生物智能与算法 Report-Final

赵昱 21821275

1 目标检测问题简介

目标检测问题分为两个阶段，即目标定位（确定目标所在位置，回归问题）和目标分类（确定候选框内的目标所属类别），常规方法通常采用滑动窗口加人工特征分类器进行检测。

YOLO 系列 [1–3] 是首个使用单个 CNN 模型同时处理回归和分类两个问题的神经网络模型。

本项目选择 YOLO 系列作为多类别目标检测（通常叫做通用目标检测）所采用的模型，尝试复现每个版本并进行比较，同时针对一些特殊应用（如人脸检测等单类别问题），根据自己的需求训练模型。

1.1 目标检测问题模型和求解思路

1.1.1 问题模型

目标检测问题要求检测到某类物体在图像中是否存在，并且给出其所在位置的包围框（Bounding Box）。目标检测问题的模型分为两个阶段，即目标定位（确定目标所在位置，回归问题）和目标分类（确定候选框内的目标所属类别）。

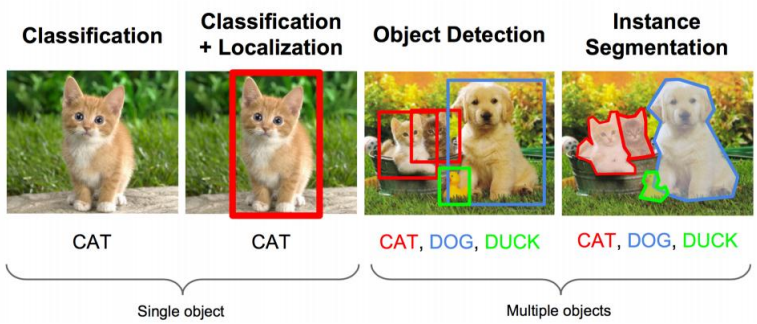


图. 1: 目标检测问题模型

常规方法的第一个阶段通常采用滑动窗口的方法来穷举包围框，由于物体的大小不一，滑动窗口的尺寸也会不断变化（在具体实现上，是通过缩放图像，如计算图像金字塔实现的）

在此基础上，使用人工特征分类器进行检测（分类），如使用基于 Haar 特征的分类器判断是否为人脸，基于 HOG 特征的分类器判断是否为行人等。

对于常规方法的改进，通常可以从这两个阶段着手。如针对目标定位的改进，有 RCNN 的 Selective Search，Fast RCNN 的 Region Proposal，或直接将其看成一个回归问题。针对分类器的改进，则通常使用更高层次或更复杂的特征，如使用 CNN 提取的特征。但是根据包围框候选策略得到的包围框通常不会完全贴合目标的边缘，所以很多方案会进行 Bounding Box Regression（位置精修），来得到更加准确的包围框。

1.1.2 检测结果后处理（Post Processing）

非极大值抑制（Non-Maximum Suppression，NMS）滑动窗口经提取特征，经分类器分类识别后，每个窗口都会得到一个分数。但是滑动窗口会导致很多窗口与其他窗口存在包含或者大部分交叉的情况。这时就需要用到 NMS 来选取那些邻域里分数最高（是目标的概率最大），并且抑制那些分数低的窗口。NMS 在计算机视觉领域有着非常重要的应用，如视频目标跟踪、数据挖掘、3D 重建、目标识别以及纹理分析等。常用的 IOU 阈值是 0.3　0.5

1.2 评估标准

常见的分类问题，通常使用混淆矩阵来表示分类的结果，即每个类别被分类的结果与真实结果的对比。对于二分类的混淆矩阵，可以计算召回率（Recall）、精确率（Precision），前者是真实样本被检出的概率，后者则是检出的准确率。

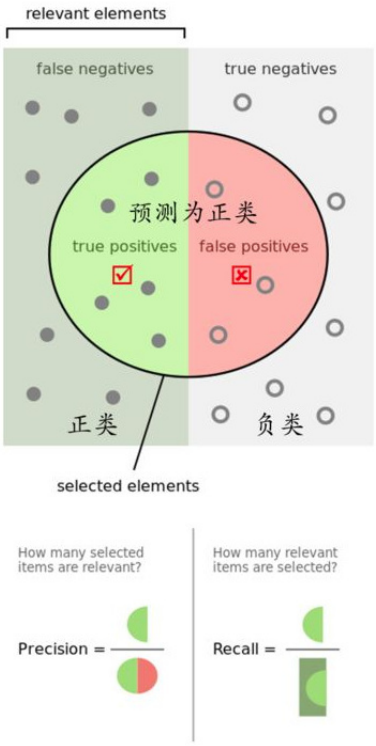


图. 2: 召回率和准确率

在此基础上，还可以计算 PR 曲线，ROC 曲线，F 指标等。

以上是针对分类算法的判别标准，评价一个检测算法时，主要看两个指标，即是否正确的预测了框内物体的类别；预测的框和人工标注框的重合程度，故需要以下两个评价标准:

