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

YOLO：统一的实时目标检测

Joseph Redmon*, Santosh Divvala*†, Ross Girshick¶, Ali Farhadi*†
University of Washington*, Allen Institute for AI†, Facebook AI Research¶
http://pjreddie.com/yolo/

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.

摘要

我们提出了 YOLO，一种新的目标检测方法。以前的目标检测工作重复利用分类器来完成检测任务。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整个图像上预测边界框和类别概率。由于整个检测流水线是单一网络，因此可以直接对检测性能进行端到端的优化。

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.

我们的统一架构非常快。我们的基础 YOLO 模型以 45 帧/秒的速度实时处理图像。Fast YOLO 是 YOLO 的一个较小版本，每秒能处理惊人的 155 帧图像，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 虽然存在较多的定位错误，但很少将背景预测成假阳性（译者注：其它先进的目标检测算法将背景预测成目标的概率较大）。最后，YOLO 能学习到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. Introduction

Humans glance at an image and instantly know what objects are in the image, where they are, and how they interact. The human visual system is fast and accurate, allowing us to perform complex tasks like driving with little conscious thought. Fast, accurate algorithms for object detection would allow computers to drive cars without specialized sensors, enable assistive devices to convey real-time scene information to human users, and unlock the potential for general purpose, responsive robotic systems.

1. 引言

人们瞥一眼图像，立即知道图像中的物体是什么，它们在哪里以及它们如何相互作用。人类的视觉系统是快速和准确的，使我们能够执行复杂的任务，例如如驾驶车辆时不会刻意地进行思考或思想。快速、准确的目标检测算法可以让计算机在没有专用传感器的情况下驾驶汽车，使辅助设备能够向人类用户传达实时的场景信息，并具有解锁通用目的和响应机器人系统的潜力。

Current detection systems repurpose classifiers to perform detection. To detect an object, these systems take a classifier for that object and evaluate it at various locations and scales in a test image. Systems like **deformable parts models** (DPM) use a sliding window approach where the classifier is run at evenly spaced locations over the entire image [10].

目前的检测系统重复利用分类器来执行检测。为了检测目标，这些系统为该目标提供一个分类器，并在不同的位置对其进行评估，并在测试图像中进行缩放。像**可变形部件模型**（DPM）这样的系统使用滑动窗口方法，其分类器在整个图像的均匀间隔的位置上运行[10]。

More recent approaches like R-CNN use **region proposal** methods to first generate potential bounding boxes in an image and then run a classifier on these proposed boxes. After classification, post-processing is used to refine the bounding boxes, eliminate duplicate detections, and rescore the boxes based on other objects in the scene [13]. These complex pipelines are slow and hard to optimize because each individual component must be trained separately.

最近的许多方法，如 R-CNN 使用 region proposal 方法首先在图像中生成潜在的边界框，然后在这些提出的框上运行分类器。在分类之后，通过后处理对边界框进行修正，消除重复的检测，并根据场景中的其它目标重新定位边界框[13]。这些复杂的流程很慢，很难优化，因为每个单独的组件都必须单独进行训练。

We reframe object detection as a single regression problem, straight from image pixels to bounding box coordinates and class probabilities. Using our system, you only look once (YOLO) at an image to predict what objects are present and where they are.

我们将目标检测重构并看作为单一的回归问题，直接从图像像素到边界框坐标和类别概率。使用我们的系统，您只需要在图像上看一次（you only look once, YOLO），以预测出现的目标和位置。

YOLO is refreshingly simple: see Figure 1. A single convolutional network simultaneously predicts multiple bounding boxes and class probabilities for those boxes. YOLO trains on full images and directly optimizes detection performance. This unified model has several benefits over traditional methods of object detection.

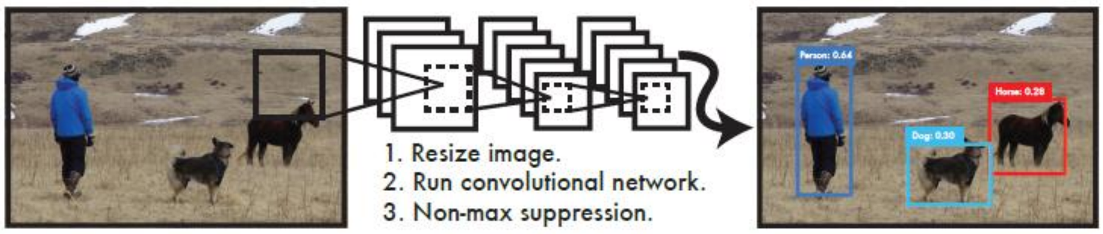


Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448 × 448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

YOLO 新奇又很简单：如图 1 所示。单个卷积网络同时预测这些框的多个边界框和类别概率值。YOLO 在全图像上训练并直接优化检测性能。这种统一的模型比传统的目标检测方法有一些好处。

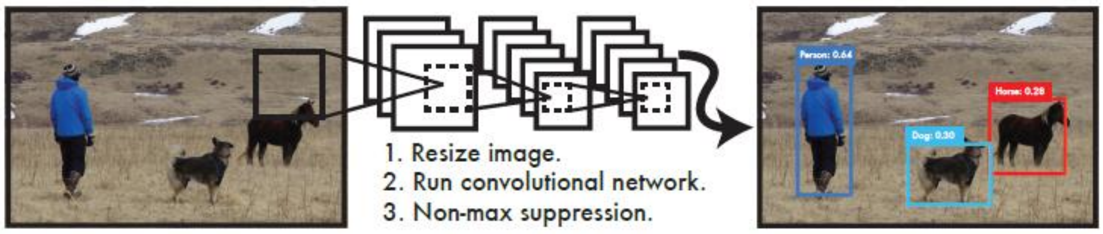


图 1：YOLO 检测系统。用 YOLO 处理图像简单直接。我们的系统（1）将输入图像调整为 448×448，（2）在图像上运行单个卷积网络，以及（3）由模型的置信度对所得到的检测进行阈值处理。

First, YOLO is extremely fast. Since we frame detection as a regression problem we don’t need a complex pipeline. We simply run our neural network on a new image at test time to predict detections. Our base network runs at 45 frames per second with no batch processing on a Titan X GPU and a fast version runs at more than 150 fps. This means we can process streaming video in real-time with less than 25 milliseconds of latency. Furthermore, YOLO achieves more than twice the **mean average precision** of other real-time systems. For a demo of our system running in real-time on a webcam please see our project webpage: http://pjreddie.com/yolo/.

首先，YOLO 速度非常快。由于我们将检测视为回归问题，所以我们不需要复杂的流程。测试时我们在一张新图像上简单的运行我们的神经网络来预测检测。我们的基础网络以每秒 45 帧的速度运行，在 Titan X GPU 上没有批处理，快速版本运行速度超过 150fps。这意味着我们可以在不到 25 毫秒的延迟内实时处理流媒体视频。此外，

YOLO 实现了其它实时系统两倍以上的 mAP。关于我们的系统在网络摄像头上实时运行的演示，请参阅我们的项目网页：http://pjreddie.com/yolo/。

Second, YOLO reasons globally about the image when making predictions. Unlike sliding window and region proposal-based techniques, YOLO sees the entire image during training and test time so it implicitly encodes contextual information about classes as well as their appearance. Fast R-CNN, a top detection method [14], mistakes background patches in an image for objects because it can’t see the larger context. YOLO makes less than half the number of background errors compared to Fast R-CNN.

