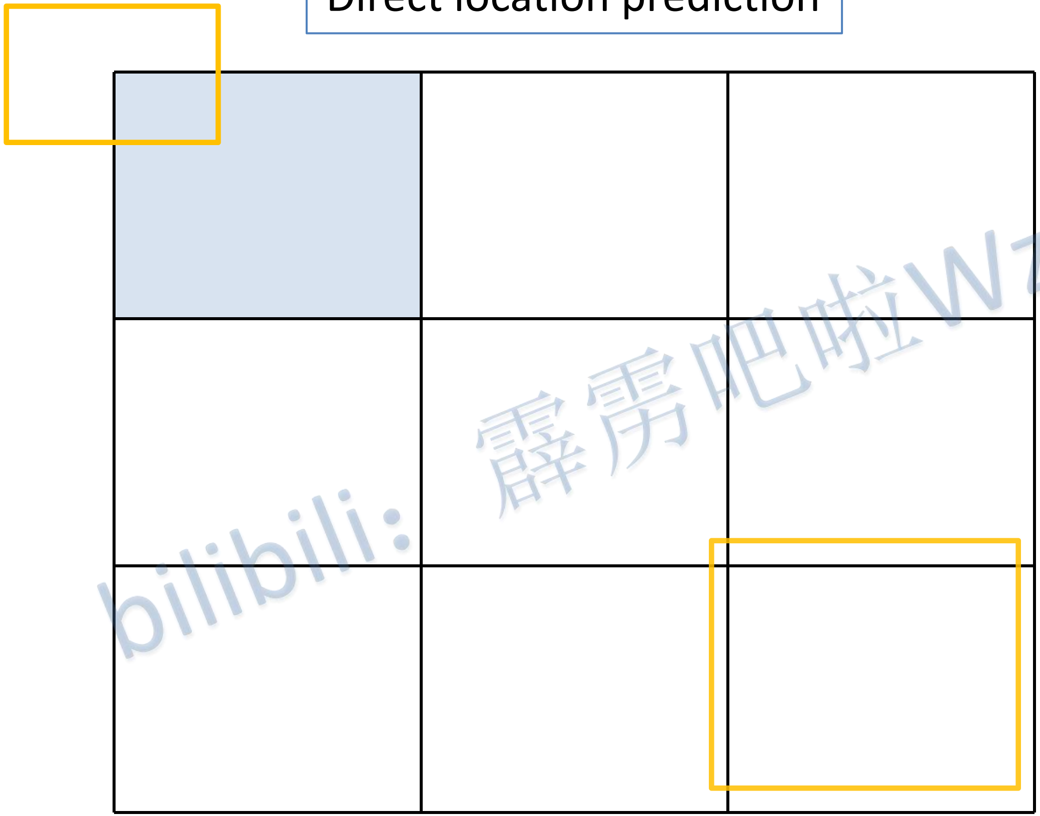
Dimension Clusters

Dimension Clusters. We encounter two issues with anchor boxes when using them with YOLO. The first is that the box dimensions are hand picked. The network can learn to adjust the boxes appropriately but if we pick better priors for the network to start with we can make it easier for the network to learn to predict good detections.

Instead of choosing priors by hand, we run k-means clustering on the training set bounding boxes to automatically find good priors. If we use standard k-means with Euclidean distance larger boxes generate more error than smaller boxes. However, what we really want are priors that lead to good IOU scores, which is independent of the size of the box. Thus for our distance metric we use:

YOLO v2

Direct location prediction



YOLO v2

Fine-Grained Features

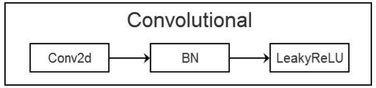
Fine-Grained Features. This modified YOLO predicts detections on a 13×13 feature map. While this is sufficient for large objects, it may benefit from finer grained features for localizing smaller objects. Faster R-CNN and SSD both run their proposal networks at various feature maps in the network to get a range of resolutions. We take a different approach, simply adding a passthrough layer that brings features from an earlier layer at 26×26 resolution.

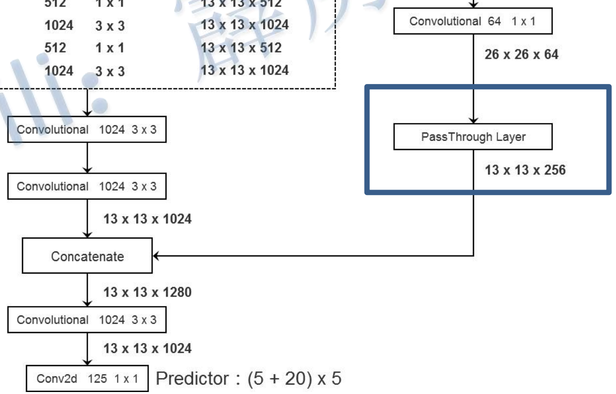
The passthrough layer concatenates the higher resolution features with the low resolution features by stacking adjacent features into different channels instead of spatial locations, similar to the identity mappings in ResNet. This turns the $26 \times 26 \times 512$ feature map into a $13 \times 13 \times 2048$ feature map, which can be concatenated with the original features. Our detector runs on top of this expanded feature map so that it has access to fine grained features. This gives a modest 1% performance increase.