- IOU(Intersection Over Union): 用来衡量预测的物体框和真实框的重合程度（通常我们规定 IOU $>$ 0.5 表示物体被检测出来，否则没有。调整 IOU 的阈值可以得到 PR 曲线。）
- 平均精度均值 (Mean Average Precision，mAP):mAP 即是把每个类别的 AP 都单独拿出来，然后计算所有类别 AP 的平均值，代表着对检测到的目标平均精度的一个综合度量。（AP 的计算方法之一：11 点插值法。就是选取 0,0.1,0.2$\cdots$1，这样的 11 个点，分别对应不同的 recall 级别，根据不同的级别计算最大 Precision（最大是指左侧最大），然后求出它的平均值。）

1.3 目标

(1)：精读论文，理解模型，分析方法的优缺点和创新之处
(2)：复现论文
(3)：和其他方法进行对比实验

1.4 论文摘要与翻译

1.4.1 [YOLO v1-CVPR2016] You Only Look Once: Unified, Real-Time Object Detection

Abstract:

We present YOLO, a new approach to object detection. Prior work on object detection repurposes classifiers to perform detection. Instead, we frame object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. A single neural network predicts bounding boxes and class probabilities directly from full images in one evaluation. Since the whole detection pipeline is a single network, it can be optimized end-to-end directly on detection performance.

Our unified architecture is extremely fast. Our base YOLO model processes images in real-time at 45 frames per second. A smaller version of the network, Fast YOLO, processes an astounding 155 frames per second while still achieving double the mAP of other real-time detectors. Compared to state-of-the-art detection systems, YOLO makes more localization errors but is less likely to predict false positives on background. Finally, YOLO learns very general representations of objects. It outperforms other detection methods, including DPM and R-CNN, when generalizing from natural images to other domains like artwork.

摘要翻译：

我们提出 YOLO——一种新的目标检测方法。以前的目标检测工作重新利用分类器来执行检测。相反，我们将目标检测框架看作回归问题从空间上分割边界框和相关的类别概率。单个神经网络在一次评估中直接从完整图像上预测边界框和类别概率。由于整个检测流程是单一网络，因此可以直接对检测性能进行端到端的优化。

我们的单一网络架构非常快。我们的基础 YOLO 模型以 45 帧/秒的速度实时处理图像。网络的一个较小版本，快速 YOLO，每秒能处理惊人的 155 帧，同时实现其它实时检测器两倍的 mAP。与最先进的检测系统相比，YOLO 产生了更多的定位误差，但不太可能在背景上的预测假阳性。最后，YOLO 学习目标非常通用

的表示。当从自然图像到艺术品等其它领域泛化时，它都优于其它检测方法，包括 DPM 和 R-CNN。

1.4.2 [YOLO v2-CVPR2017] YOLO9000: Better, Faster, Stronger

Abstract:

We introduce YOLO9000, a state-of-the-art, real-time object detection system that can detect over 9000 object categories. First we propose various improvements to the YOLO detection method, both novel and drawn from prior work. The improved model, YOLOv2, is state-of-the-art on standard detection tasks like PASCAL VOC and COCO. Us ing a novel, multi-scale training method the same YOLOv2 model can run at varying sizes, offering an easy tradeoff between speed and accuracy. At 67 FPS, YOLOv2 gets 76.8 mAP on VOC 2007. At 40 FPS, YOLOv2 gets 78.6 mAP, outperforming state-of-the-art methods like Faster R-CNN with ResNet and SSD while still running significantly faster. Finally we propose a method to jointly train on object detection and classification. Using this method we train YOLO9000 simultaneously on the COCO detection dataset and the ImageNet classification dataset. Our joint training allows YOLO9000 to predict detections for object classes that don’ t have labelled detection data. We validate our approach on the ImageNet detection task. YOLO9000 gets 19.7 mAP on the ImageNet detection validation set despite only having detection data for 44 of the 200 classes. On the 156 classes not in COCO, YOLO9000 gets 16.0 mAP. YOLO9000 predicts detections for more than 9000 different object categories, all in real-time.

摘要翻译：

我们提出 YOLO9000，这是一种先进的、实时的目标检测系统，可以检测超过 9000 个类别。首先，我们对 YOLO 做了一些改进，得到了提升后的模型 YOLO v2，这是一种在如 Pascal VOC 以及 COCO 等任务上表现出色，使用一种新奇的、多尺度的训练方法，YOLO v2 可以适应不同的尺寸，可以在速度与精度之间提供一个很好的平衡。在 67 帧率的时候，YOLO V2 可以获得 76.8 的 mAP 在 VOC2007 上，40 帧率的时候，可以获得 78.6 的 mAP 值，这比 Faster R-cnn 要好。最后，我们提出了一种联合训练的方法来进行目标检测和分类。使用这种方法，我们训练了 YOLO 9000，它可以在 ImageNet 和 COCO 上同时进行训练，我们的这种联合训练方法允许 YOLO9000 可以进行无标签目标的预测。我们在 ImageNet 检测任务中验证了我们的方法，在 44 类已有数据集中 YOLO9000 获得了 19.7 的mAP值，在其余的 156 类没有的样本中，获得了 16.0 的 mAP 值。但是 YOLO9000 可不是