其次，YOLO 在进行预测时，会对图像进行全局地推理。与基于滑动窗口和 region proposal 的技术不同，YOLO 在训练期间和测试时会看到整个图像，所以它隐式地编码了关于类的上下文信息以及它们的外形。Fast R-CNN 是一种顶级的检测方法[14]，但因为它看不到更大的上下文，所以在图像中会将背景块误检为目标。与 Fast R-CNN 相比，YOLO 的背景误检数量少了一半。

Third, YOLO learns generalizable representations of objects. When trained on natural images and tested on artwork, YOLO outperforms top detection methods like DPM and R-CNN by a wide margin. Since YOLO is highly generalizable it is less likely to break down when applied to new domains or unexpected inputs.

第三，YOLO 学习目标的泛化表示。当在自然的图像上进行训练并对艺术作品进行测试时，YOLO 大幅优于 DPM 和 R-CNN 等顶级检测方法。由于 YOLO 具有高度泛化能力，因此在应用于新领域或碰到非正常输入时很少出故障。

YOLO still lags behind state-of-the-art detection systems in accuracy. While it can quickly identify objects in images it struggles to precisely localize some objects, especially small ones. We examine these tradeoffs further in our experiments.

YOLO 在准确度上仍然落后于最先进的检测系统。虽然它可以快速识别图像中的目标，但它仍在努力精确定位一些目标，尤其是一些小目标。我们在实验中会进一步检查这些权衡。

All of our training and testing code is open source. A variety of pretrained models are also available to download.

我们所有的训练和测试代码都是开源的。各种预训练模型也都可以下载。

2. Unified Detection

We unify the separate components of object detection into a single neural network. Our network uses features from the entire image to predict each bounding box. It also predicts all bounding boxes across all classes for an image simultaneously. This means our network reasons globally about the full image and all the objects in the image. The YOLO design enables end-to-end training and real-time speeds while maintaining high average precision.

2. 统一的检测

我们将目标检测的单独组件集成到单个神经网络中。我们的网络使用整个图像的特征来预测每个边界框。它还可以同时预测一张图像中的所有类别的所有边界框。这意味着我们的网络全面地推理整张图像和图像中的所有目标。YOLO 设计可实现端到端训练和实时的速度，同时保持较高的平均精度。

Our system divides the input image into an S×S grid. If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object.

我们的系统将输入图像分成 S×S 的网格。如果一个目标的中心落入一个网格单元中，该网格单元负责检测该目标。

Each grid cell predicts B bounding boxes and confidence scores for those boxes. These confidence scores reflect how confident the model is that the box contains an object and also how accurate it thinks the box is that it predicts. Formally we define confidence as Pr(Object)∗IOUtruthpred. If no object exists in that cell, the confidence scores should be zero. Otherwise we want the confidence score to equal the intersection over union (IOU) between the predicted box and the ground truth.

每个网格单元预测这些盒子的 B 个边界框和置信度分数。这些置信度分数反映了该模型对盒子是否包含目标的置信度，以及它预测盒子的准确程度。在形式上，我们将置信度定义为Pr(Object)∗IOUtruthpred。如果该单元格中不存在目标，则置信度分数应为零。否则，我们希望置信度分数等于预测框与真实值之间联合部分的交集（IOU）。

Each bounding box consists of 5 predictions: x, y, w, h, and confidence. The (x,y) coordinates represent the center of the box relative to the bounds of the grid cell. The width and height are predicted relative to the whole image. Finally the confidence prediction represents the IOU between the predicted box and any ground truth box.

每个边界框包含 5 个预测：x、y、w、h 和置信度。(x，y)坐标表示边界框相对于网格单元边界框的中心。宽度和高度是相对于整张图像预测的。最后，置信度预测表示预测框与实际边界框之间的 IOU。

Each grid cell also predicts C conditional class probabilities, Pr(Classi|Object). These probabilities are conditioned on the grid cell containing an object. We only predict one set of class probabilities per grid cell, regardless of the number of boxes BB.

每个网格单元还预测 C 个条件类别概率 Pr(Classi|Object)。这些概率以包含目标的网格单元为条件。每个网格单元我们只预测的一组类别概率，而不管边界框的的数量 B 是多少。

At test time we multiply the conditional class probabilities and the individual box confidence predictions,

$$\Pr(\text{Class}_i|\text{Object}) * \Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}} = \Pr(\text{Class}_i) * \text{IOU}_{\text{pred}}^{\text{truth}} \quad (1)$$

which gives us class-specific confidence scores for each box. These scores encode both the probability of that class appearing in the box and how well the predicted box fits the object.

在测试时，我们乘以条件类概率和单个盒子的置信度预测值：

$$\Pr(\text{Class}_i|\text{Object}) * \Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}} = \Pr(\text{Class}_i) * \text{IOU}_{\text{pred}}^{\text{truth}} \quad (1)$$

它为我们提供了每个框特定类别的置信度分数。这些分数编码了该类出现在框中的概率以及预测框拟合目标的程度。

For evaluating YOLO on Pascal VOC, we use S=7, B=2. Pascal VOC has 20 labelled classes so C=20. Our final prediction is a 7×7×30 tensor.

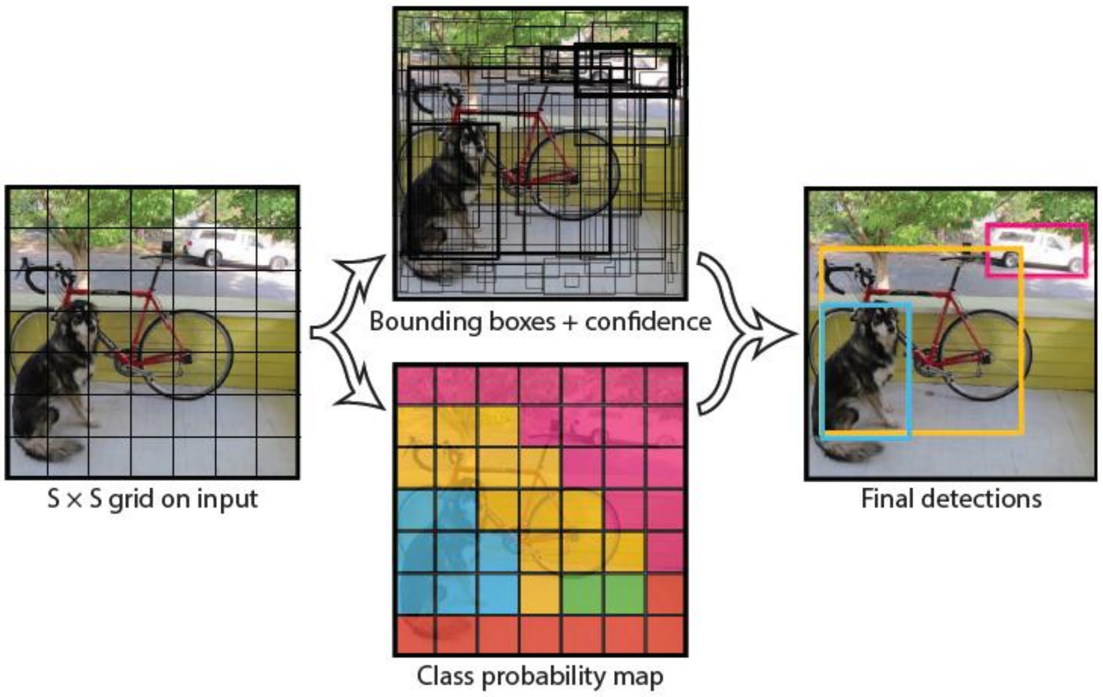


Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an S×S grid and for each grid cell predicts B bounding boxes,

confidence for those boxes, and C class probabilities. These predictions are encoded as an S×S×(B∗5+C) tensor.