YOLO v2

Darknet19

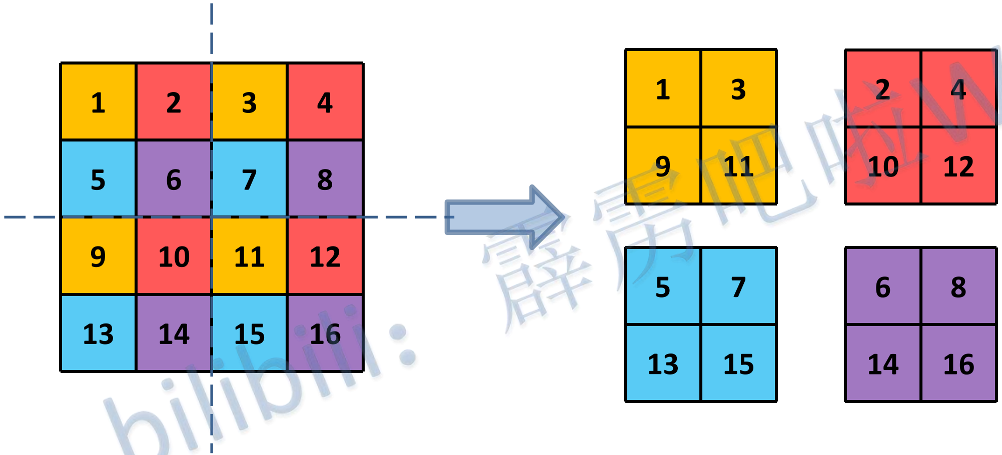
Type	Filters	Size	Output
Convolutional	32	3 x 3	416 x 416 x 32
Maxpool		2 x 2 / 2	208 x 208 x 32
Convolutional	64	3 x 3	208 x 208 x 64
Maxpool		2 x 2 / 2	104 x 104 x 64
Convolutional	128	3 x 3	104 x 104 x 128
Convolutional	64	1 x 1	104 x 104 x 64
Convolutional	128	3 x 3	104 x 104 x 128
Maxpool		2 x 2 / 2	52 x 52 x 128
Convolutional	256	3 x 3	52 x 52 x 256
Convolutional	128	1 x 1	52 x 52 x 128
Convolutional	256	3 x 3	52 x 52 x 256
Maxpool		2 x 2 / 2	26 x 26 x 256
Convolutional	512	3 x 3	26 x 26 x 512
Convolutional	256	1 x 1	26 x 26 x 256
Convolutional	512	3 x 3	26 x 26 x 512
Convolutional	256	1 x 1	26 x 26 x 256
Convolutional	512	3 x 3	26 x 26 x 512
Maxpool		2 x 2 / 2	13 x 13 x 512
Convolutional	1024	3 x 3	13 x 13 x 1024
Convolutional	512	1 x 1	13 x 13 x 512
Convolutional	1024	3 x 3	13 x 13 x 1024
Convolutional	512	1 x 1	13 x 13 x 512
Convolutional	1024	3 x 3	13 x 13 x 1024





YOLO v2

PassThrough Layer (W/2, H/2, Cx4)



YOLO v2

Multi-Scale Training

Multi-Scale Training. The original YOLO uses an input resolution of 448×448. With the addition of anchor boxes we changed the resolution to 416×416. However, since our model only uses convolutional and pooling layers it can be resized on the fly. We want YOLOv2 to be robust to running on images of different sizes so we train this into the model.

Instead of fixing the input image size we change the network every few iterations. Every 10 batches our network randomly chooses a new image dimension size. Since our model downsamples by a factor of 32, we pull from the following multiples of 32: $\{320, 352, ..., 608\}$. Thus the smallest option is 320×320 and the largest is 608×608. We resize the network to that dimension and continue training.