只能预测 200 类数据，它可以同时检测出 9000 种目标，并且仍然可以实时的进行。

1.4.3 [YOLO v3-(arxiv)2018] YOLOv3: An Incremental Improvement

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 x 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 AP50 in 51 ms on a Titan X, compared to 57:5 AP50 in 198 ms by RetinaNet, similar performance but 3.8x faster. As always, all the code is online at https://pjreddie.com/yolo/.

摘要翻译：

本文为 YOLO 提供了一系列更新！它包含一堆小设计，可以使系统的性能得到更新；也包含一个新训练的、非常棒的神经网络，虽然比上一版更大一些，但精度也提高了。不用担心，虽然体量大了点，它的速度还是有保障的。在输入 320×320 的图片后，YOLOv3 能在 22 毫秒内完成处理，并取得 28.2mAP 的成绩。它的精度和 SSD 相当，但速度要快上 3 倍。和旧版数据相比，v3 版进步明显。在 Titan X 环境下，YOLOv3 的检测精度为 57.9AP，用时 51ms；而 RetinaNet 的精度只有 57.5AP，但却需要 198ms，相当于 YOLOv3 的 3.8 倍。

2 精读论文

2.1 [YOLO v1-CVPR2016] You Only Look Once: Unified, Real-Time Object Detection

方法: 单阶段（one-stage）目标检测。YOLO 将目标检测重新定义为单个回归问题，从图像像素直接到边界框坐标和类概率。

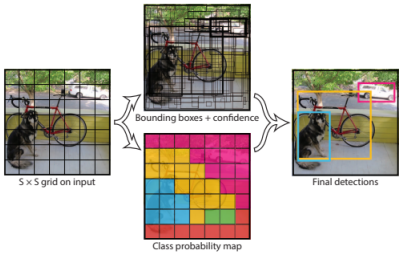


Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

图. 3: 方法

特色:

- 在预训练的时候用的是 224x224 的输入，一般预训练的分类模型都是在 ImageNet 数据集上进行的，然后在检测的时候采用 448x448 的输入
- 在图像上运行单个卷积网络
- 根据模型的置信度对得到的检测进行阈值化。(输出为 BB+ 类概率)

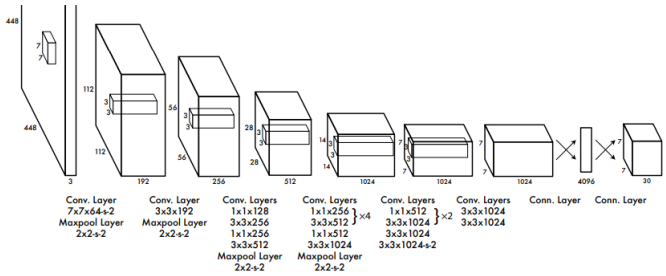


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

图. 4: 特色

结果:

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
YOLO VGG-16	2007+2012	66.4	21

Table 1: Real-Time Systems on PASCAL VOC 2007. Comparing the performance and speed of fast detectors. Fast YOLO is the fastest detector on record for PASCAL VOC detection and is still twice as accurate as any other real-time detector. YOLO is 10 mAP more accurate than the fast version while still well above real-time in speed.

图. 5: 结果

2.2 [YOLO v2-CVPR2017] YOLO9000: Better, Faster, Stronger

特色:

- Batch Normalization
- High Resolution Classifier：预训练分成两步：先用 224*224 的输入从头开始训练网络，大概 160 个 epoch，然后再将输入调整到 448*448，再训练 10 个 epoch。最后再在检测的数据集上 fine-tuning，也就是检测的时候用 448*448 的图像作为输入就可以顺利过渡了。
- Convolutional With Anchor Boxes
- Dimension Clusters：Faster R-CNN 中 anchor box 的大小和比例是按经验设定的，然后网络会在训练过程中调整 anchor box 的尺寸。如果一开始就能选择到合适尺寸的 anchor box，那肯定可以帮助网络更好地预测。所以作者采用 k-means 的方式对训练集的 bounding boxes 做聚类，试图找到合适的 anchor box。

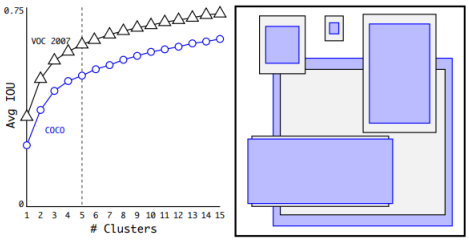


Figure 2: Clustering box dimensions on VOC and COCO. We run k-means clustering on the dimensions of bounding boxes to get good priors for our model. The left image shows the average IOU we get with various choices for k. $k = 5$ gives a good tradeoff for recall vs. complexity of the model. The right image shows the relative centroids for VOC and COCO. COCO has greater variation in size than VOC.