为了在 Pascal VOC 上评估 YOLO，我们使用 S=7，B=2。Pascal VOC 有 20 个标注类，所以 C=20。我们最终的预测是 7×7×30 的张量。

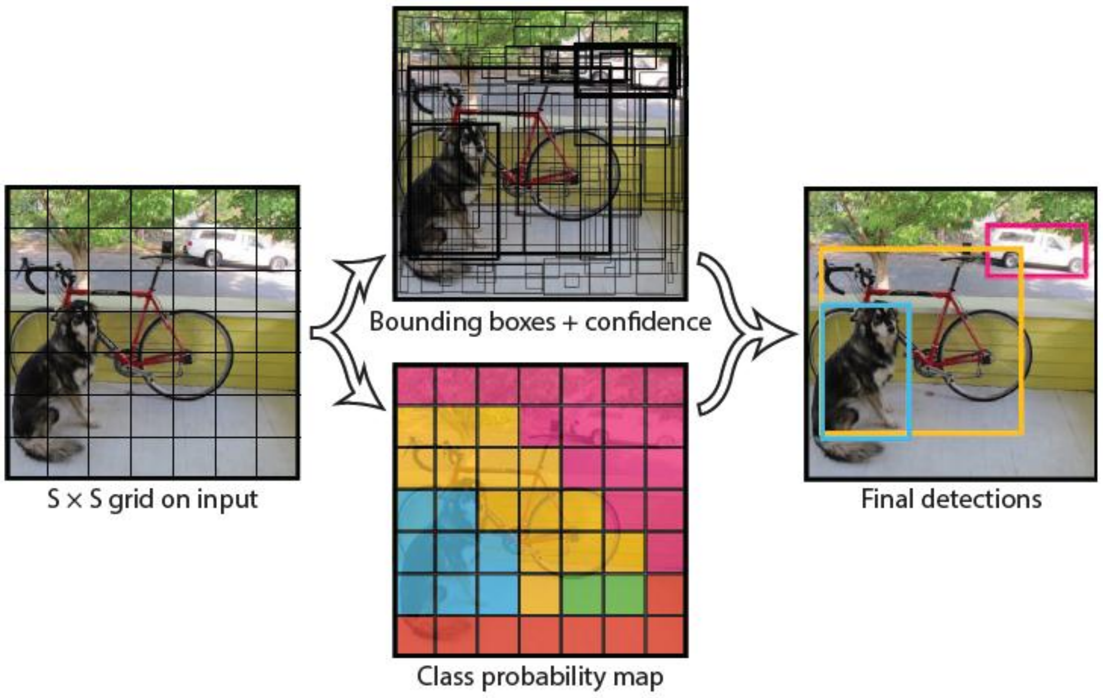


图 2：模型。我们的系统将检测任务构建为回归问题。它将图像分成 S×S 的网格，并且对于每个网格单元都预测 B 个边界框、这些边界框的置信度以及 C 个类别概率。这些预测结果被编码为 S×S×(B∗5+C)的张量。

2.1. Network Design

We implement this model as a convolutional neural network and evaluate it on the Pascal VOC detection dataset [9]. The initial convolutional layers of the network extract features from the image while the fully connected layers predict the output probabilities and coordinates.

2.1. 网络设计

我们将此模型实现为卷积神经网络，并在 Pascal VOC 检测数据集[9]上进行评估。网络的初始卷积层从图像中提取特征，而全连接层预测输出概率和坐标。

Our network architecture is inspired by the GoogLeNet model for image classification [34]. Our network has 24 convolutional layers followed by 2 fully connected layers. Instead of the inception modules used by GoogLeNet, we simply use 1×1 reduction layers followed by 3×3 convolutional layers, similar to Lin et al [22]. The full network is shown in Figure 3.

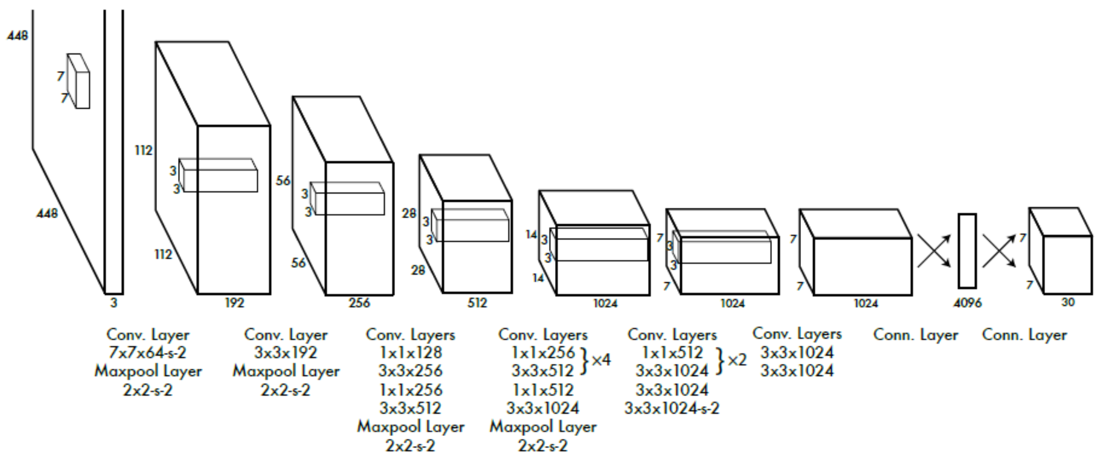

448
7
7
448
3
112
3
3
112
192
56
3
3
56
256
28
3
3
28
512
14
3
3
14
1024
7
3
3
7
1024
7
7
1024
4096
7
7
30
Conv. Layer 7x7x64-s-2 Maxpool Layer 2x2-s-2
Conv. Layer 3x3x192 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x128 3x3x256 1x1x256 3x3x512 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x256 3x3x512 ×4 1x1x512 3x3x1024 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x512 3x3x1024 ×2 3x3x1024 3x3x1024-s-2
Conv. Layers 3x3x1024 3x3x1024
Conn. Layer
Conn. Layer

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.

我们的网络架构受到 GoogLeNet 图像分类模型的启发[34]。我们的网络有 24 个卷积层，后面是 2 个全连接层。我们只使用 1×1 降维层，后面是 3×3 卷积层，这与 Lin 等人[22]的模型结构类似，而不是 GoogLeNet 使用的 Inception 模块。完整的网络如图 3 所示。

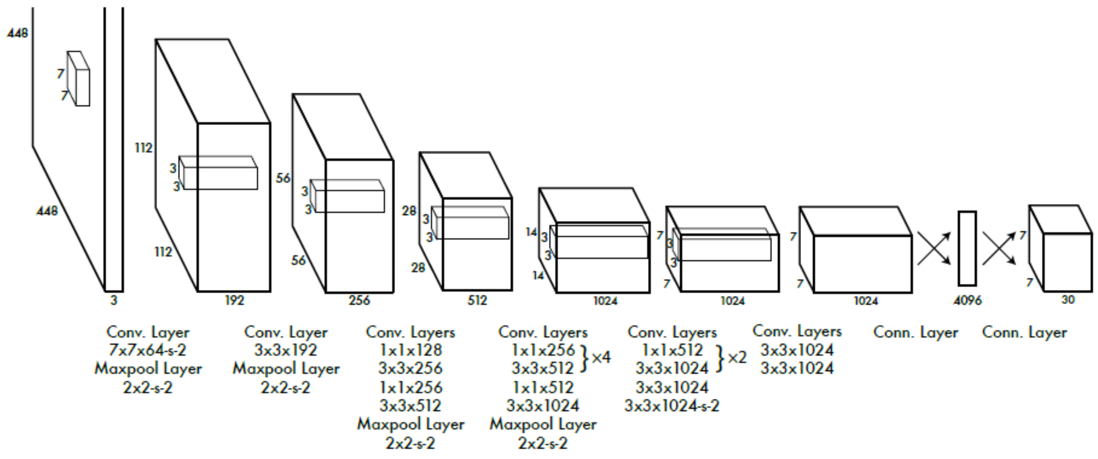


图 3：模型架构。我们的检测网络有 24 个卷积层，其次是 2 个全连接层。交替使用 1×1 卷积层减少了前面层的特征空间。我们在 ImageNet 分类任务上以一半的分辨率（224×224 的输入图像）预训练卷积层，然后将分辨率加倍来进行检测。