YOLO v2

BackBone: Darknet-19

Faster章节

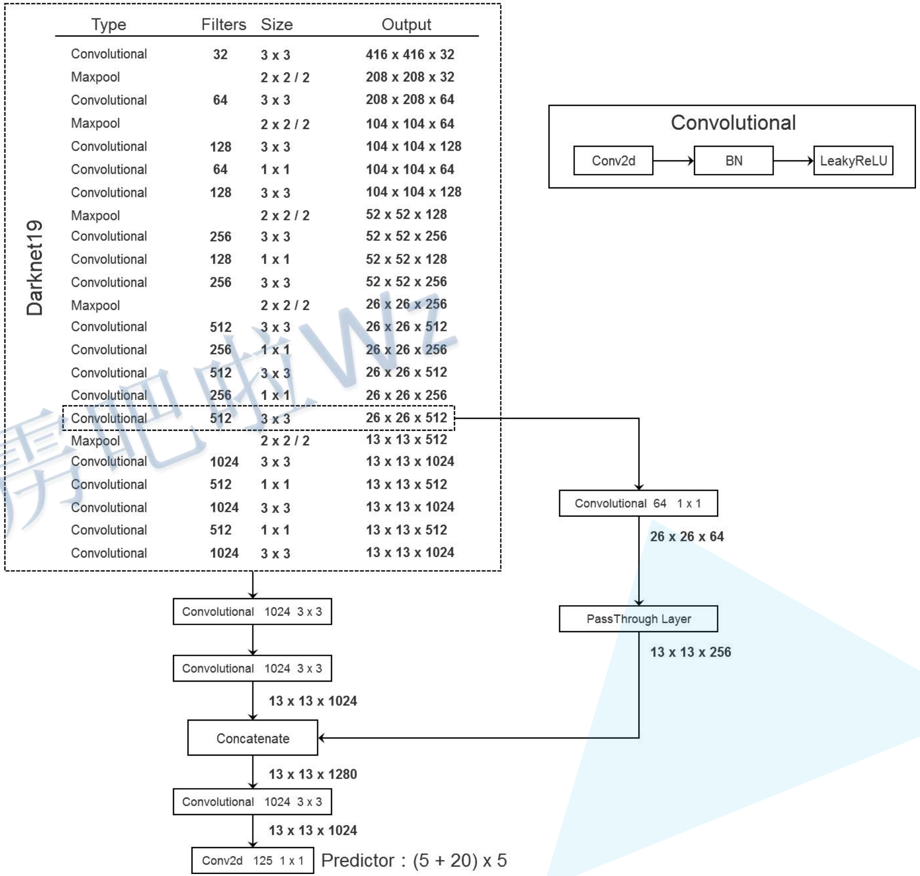
Darknet-19(224x224) only requires **5.58 billion operations** to process an image yet achieves **72.9% top-1 accuracy** and **91.2% top-5 accuracy** on **ImageNet**.

Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	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		$2 \times 2/2$	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			

YOLO v2

YOLOv2模型框架

Training for detection. We modify this network for detection by removing the last convolutional layer and instead adding on three 3×3 convolutional layers with 1024 filters each followed by a final 1×1 convolutional layer with the number of outputs we need for detection. For VOC we predict 5 boxes with 5 coordinates each and 20 classes per box so 125 filters. We also add a passthrough layer from the final $3 \times 3 \times 512$ layer to the second to last convolutional layer so that our model can use fine grain features.



Type	Filters	Size	Output
Convolutional	32	3 x 3	416 x 416 x 32
Maxpool		2 x 2 / 2	208 x 208 x 32
Convolutional	64	3 x 3	208 x 208 x 64
Maxpool		2 x 2 / 2	104 x 104 x 64
Convolutional	128	3 x 3	104 x 104 x 128
Convolutional	64	1 x 1	104 x 104 x 64
Convolutional	128	3 x 3	104 x 104 x 128
Maxpool		2 x 2 / 2	52 x 52 x 128
Convolutional	256	3 x 3	52 x 52 x 256
Convolutional	128	1 x 1	52 x 52 x 128
Convolutional	256	3 x 3	52 x 52 x 256
Maxpool		2 x 2 / 2	26 x 26 x 256
Convolutional	512	3 x 3	26 x 26 x 512
Convolutional	256	1 x 1	26 x 26 x 256
Convolutional	512	3 x 3	26 x 26 x 512
Convolutional	256	1 x 1	26 x 26 x 256
Convolutional	512	3 x 3	26 x 26 x 512
Maxpool		2 x 2 / 2	13 x 13 x 512
Convolutional	1024	3 x 3	13 x 13 x 1024
Convolutional	512	1 x 1	13 x 13 x 512
Convolutional	1024	3 x 3	13 x 13 x 1024
Convolutional	512	1 x 1	13 x 13 x 512
Convolutional	1024	3 x 3	13 x 13 x 1024

YOLO v2

关于网络的训练细节

如何匹配正负样本？

如何计算误差？

We use a weight decay of 0.0005 and momentum of 0.9. We use a similar data augmentation to YOLO and SSD with random crops, color shifting, etc. We use the same training strategy on COCO and VOC.

沟通方式

1.github

https://github.com/WZMIAOMIAO/deep-learning-for-image-processing

2.CSDN

https://blog.csdn.net/qq_37541097/article/details/103482003

3.bilibili

https://space.bilibili.com/18161609/channel/index

尽可能每周更新