图. 6: anchor

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			

Table 6: Darknet-19.

图. 7: backbone

结果:

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

Table 3: Detection frameworks on PASCAL VOC 2007. YOLOv2 is faster and more accurate than prior detection methods. It can also run at different resolutions for an easy tradeoff between speed and accuracy. Each YOLOv2 entry is actually the same trained model with the same weights, just evaluated at a different size. All timing information is on a Geforce GTX Titan X (original, not Pascal model).

图. 8: 结果

2.3 [YOLO v3-(arxiv)2018] YOLOv3: An Incremental Improvement

特色:

- 借鉴了残差网络结构，形成更深的网络层次。
- 9 种尺度的先验框，mAP 以及对小物体的检测效果有一定的提升。

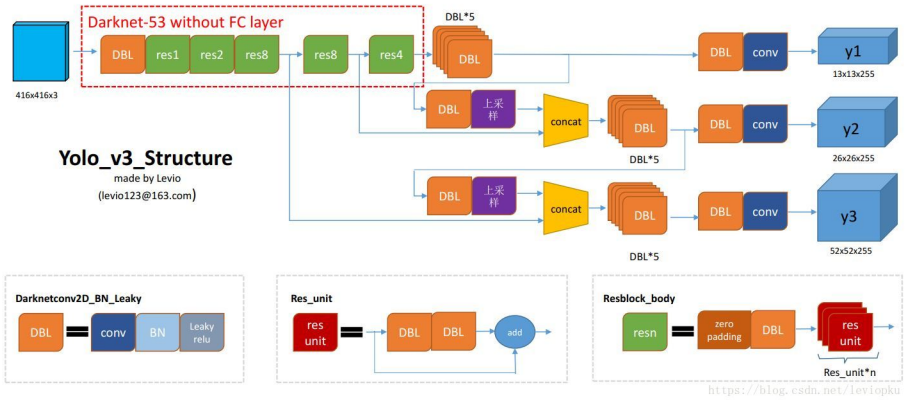


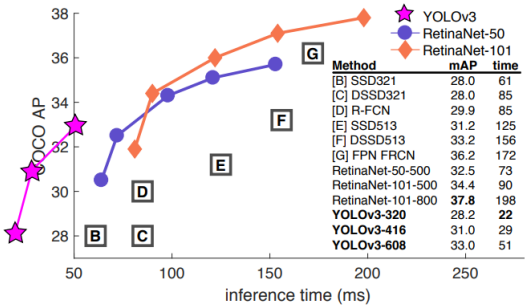
图. 9: model

	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			

Table 1. **Darknet-53.**

图. 10: backbone

结果:



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

Figure 1. We adapt this figure from the Focal Loss paper [9]. YOLOv3 runs significantly faster than other detection methods with comparable performance. Times from either an M40 or Titan X, they are basically the same GPU.

图. 11: 结果

3 论文创新点分析与复现

3.1 改进目标

如之前所描述的，目标检测为题分为两个阶段，即目标定位（确定目标所在位置，回归问题）和目标分类（确定候选框内的目标所属类别），因此改进通常针对

这两个阶段进行:

表. 1: 目标检测问题改进

类别	改进点	举例
目标定位	高效定位	Sliding Window
		Selective Search
	边框回归	Region Proposal
		BB Regression
	减少重复	NMS
目标分类	高层特征	Haar
		SIFT
	多尺度	CNN(VGG、ResNet)
		Image Pyramid

3.2 单阶段网络的定位改进

YOLO 将目标检测重新定义为单个回归问题（one-stage object detection），从图像像素直接到边界框坐标和类概率。这种方法最大的特点是不需要一一确定包围框（region proposal）的位置，因此效率较高，但主要有两个缺点:

（1）定位不准确

（2）和基于 region proposal 的方法相比召回率较低。

3.2.1 定位改进——Anchor

YOLOv1 是利用全连接层直接预测 bounding box 的坐标。

YOLOv2 则借鉴了 Faster R-CNN 的思想，引入 anchor。

YOLOv2 做了以下改变:

（1）删掉全连接层和最后一个 pooling 层，使得最后的卷积层可以有更高分辨率的特征;

（2）缩减网络，用 416*416 大小的输入代替原来 448*448。这样做是希望希望得到的特征图都有奇数大小的宽和高，奇数大小的宽和高会使得每个特征图在划分 cell 的时候就只有一个中心 cell。因为大的目标一般会占据图像的中心，所以希望用一个中心 cell 去预测，而不是 4 个中心 cell。网络最终将 416*416 的输入下采样 32 倍变为 13*13 大小的 feature map 输出，查看.cfg 文件可以看到有 8 个 pooling 层。