We also train a fast version of YOLO designed to push the boundaries of fast object detection. Fast YOLO uses a neural network with fewer convolutional layers (9 instead of 24) and fewer filters in those layers. Other than the size of the network, all training and testing parameters are the same between YOLO and Fast YOLO.

我们还训练了快速版本的 YOLO，旨在挑战快速目标检测的界限。快速 YOLO 使用具有较少卷积层（9 层而非 24 层）的神经网络，在这些层中使用较少的卷积核。除了网络规模之外，YOLO 和快速 YOLO 的所有训练和测试参数都是相同的。

The final output of our network is the 7×7×30 tensor of predictions.

我们网络的最终输出是 7×7×30 的预测张量。

2.2. Training

We pretrain our convolutional layers on the ImageNet 1000-class competition dataset [30]. For pretraining we use the first 20 convolutional layers from Figure 3 followed by a average-pooling layer and a fully connected layer. We train this network for approximately a week and achieve a single crop top-5 accuracy of 88% on the ImageNet 2012 validation set, comparable to the GoogLeNet models in Caffe's Model Zoo [24]. We use the Darknet framework for all training and inference [26].

2.2. 训练

我们在 ImageNet 1000 类竞赛数据集[30]上预训练我们的卷积层。对于预训练，我们使用图 3 中的前 20 个卷积层，接着是平均池化层和全连接层。我们对这个网络进行了大约一周的训练，并且在 ImageNet 2012 验证集上获得了单一裁剪图像 88%的 top-5 准确率，与 Caffe 模型池中的 GoogLeNet 模型相当。我们使用 Darknet 框架进行所有的训练和推断[26]。

We then convert the model to perform detection. Ren et al. show that adding both convolutional and connected layers to pretrained networks can improve performance [29]. Following their example, we add four convolutional layers and two fully connected layers with randomly initialized weights. Detection often requires fine-grained visual information so we increase the input resolution of the network from 224×224 to 448×448.

然后我们转换模型来进行检测任务。Ren 等人表明，预训练网络中增加卷积层和连接层可以提高性能[29]。按照他们的例子，我们添加了四个卷积层和两个全连接层，并且对权重进行随机初始化。检测

通常需要细粒度的视觉信息，因此我们将网络的输入分辨率从 224×224 变为 448×448。

Our final layer predicts both class probabilities and bounding box coordinates. We normalize the bounding box width and height by the image width and height so that they fall between 0 and 1. We parametrize the bounding box x and y coordinates to be offsets of a particular grid cell location so they are also bounded between 0 and 1.

我们的最后一层预测类别概率和边界框坐标。我们通过图像宽度和高度来归一化边界框的宽度和高度，使它们落在 0 和 1 之间。我们将边界框 x 和 y 坐标参数化为特定网格单元位置的偏移量，所以它们边界也在 0 和 1 之间。

We use a linear activation function for the final layer and all other layers use the following leaky rectified linear activation:

$$\phi(x) = \begin{cases} x, & \text{if } x > 0 \\ 0.1x, & \text{otherwise} \end{cases} \qquad (2)$$

我们对最后一层使用线性激活函数，所有其它层使用下面的 leaky ReLU 激活函数：

$$\phi(x) = \begin{cases} x, & \text{if } x > 0 \\ 0.1x, & \text{otherwise} \end{cases} \qquad (2)$$

We optimize for sum-squared error in the output of our model. We use sum-squared error because it is easy to optimize, however it does not perfectly align with our goal of maximizing average precision. It weights localization error equally with classification error which may not be ideal.

Also, in every image many grid cells do not contain any object. This pushes the “confidence” scores of those cells towards zero, often overpowering the gradient from cells that do contain objects. This can lead to model instability, causing training to diverge early on.

我们优化了模型输出的平方和误差。我们使用平方和误差是因为它很容易进行优化，但是它并不完全符合我们最大化平均精度的目标。分类误差与定位误差的权重是一样的，这可能并不理想。另外，在每张图像中，许多网格单元不包含任何对象。这将导致这些单元格的“置信度”分数为零，通常压倒了包含目标的单元格的梯度。这可能导致模型不稳定，从而导致训练过早发散。

To remedy this, we increase the loss from bounding box coordinate predictions and decrease the loss from confidence predictions for boxes that don’t contain objects. We use two parameters, λcoord and λnoobj to accomplish this. We set λcoord=5λcoord=5 and λnoobj=.5λnoobj=.5.

为了改善这一点，我们增加了边界框坐标预测的损失，并减少了不包含目标边界框的置信度预测的损失。我们使用两个参数 λcoord 和 λnoobj 来完成这个调整工作。我们设置 λcoord=5 和 λnoobj=0.5。

Sum-squared error also equally weights errors in large boxes and small boxes. Our error metric should reflect that small deviations in large boxes matter less than in small boxes. To partially address this we predict the square root of the bounding box width and height instead of the width and height directly.

平方和误差也可以在大盒子和小盒子中同样加权误差。我们的误差指标应该反映出，大盒子中小偏差的重要性不如小盒子中小偏差的重要性。为了部分解决这个问题，我们直接预测边界框宽度和高度的平方根，而不是宽度和高度。

YOLO predicts multiple bounding boxes per grid cell. At training time we only want one bounding box predictor to be responsible for each object. We assign one predictor to be “responsible” for predicting an object based on which prediction has the highest current IOU with the ground truth. This leads to specialization between the bounding box predictors. Each predictor gets better at predicting certain sizes, aspect ratios, or classes of object, improving overall recall.

YOLO 每个网格单元预测多个边界框。在训练时，每个目标我们只需要一个边界框预测器来负责。我们根据哪个预测器的预测值与真实值之间具有当前最高的 IOU 来指定哪个预测器“负责”预测该目标。这导致边界框预测器之间的专一化。每个预测器可以更好地预测特定大小、长宽比或目标的类别，从而改善整体召回率。

During training we optimize the following, multi-part loss function:

$$
\begin{aligned}
&\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2 \qquad (3)
\end{aligned}
$$

where 1obji denotes if object appears in cell I and 1objij denotes that the jth bounding box predictor in cell I is “responsible” for that prediction.

在训练期间，我们优化以下由多部分组成的损失函数：

$$
\begin{aligned}
&\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2 \qquad (3)
\end{aligned}
$$

其中 1obji 表示目标是否出现在网格单元 i 中，1objij 表示网格单元 i 中的第 j 个边界框预测器“负责”该预测。

Note that the loss function only penalizes classification error if an object is present in that grid cell (hence the conditional class probability discussed earlier). It also only penalizes bounding box coordinate error if that predictor is “responsible” for the ground truth box (i.e. has the highest IOU of any predictor in that grid cell).

注意，如果目标存在于该网格单元中（前面讨论的条件类别概率），则损失函数仅惩罚分类误差。如果预测器“负责”真实边界框（即该网格单元中具有最高 IOU 的预测器），则它也仅惩罚边界框坐标误差。

We train the network for about 135 epochs on the training and validation data sets from Pascal VOC 2007 and 2012. When testing on 2012 we also include the VOC 2007 test data for training. Throughout training we use a batch size of 64, a momentum of 0.9 and a decay of 0.0005.

我们在 Pascal VOC 2007 和 2012 的训练和验证数据集上进行了大约 135 个迭代周期的网络训练。在 Pascal VOC 2012 上进行测试时，我们的训练包含了 Pascal VOC 2007 的测试数据。在整个训练过程中，我们使用了 64 的批大小，0.9 的动量和 0.0005 的衰减率。

Our learning rate schedule is as follows: For the first epochs we slowly raise the learning rate from 10^{-3} to 10^{-2}. If we start at a high learning rate our model often diverges due to unstable gradients. We continue training with 10^{-2} for 75 epochs, then 10^{-3} for 30 epochs, and finally 10^{-4} for 30 epochs.

我们的学习率方案如下：对于第一个迭代周期，我们慢慢地将学习率从 10^{-3} 提高到 10^{-2}。如果我们从高学习率开始，我们的模型往往会由于梯度不稳定而发散。我们继续以 10^{-2} 的学习率训练 75 个迭代周期，然后用 10^{-3} 的学习率训练 30 个迭代周期，最后用 10^{-4} 的学习率训练 30 个迭代周期。

To avoid overfitting we use dropout and extensive data augmentation. A dropout layer with rate=.5 after the first connected layer prevents co-adaptation between layers [18]. For data augmentation we introduce random scaling and translations of up to 20% of the original image size. We also randomly adjust the exposure and saturation of the image by up to a factor of 1.5 in the HSV color space.

为了避免过度拟合，我们使用 dropout 和大量的数据增强。在第一个连接层之后，dropout 层使用 rate=0.5 的比例，防止层之间的互相适应[18]。对于数据增强，我们引入高达原始图像 20%大小的随机缩放和转换。我们还在 HSV 色彩空间中使用高达 1.5 的因子来随机调整图像的曝光和饱和度。

2.3. Inference

Just like in training, predicting detections for a test image only requires one network evaluation. On Pascal VOC the network predicts 98 bounding boxes per image and class probabilities for each box. YOLO is extremely fast at test time since it only requires a single network evaluation, unlike classifier-based methods.

2.3. 推断

与训练时一样，预测测试图像的检测只需要一次网络评估。在 Pascal VOC 上，每张图像上网络预测 98 个边界框（译者注：每张图像被划分成 7*7 的格子，每个格子预测两个边界框，总共 98 个边界框）和每个框的类别概率。YOLO 在测试时非常快，因为它只需要运行一次网络评估，不像基于分类器的方法。

The grid design enforces spatial diversity in the bounding box predictions. Often it is clear which grid cell an object falls in to and the network only predicts one box for each object. However, some large objects or objects near the border of multiple cells can be well localized by multiple cells. Non-maximal suppression can be used to fix these multiple detections. While not critical to performance as it is for R-CNN or DPM, non-maximal suppression adds 2−3% in mAP.

网格设计强化了边界框预测中的空间多样性。通常很明显一个目标落在哪一个网格单元中，而网络只能为每个目标预测一个边界框。然而，一些大的目标或靠近多个网格单元边界的目标可以被多个网格单元很好地定位。非极大值抑制（译者注：NMS）可以用来修正这些多重检测。对于 R-CNN 或 DPM 而言，虽然非极大值抑制对性能影响不大，但会增加 2−3%的 mAP。

2.4. Limitations of YOLO

YOLO imposes strong spatial constraints on bounding box predictions since each grid cell only predicts two boxes and can only have one class. This spatial constraint limits the number of nearby objects that our model can predict. Our model struggles with small objects that appear in groups, such as flocks of birds.

2.4. YOLO 的缺点

YOLO 对边界框预测强加空间约束，因为每个网格单元只预测两个框，只能有一个类别。这个空间约束限制了我们的模型可以预测的邻近目标的数量。我们的模型对成群出现的小物体（比如鸟群）预测效果较差。

Since our model learns to predict bounding boxes from data, it struggles to generalize to objects in new or unusual aspect ratios or configurations. Our model also uses relatively coarse features for predicting bounding boxes since our architecture has multiple downsampling layers from the input image.

由于我们的模型学习从数据中预测边界框，因此它很难泛化到新的、不常见的长宽比或配置中的目标。我们的模型也使用相对较粗糙的特征来预测边界框，因为我们的架构具有来自输入图像的多个下采样层。

Finally, while we train on a loss function that approximates detection performance, our loss function treats errors the same in small bounding boxes versus large bounding boxes. A small error in a large box is generally benign but a small error in a small box has a much greater effect on IOU. Our main source of error is incorrect localizations.

最后，当我们训练一个近似检测性能的损失函数时，我们的损失函数会同样的对待小边界框与大边界框的误差。大边界框的小误差通常是良性的，但小边界框的小误差对 IOU 的影响要大得多。我们的主要误差来源是定位误差。

3. Comparison to Other Detection Systems

Object detection is a core problem in computer vision. Detection pipelines generally start by extracting a set of robust features from input images (Haar [25], SIFT [23], HOG [4], convolutional features [6]). Then, classifiers [36, 21, 13, 10] or localizers [1, 32] are used to identify objects in the feature space. These classifiers or localizers are run either in sliding window fashion over the whole image or on some subset of regions in the image [35, 15, 39]. We compare the YOLO detection system to several top detection frameworks, highlighting key similarities and differences.

3. 与其它检测系统的比较

目标检测是计算机视觉中的核心问题。检测流程通常从输入图像上提取一组鲁棒特征（Haar [25]，SIFT [23]，HOG [4]，卷积特征[6]）开始。然后，分类器[36,21,13,10]或定位器[1,32]被用来识别特征空间中的目标。这些分类器或定位器在整个图像上或在图像中的一些子区域上以滑动窗口的方式运行[35,15,39]。我们将 YOLO 检测系统与几种顶级检测框架进行比较，以突出了主要的相似性和差异性。

Deformable parts models. Deformable parts models (DPM) use a sliding window approach to object detection [10]. DPM uses a disjoint pipeline to extract static features, classify regions, predict bounding boxes for high scoring regions, etc. Our system replaces all of these disparate parts with a single convolutional neural network. The network performs feature extraction, bounding box prediction, non-maximal suppression, and contextual reasoning all concurrently. Instead of static features, the network trains the features in-line and optimizes them for the detection task. Our unified architecture leads to a faster, more accurate model than DPM.

可变形部件模型（DPM）。可变形零件模型（DPM）使用滑动窗口方法进行目标检测[10]。DPM 使用不相交的流程来提取静态特征，对区域进行分类，预测高评分区域的边界框等。我们的系统用单个卷积神经网络替换所有这些不同的部分。网络同时进行特征提取、边界框预测、非极大值抑制和上下文推理。网络内嵌训练特征而不是静态特征，并优化它们完成检测任务。我们的统一架构获得了比 DPM 更快、更准确的模型。

R-CNN. R-CNN and its variants use region proposals instead of sliding windows to find objects in images. Selective Search [35] generates potential bounding boxes, a convolutional network extracts features, an SVM scores the boxes, a linear model adjusts the bounding boxes, and non-max suppression eliminates duplicate detections. Each stage of this complex pipeline must be precisely tuned independently and the resulting system is very slow, taking more than 40 seconds per image at test time [14].