YOLOv1 中将输入图像分成 7*7 的网格，每个网格预测 2 个 bounding box，一共只有 7*7*2=98 个 box。

YOLOv2 中引入 anchor boxes，输出 feature map 大小为 13*13，每个 cell 有 5 个 anchor box 预测得到 5 个 bounding box，一共有 13*13*5=845 个 box。增加 box 数量是为了提高目标的定位准确率。

Faster R-CNN 中 anchor box 的大小和比例是按经验设定的，然后网络会在训练过程中调整 anchor box 的尺寸。如果一开始就能选择到合适尺寸的 anchor box，那肯定可以帮助网络更好地预测。所以作者采用 k-means 的方式对训练集的 bounding boxes 做聚类，试图找到合适的 anchor box。

作者发现采用标准的 k-means（即用欧式距离来衡量差异），在 box 的尺寸比较大的时候其误差也更大，而我们希望的是误差和 box 的尺寸没有太大关系。所以通过 IOU 定义了距离函数，使得误差和 box 的大小无关：设置先验框的主要目的是为了使得预测框与 ground truth 的 IOU 更好，所以聚类分析师使用 box 与聚类中的 box 之间的 IOU 值作为距离指标。在 VOC 和 COCO 数据集上的聚类分析结果，随着聚类中心数目的增加，平均 IOU 值（各个边界框与聚类中心的 IOU 的平均值）是增加的，但是综合考虑模型复杂度和召回率，作者最终选取 5 个聚类中心作为先验框。

实验对比：

（1）采用聚类分析得到的先验框比手动设置的先验框平均 IOU 值更高，因此模型更容易训练学习。

（2）仅选取 5 种 box 就能达到 Faster RCNN 的 9 种 box 的效果。

3.2.2 分类改进——CNN 特征

YOLOv2 的总体结构，只有卷积层、降采样层（Maxpooling、Averagepooling）以及一个 shortcut 层（细粒度特征）：后端网络结构如下图所示：

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			

Table 6: Darknet-19.

图. 12: backbone

3.2.3 分类改进——多尺度训练

现在基本跑个分类或目标检测模型都不会从随机初始化所有参数开始，所以一般都是用预训练的网络来 fine-tuning 自己的网络，而且预训练的网络基本上都是在 ImageNet 数据集上跑的，一方面数据量大，另一方面训练时间久，而且也比较容易得到。

YOLOv1 在预训练的时候用的是 224*224 的输入，一般预训练的分类模型都

是在 ImageNet 数据集上进行的，然后在检测的时候采用 448*448 的输入。这会导致从分类模型切换到检测模型的时候，模型还要适应图像分辨率的改变。

YOLOv2 中将预训练分成两步：先用 224*224 的输入从头开始训练网络，大概 160 个 epoch，然后再将输入调整到 448*448，再训练 10 个 epoch。** 注意这两步都是在 ImageNet 数据集上操作。** 最后再在检测的数据集上 fine-tuning，也就是检测的时候用 448*448 的图像作为输入就可以顺利过渡了。

YOLOv2 中只有卷积层和池化层，因此不需要固定的输入图片的大小。

为了让模型更有鲁棒性，作者引入了多尺度训练。就是在训练过程中，每迭代一定的次数，改变模型的输入图片大小。

网络输入是 416*416，经过 5 次 max pooling 之后会输出 13*13 的 feature map，也就是下采样 32 倍，因此作者采用 32 的倍数作为输入的 size，具体采用 320、352、384、416、448、480、512、544、576、608 共 10 种 size。

输入图片大小为 320*320 时，特征图大小为 10*10，输入图片大小为 608*608 时，特征图大小为 19*19。每次改变输入图片大小还需要对最后检测层进行处理，然后开始训练。

这种网络训练方式使得相同网络可以对不同分辨率的图像做检测。

在输入 size 较大时，训练速度较慢，在输入 size 较小时，训练速度较快，而 multi-scale training 又可以提高准确率，因此算是准确率和速度都取得一个不错的平衡。

3.2.4 综合改进——细粒度特征

这里添加了一个直通层（passthrough layer），即就是源码中的 reorg layer，将前面一层的 26*26 的特征图和本层 13*13 的特征图进行连接，与 ResNet 网络的 shortcut 类似，以前面更高分辨率的特征图为输入，然后将其连接到后面的低分辨率特征图上。在 13*13 的特征图上做预测，虽然对于大目标已经足够了，但对小目标不一定足够好，这里合并前面大一点的特征图可以有效的检测小目标。具体操作：对于 26*26*512 的特征图，经 passthrough 层处理之后就变成了 13*13*2048 的新特征图（特征图大小变为 1/4，而通道数变为以前的 4 倍），然后与后面的 13*13*1024 特征图连接在一起形成 13*13*3072 的特征图，最后在该特征图上卷积做预测。