R-CNN。R-CNN 及其变种使用 region proposals 而不是滑动窗口来查找图像中的目标。Selective Search [35]产生潜在的边界框、卷积网络提取特征、SVM 对边界框进行评分、线性模型调整边界框、非极大值抑制消除重复检测。这个复杂流程的每个阶段都必须独立地进行精确调整，所得到的系统非常慢，测试时每张图像需要超过 40 秒[14]。

YOLO shares some similarities with R-CNN. Each grid cell proposes potential bounding boxes and scores those boxes using convolutional features. However, our system puts spatial constraints on the grid cell proposals which helps mitigate multiple detections of the same object. Our system also proposes far fewer bounding boxes, only 98 per image compared to about 2000 from Selective Search. Finally, our system combines these individual components into a single, jointly optimized model.

YOLO 与 R-CNN 有一些相似之处。每个网格单元提出潜在的边界框并使用卷积特征对这些框进行评分。但是，我们的系统对网格单元提出进行了空间限制，这有助于缓解对同一目标的多次检测。我们的系统还提出了更少的边界框，每张图像只有 98 个，而 Selective Search 则需要 2000 个左右。最后，我们的系统将这些单独的组件组合成一个单一的、共同优化的模型。

Other Fast Detectors. Fast and Faster R-CNN focus on speeding up the R-CNN framework by sharing computation and using neural networks to propose regions instead of Selective Search [14] [28]. While they offer speed and accuracy improvements over R-CNN, both still fall short of real-time performance.

其它快速检测器。Fast 和 Faster R-CNN 通过共享计算和使用神经网络替代 Selective Search 来提出区域加速 R-CNN 框架[14] [28]。虽然它们提供了比 R-CNN 更快的速度和更高的准确度，但两者仍然不能达到实时性能。

Many research efforts focus on speeding up the DPM pipeline [31] [38] [5]. They speed up HOG computation, use cascades, and push computation to GPUs. However, only 30Hz DPM [31] actually runs in real-time.

许多研究工作集中在加快 DPM 流程上[31] [38] [5]。它们加速 HOG 计算，使用级联，并将计算推动到 GPU 上。但是，实际上 DPM [31]实时运行只达到 30Hz。

Instead of trying to optimize individual components of a large detection pipeline, YOLO throws out the pipeline entirely and is fast by design.

YOLO 不是试图优化大型检测流程的单个组件，而是完全抛弃流程，为提升检测速度而重新设计。

Detectors for single classes like faces or people can be highly optimized since they have to deal with much less variation [37]. YOLO is a general purpose detector that learns to detect a variety of objects simultaneously.

像人脸或行人等单类别的检测器可以被高度优化，因为他们只需处理更少的多样性[37]。YOLO 是一种通用的检测器，可以学习同时检测不同的多个目标。

Deep MultiBox. Unlike R-CNN, Szegedy et al. train a convolutional neural network to predict regions of interest [8] instead of using Selective Search. MultiBox can also perform single object detection by replacing the confidence prediction with a single class prediction. However, MultiBox cannot perform general object detection and is still just a piece in a larger detection pipeline, requiring further image patch classification. Both YOLO and MultiBox use a convolutional network to predict bounding boxes in an image but YOLO is a complete detection system.

Deep MultiBox。与 R-CNN 不同，Szegedy 等人训练了一个卷积神经网络来预测感兴趣区域（ROI）[8]，而不是使用 Selective Search。MultiBox 还可以通过用单类别预测替换置信度预测来执行单目标检测。然而，MultiBox 无法执行通用的目标检测，并且仍然只是一个较大的检测流程中的一部分，需要进一步的对图像块进行分类。YOLO 和 MultiBox 都使用卷积网络来预测图像中的边界框，但是 YOLO 是一个完整的检测系统。

OverFeat. Sermanet et al. train a convolutional neural network to perform localization and adapt that localizer to perform detection [32].

OverFeat efficiently performs sliding window detection but it is still a disjoint system. OverFeat optimizes for localization, not detection performance. Like DPM, the localizer only sees local information when making a prediction. OverFeat cannot reason about global context and thus requires significant post-processing to produce coherent detections.

OverFeat。Sermanet 等人训练了一个卷积神经网络来完成定位工作，并使该定位器进行检测[32]。OverFeat 高效地执行滑动窗口检测，但它仍然是一个不相交的系统。OverFeat 优化了定位，而不是检测性能。像 DPM 一样，定位器在进行预测时只能看到局部信息。OverFeat 不能推断全局上下文，因此需要大量的后处理来完成连贯的检测。

MultiGrasp. Our work is similar in design to work on grasp detection by Redmon et al [27]. Our grid approach to bounding box prediction is based on the MultiGrasp system for regression to grasps. However, grasp detection is a much simpler task than object detection. MultiGrasp only needs to predict a single graspable region for an image containing one object. It doesn’t have to estimate the size, location, or boundaries of the object or predict it’s class, only find a region suitable for grasping. YOLO predicts both bounding boxes and class probabilities for multiple objects of multiple classes in an image.

MultiGrasp。我们的工作在设计上类似于 Redmon 等[27]的 grasp 检测。我们对边界框预测的网格方法是基于 MultiGrasp 系统 grasp 的回归分析。然而，grasp 检测比目标检测任务要简单得多。MultiGrasp

只需要为包含一个目标的图像预测一个可以 grasp 的区域。不必估计目标的大小、位置或目标边界或预测目标的类别，只需找到适合 grasp 的区域。YOLO 可以预测图像中多个类别的多个目标的边界框和类别概率。

4. Experiments

First we compare YOLO with other real-time detection systems on PASCAL VOC 2007. To understand the differences between YOLO and R-CNN variants we explore the errors on VOC 2007 made by YOLO and Fast R-CNN, one of the highest performing versions of R-CNN [14]. Based on the different error profiles we show that YOLO can be used to rescore Fast R-CNN detections and reduce the errors from background false positives, giving a significant performance boost. We also present VOC 2012 results and compare mAP to current state-of-the-art methods. Finally, we show that YOLO generalizes to new domains better than other detectors on two artwork datasets.

4. 实验

首先，我们在 PASCAL VOC 2007 上比较了 YOLO 和其它的实时检测系统。为了理解 YOLO 和 R-CNN 变种之间的差异，我们探索了 YOLO 和 R-CNN 性能最高的版本之一 Fast R-CNN[14]在 VOC 2007 上错误率。根据不同的误差曲线，我们的研究显示 YOLO 可以用来重新评估 Fast R-CNN 检测，并减少背景假阳性带来的误差，从而显著提升性能。我们还展示了在 VOC 2012 上的结果，并与目前最

先进的方法比较了 mAP。最后，在两个艺术品数据集上我们显示了 YOLO 可以比其它检测器更好地泛化到新领域。

4.1. Comparison to Other Real-Time Systems

Many research efforts in object detection focus on making standard detection pipelines fast [5] [38] [31] [14] [17] [28]. However, only Sadeghi et al. actually produce a detection system that runs in real-time (30 frames per second or better) [31]. We compare YOLO to their GPU implementation of DPM which runs either at 30Hz or 100Hz. While the other efforts don't reach the real-time milestone we also compare their relative mAP and speed to examine the accuracy-performance tradeoffs available in object detection systems.