3.2.5 综合改进——BN

BN（Batch Normalization）层简单讲就是对网络的每一层的输入都做了归一化，这样网络就不需要每层都去学数据的分布，收敛会快点。作者在 YOLOv2 种

为每个卷积层都添加了 BN 层，由于 BN 可以规范模型，所以加入 BN 后就把 dropout 去掉了，实验证明添加了 BN 层可以提高 2% 的 mAP。

3.3 论文复现

3.3.1 目标

YOLO 是一个多类别的目标检测框架，在复现时，我们使用它来训练一个效果较好的人脸检测器（单类别）。

3.3.2 网络结构

网络的总体结构和之前看到的一致，在每个卷积层后有一个 BN 层，并且使用 leaky Relu 作为激活函数。可以看到网络一共进行了 5 次降采样，每次的大小为 2x2，因此输入图片的大小必须为 32 的倍数。

Layer description	Output size
input	(?, 416, 416, 3)
conv 3x3p1_1 +bnorm leaky	(?, 416, 416, 32)
maxp 2x2p0_2	(?, 208, 208, 32)
conv 3x3p1_1 +bnorm leaky	(?, 208, 208, 64)
maxp 2x2p0_2	(?, 104, 104, 64)
conv 3x3p1_1 +bnorm leaky	(?, 104, 104, 128)
conv 1x1p0_1 +bnorm leaky	(?, 104, 104, 64)
conv 3x3p1_1 +bnorm leaky	(?, 104, 104, 128)
maxp 2x2p0_2	(?, 52, 52, 128)
conv 3x3p1_1 +bnorm leaky	(?, 52, 52, 256)
conv 1x1p0_1 +bnorm leaky	(?, 52, 52, 128)
conv 3x3p1_1 +bnorm leaky	(?, 52, 52, 256)
maxp 2x2p0_2	(?, 26, 26, 256)
conv 3x3p1_1 +bnorm leaky	(?, 26, 26, 512)
conv 1x1p0_1 +bnorm leaky	(?, 26, 26, 256)
conv 3x3p1_1 +bnorm leaky	(?, 26, 26, 512)
conv 1x1p0_1 +bnorm leaky	(?, 26, 26, 256)
conv 3x3p1_1 +bnorm leaky	(?, 26, 26, 512)
maxp 2x2p0_2	(?, 13, 13, 512)
conv 3x3p1_1 +bnorm leaky	(?, 13, 13, 1024)
conv 1x1p0_1 +bnorm leaky	(?, 13, 13, 512)
conv 3x3p1_1 +bnorm leaky	(?, 13, 13, 1024)
conv 1x1p0_1 +bnorm leaky	(?, 13, 13, 512)
conv 3x3p1_1 +bnorm leaky	(?, 13, 13, 1024)
conv 3x3p1_1 +bnorm leaky	(?, 13, 13, 1024)
conv 3x3p1_1 +bnorm leaky	(?, 13, 13, 1024)
concat [16]	(?, 26, 26, 512)
conv 1x1p0_1 +bnorm leaky	(?, 26, 26, 64)
local flatten 2x2	(?, 13, 13, 256)
concat [27, 24]	(?, 13, 13, 1280)
conv 3x3p1_1 +bnorm leaky	(?, 13, 13, 1024)
conv 1x1p0_1 linear	(?, 13, 13, 30)

图. 13: 网络总体结构

3.3.3 数据集

数据集为来自直播平台（喵播）的一些实时数据截图，一共 591581 张图，使用 Face++ 在线 API 进行标记，如下图所示（检测出的人脸区域上至眉心、下至下巴）：