4.1. 与其它实时系统的比较

目标检测方面的许多研究工作都集中在对标准检测流程[5]，[38]，[31]，[14]，[17]，[28]提升速度上。然而，只有 Sadeghi 等真正研究出了一个实时运行的检测系统（每秒 30 帧或更好）[31]。我们将 YOLO 与他们 DPM 的 GPU 实现进行了比较，其在 30Hz 或 100Hz 下运行。虽然其它的研究工作没有达到实时检测的标准，我们也比较了它们的相对 mAP 和速度来检查目标检测系统中精度—性能之间的权衡。

Fast YOLO is the fastest object detection method on PASCAL; as far as we know, it is the fastest extant object detector. With 52.7% mAP, it is more than twice as accurate as prior work on real-time detection. YOLO pushes mAP to 63.4% while still maintaining real-time performance.

快速 YOLO 是 PASCAL 上最快的目标检测方法；据我们所知，它是现有的最快的目标检测器。具有 52.7%的 mAP，其精度是前人实时检测精度的两倍以上。YOLO 将 mAP 提升到 63.4%的同时保持了实时检测的性能。

We also train YOLO using VGG-16. This model is more accurate but also significantly slower than YOLO. It is useful for comparison to other detection systems that rely on VGG-16 but since it is slower than real-time the rest of the paper focuses on our faster models.

我们还使用 VGG-16 训练了 YOLO。（译者注：YOLO 使用了作者自己开发的 DarkNet 模型为 baseline）这个模型比 YOLO 更准确，但速度慢得多。这个模型可以用来与依赖于 VGG-16 的其它检测系统作比较，但由于它比实时的 YOLO 更慢，本文的剩余部分将重点放在我们更快的模型上。

Fastest DPM effectively speeds up DPM without sacrificing much mAP but it still misses real-time performance by a factor of 2 [38]. It also is limited by DPM’s relatively low accuracy on detection compared to neural network approaches.

Fastest DPM 可以在不牺牲太多 mAP 的情况下有效地加速 DPM，但仍然会将实时性能降低 2 倍[38]。与神经网络方法相比，DPM 相对较低的检测精度也是其限制。

R-CNN minus R replaces Selective Search with static bounding box proposals [20]. While it is much faster than R-CNN, it still falls short of real-time and takes a significant accuracy hit from not having good proposals.

减去 R 的 R-CNN 用静态边界框 proposals 取代 Selective Search [20]。虽然速度比 R-CNN 更快，但仍然达不到实时，并且由于没有好的边界框 proposals，准确性受到了严重影响。

Fast R-CNN speeds up the classification stage of R-CNN but it still relies on selective search which can take around 2 seconds per image to generate bounding box proposals. Thus it has high mAP but at 0.5 fps it is still far from real-time.

Fast R-CNN 加快了 R-CNN 的分类阶段，但是仍然依赖 selective search，每张图像需要花费大约 2 秒来生成边界框 proposals。因此，它具有很高的 mAP，但是 0.5 fps 的速度仍离实时性很远。

The recent Faster R-CNN replaces selective search with a neural network to propose bounding boxes, similar to Szegedy et al. [8]. In our tests, their most accurate model achieves 7 fps while a smaller, less accurate one runs at 18 fps. The VGG-16 version of Faster R-CNN is 10 mAP higher but is also 6 times slower than YOLO. The Zeiler-Fergus Faster R-CNN is only 2.5 times slower than YOLO but is also less accurate.

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.

最近 Faster R-CNN 用神经网络替代了 selective search 来提出边界框，类似于 Szegedy 等[8]。在我们的测试中，他们最精确的模型达到了 7fps，而较小的、不太精确的模型运行速度达到 18fps。VGG-16 版本的 Faster R-CNN 要高出 10mAP，但速度比 YOLO 慢 6 倍。Zeiler-Fergus 的 Faster R-CNN 只比 YOLO 慢了 2.5 倍，但也不太准确。

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

表 1：Pascal VOC 2007 上的实时检测系统。比较了快速检测器的性能和速度。Fast YOLO 是 Pascal VOC 检测记录中速度最快的检测器，其精度仍然是其它实时检测器的两倍。YOLO 比快速版本更精确 10mAP，同时在速度上仍保持了实时性。

4.2. VOC 2007 Error Analysis

To further examine the differences between YOLO and state-of-the-art detectors, we look at a detailed breakdown of results on VOC 2007. We compare YOLO to Fast R-CNN since Fast R-CNN is one of the highest performing detectors on PASCAL and it's detections are publicly available.

4.2. VOC 2007 检测误差分析

为了进一步检查 YOLO 和最先进的检测器之间的差异，我们详细分析了 VOC 2007 的结果。我们将 YOLO 与 Fast R-CNN 进行比较，因为 Fast R-CNN 是 PASCAL 上性能最高的检测器之一并且它的检测代码是可公开得到的。

We use the methodology and tools of Hoiem et al. [19] For each category at test time we look at the top N predictions for that category. Each prediction is either correct or it is classified based on the type of error:

- Correct: correct class and IOU>.5
- Localization: correct class, .1<IOU<.5
- Similar: class is similar, IOU>.1
- Other: class is wrong, IOU>.1
- Background: IOU<.1 for any object

Figure 4 shows the breakdown of each error type averaged across all 20 classes.

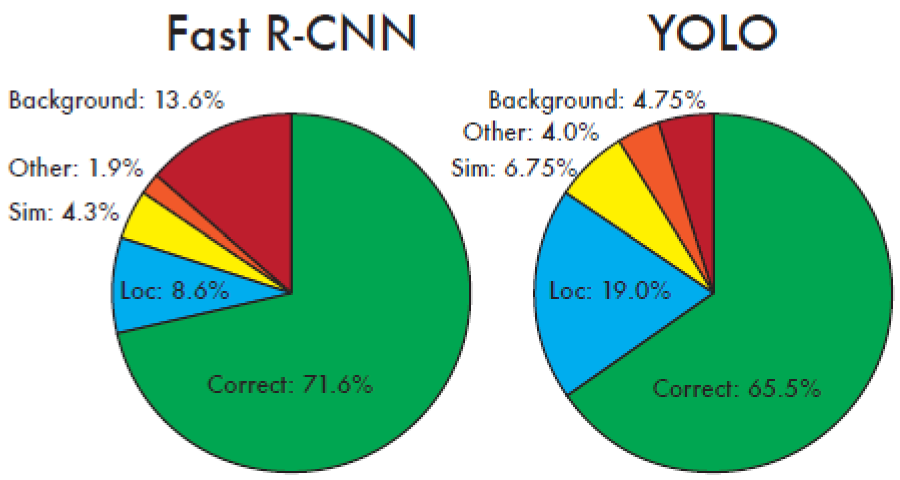


Figure 4: Error Analysis: Fast R-CNN vs. YOLO. These charts show the percentage of localization and background errors in the top N detections for various categories (N = # objects in that category).