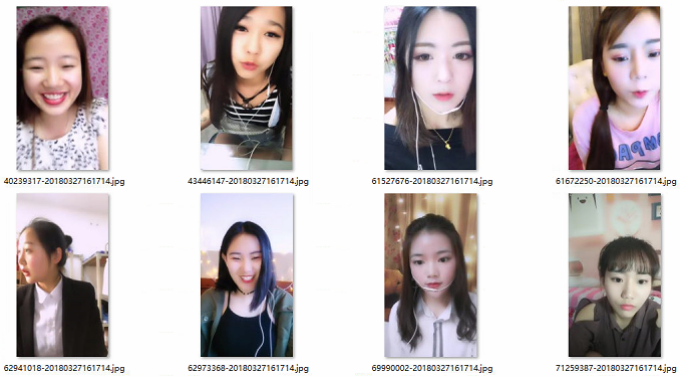


图. 14: 数据集

数据集的标记，即在对应的图片中标记出 Bounding Box，一个框使用四个分量表示，如下图所示：

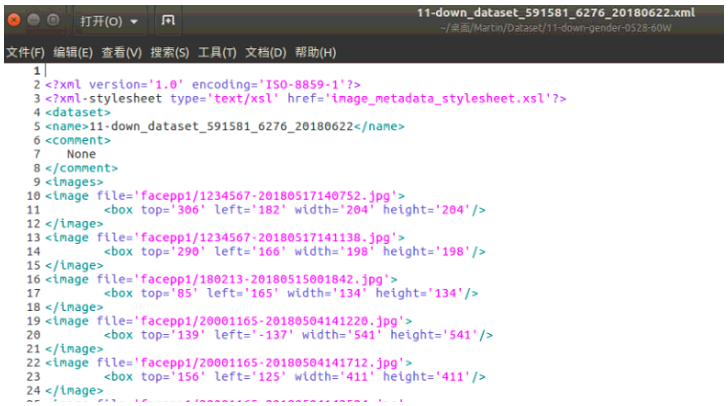


图. 15: 数据集标记

在数据集中，存在遮挡、低头等较难检测的场景，针对这些数据，使用手工标注的方式，如下图所示：

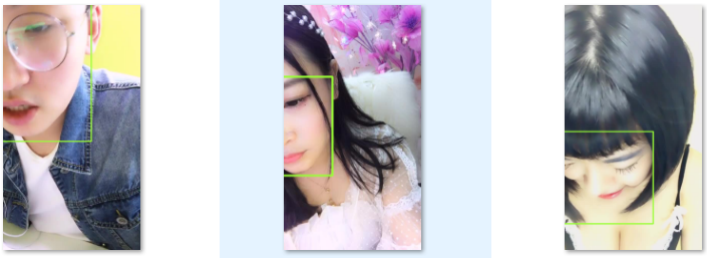


图. 16: 较困难的样本

3.3.4 结果与对比实验

测试的数据集为来自直播平台（喵播）的一些实时数据截图，一共 120044 张图，使用 Face++ 在线 API 进行标记，有人脸的图片为 102680 张，算法的表现如下图所示：

YOLOv2-0806 人脸检测结果		真实值		总计
		有	无	
判定值	有	**102028**	1452	103480
	无	652	**15912**	16564
总计		102680	17364	120044
检测率		**99.37%**	**91.64%**	**98.25%**

图. 17: 训练结果

此外，和一些主流的 SDK 和方案进行了比较，如下图所示：

Dlib-cnn		真实值		总计
人脸检测结果		有	无	
判定值	有	**87031**	14	87045
	无	15649	**17350**	32999
总计		102680	17364	120044
检测率		**84.76%**	**99.92%**	**86.95%**

商汤移动端SDK		真实值		总计
人脸检测结果		有	无	
判定值	有	**79539**	47	79586
	无	23141	**17317**	40458
总计		102680	17364	120044
检测率		**77.46%**	**99.73%**	**80.68%**

Face++在线API		真实值		总计
人脸检测结果		有	无	
判定值	有	**95531**	0	95531
	无	7149	**17364**	24513
总计		102680	17364	120044
检测率		**93.04%**	**100.00%**	**94.04%**

图. 18: 与一些方案的比较

4 几何仿真与可视化

4.1 Anchor 聚类可视化

Anchor 的比例通过 K-Means 针对训练集进行聚类，和常见的聚类中使用 L2 距离不同，此处使用 IOU 作为度量标准，如下所示：

$d(box, centroid) = 1 - IOU(box, centroid)$

针对 VOC 数据集中的 3 个类别 aeroplane、bicycle、bottle，以此为标准针对 BoundingBox 的宽高进行聚类，原始的数据分布如下图所示：

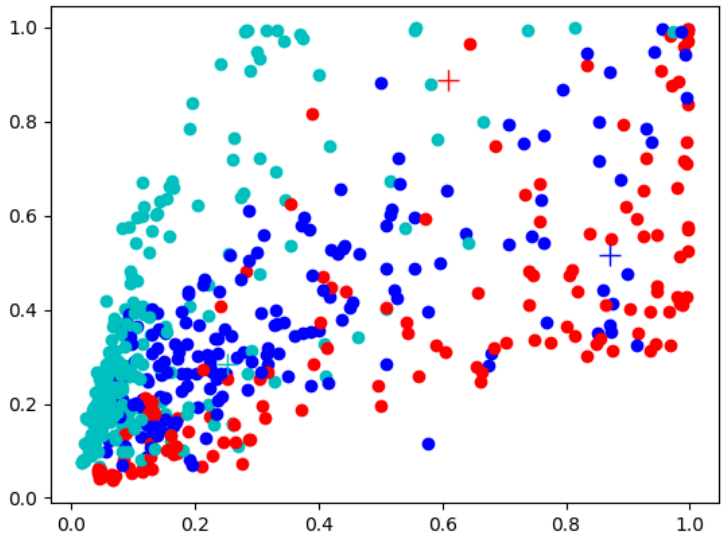


图. 19: 原始数据宽高分布

聚类得到的结果如下图所示：

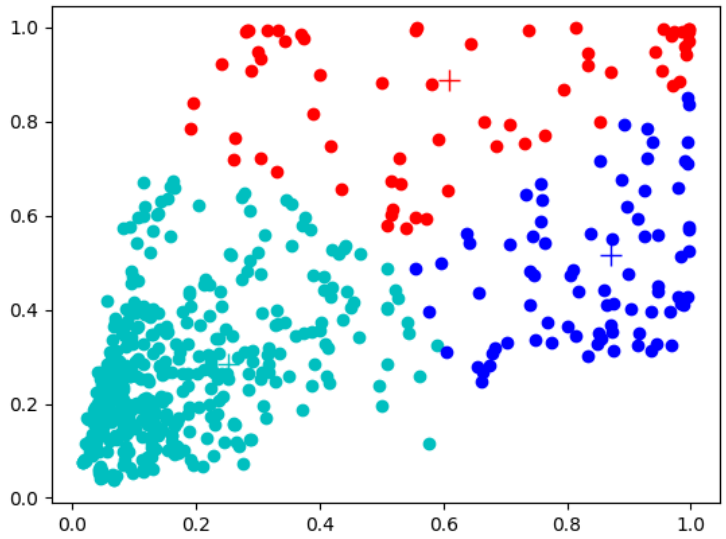


图. 20: 聚类结果作为初始预测框

4.2 模型训练可视化

YOLO 训练的 Loss 函数定义如下，

$loss = \lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B} l_{ij}^{obj}[(x_i-\hat{x_i})^2+(y_i-\hat{y_i})^2]+\lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B} l_{ij}^{obj}[(\sqrt{w_i}-\sqrt{\hat{w_i}})^2+(\sqrt{h_i}-\sqrt{\hat{h_i}})^2]+\sum_{i=0}^{S^2}\sum_{j=0}^{B} l_{ij}^{obj}(c_i-\hat{c_i})^2+\lambda_{noobj}\sum_{i=0}^{S^2}\sum_{j=0}^{B} l_{ij}^{noobj}(c_i-\hat{c_i})^2+\sum_{i=0}^{S^2}\sum_{c\in classes}(p_i(c)-\hat{p_i}(c))^2$

五个部分分别表示中心误差，宽高误差，BoundingBox 置信度误差，类别概率误差。

针对训练过程中 Loss 的变化，其可视化的代码如下所示：

```
import argparse
import sys
import matplotlib.pyplot as plt
def main(argv):
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "-file",
        help = "path to log file"
```

```
        )
    args = parser.parse_args()
    f = open(args.file)

    lines  = [line.rstrip("\n") for line in f.readlines()]

    numbers = {'1','2','3','4','5','6','7','8','9'}
    iters = []
    loss = []

    fig,ax = plt.subplots()
    prev_line = ""
    for line in lines:
        args = line.split(' ')
        if args[0][-1:]==':' and args[0][0] in numbers :
            iters.append(int(args[0][:-1]))
            loss.append(float(args[2]))

    ax.plot(iters,loss)
    plt.xlabel('iters')
    plt.ylabel('loss')
    plt.grid()
    ticks = range(0,250,10)

    #ax.set_yticks(ticks)
    plt.show()

if __name__ == "__main__":
    main(sys.argv)
```

Loss 的变化如下图所示:

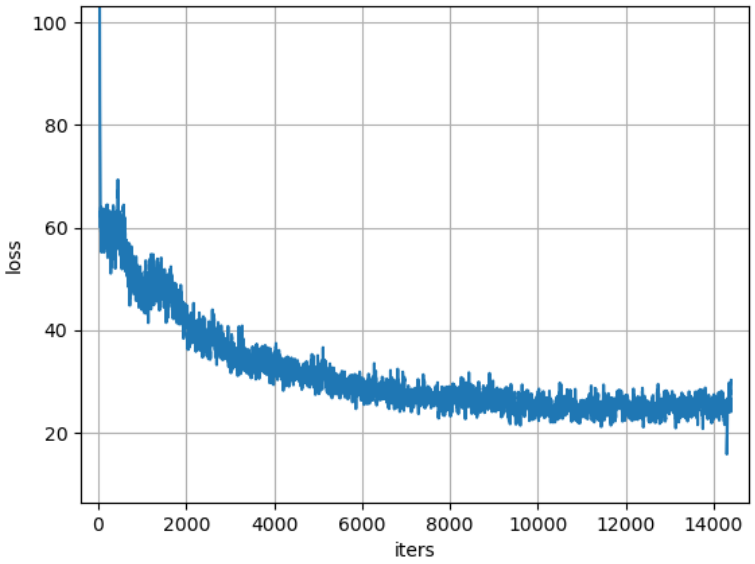


图. 21: YOLO 训练过程中 Loss 变化曲线

参考文献

[1] Redmon J, Divvala S, Girshick R, et al. You only look once: Unified, real-time object detection[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 779-788.

[2] Redmon J, Farhadi A. YOLO9000: better, faster, stronger[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 7263-7271.

[3] Redmon J, Farhadi A. Yolov3: An incremental improvement[J]. arXiv preprint arXiv:1804.02767, 2018.