我们使用 Hoiem 等人[19]的方法和工具。对于测试时的每个类别，我们只关注这个类别的前 N 个预测。每个预测要么归为正确，要么根据错误类型进行归类：

- Correct：分类正确且 IOU>0.5。
- Localization：分类正确但 0.1<IOU<0.5。
- Similar：分类的类别相似且 IOU>0.1。
- Other：类别错误，IOU>0.1。
- Background：分类为其它任何目标，IOU<0.1。

图 4 显示了在所有的 20 个类别上每种错误类型平均值的分解图。

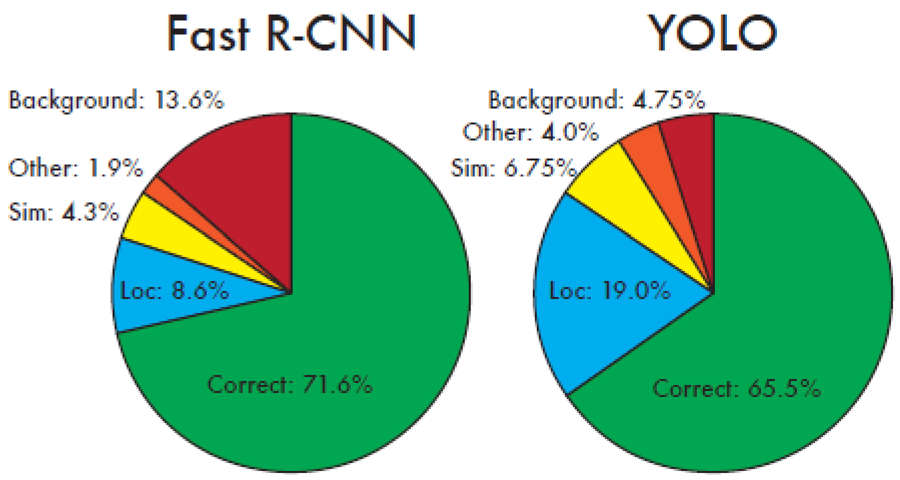


图 4，误差分析：Fast R-CNN 对比 YOLO。这些图显示了各种类别的前 N 个预测中定位错误和背景错误的百分比（N = #表示目标在那个类别中）。

YOLO struggles to localize objects correctly. Localization errors account for more of YOLO’s errors than all other sources combined. Fast R-CNN makes much fewer localization errors but far more background errors. 13.6% of it’s top detections are false positives that don’t contain any objects. Fast R-CNN is almost 3x more likely to predict background detections than YOLO.

YOLO 需要改善正确定位目标。定位误差占 YOLO 模型误差的大多数，比其它误差错误来源总合都多。Fast R-CNN 定位误差少很多，但背景误差更多。它的检测结果中 13.6%是不包含任何目标的假阳性。Fast R-CNN 与 YOLO 相比，将背景预测成目标的可能性高出近 3 倍。（译者注：根据图 4，13.6/4.75=2.86）

4.3. Combining Fast R-CNN and YOLO

YOLO makes far fewer background mistakes than Fast R-CNN. By using YOLO to eliminate background detections from Fast R-CNN we get a significant boost in performance. For every bounding box that R-CNN predicts we check to see if YOLO predicts a similar box. If it does, we give that prediction a boost based on the probability predicted by YOLO and the overlap between the two boxes.

4.3. 结合 Fast R-CNN 和 YOLO

YOLO 比 Fast R-CNN 的背景误检要少得多。通过使用 YOLO 消除 Fast R-CNN 的背景检测，我们获得了显著的性能提升。对于 R-CNN 预测的每个边界框，我们检查 YOLO 是否预测一个类似的框。如果是这样，我们根据 YOLO 预测的概率和两个盒子之间的重叠来对这个预测进行改进。

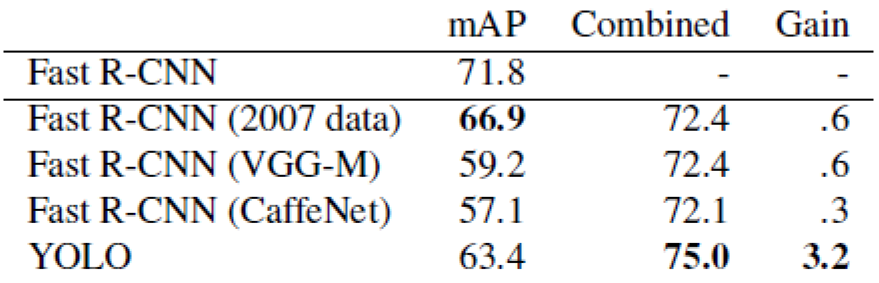
The best Fast R-CNN model achieves a mAP of 71.8% on the VOC 2007 test set. When combined with YOLO, its mAP increases by 3.2% to 75.0%. We also tried combining the top Fast R-CNN model with several other versions of Fast R-CNN. Those ensembles produced small increases in mAP between .3 and .6%, see Table 2 for details.

	mAP	Combined	Gain
Fast R-CNN	71.8	-	-
Fast R-CNN (2007 data)	**66.9**	72.4	.6
Fast R-CNN (VGG-M)	59.2	72.4	.6
Fast R-CNN (CaffeNet)	57.1	72.1	.3
YOLO	63.4	**75.0**	**3.2**

Table 2: Model combination experiments on VOC 2007. We examine the effect of combining various models with the best version of Fast R-CNN. Other versions of Fast R-CNN provide only a small benefit while YOLO provides a significant performance boost.

最好的 Fast R-CNN 模型在 VOC 2007 测试集上达到了 71.8%的 mAP。当与 YOLO 结合时，其 mAP 增加了 3.2%达到了 75.0%。我们

也尝试将最好的 Fast R-CNN 模型与其它几个版本的 Fast R-CNN 结合起来。这些模型组合产生了 0.3-0.6%的小幅增加，详见表 2。



	mAP	Combined	Gain
Fast R-CNN	71.8	-	-
Fast R-CNN (2007 data)	**66.9**	72.4	.6
Fast R-CNN (VGG-M)	59.2	72.4	.6
Fast R-CNN (CaffeNet)	57.1	72.1	.3
YOLO	63.4	**75.0**	**3.2**

表 2：VOC 2007 模型组合实验。我们检验了各种模型与 Fast R-CNN 最佳版本结合的效果。Fast R-CNN 的其它版本只产生了很小的改进，而 YOLO 则提供了显著的性能提升。

The boost from YOLO is not simply a byproduct of model ensembling since there is little benefit from combining different versions of Fast R-CNN. Rather, it is precisely because YOLO makes different kinds of mistakes at test time that it is so effective at boosting Fast R-CNN’s performance.

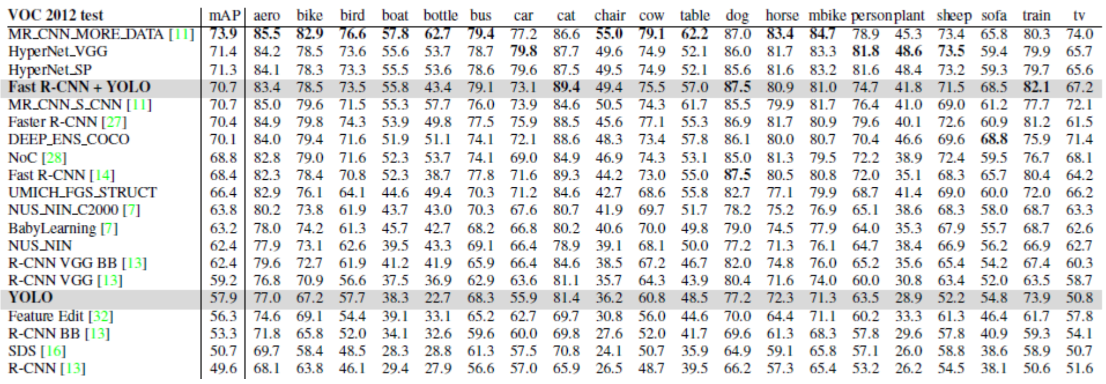
来自 YOLO 的提升不仅仅是模型集成的副产品，因为组合不同版本的 Fast R-CNN 几乎没有什么改进。相反，正是因为 YOLO 在测试时出现了各种各样的误差，所以在提高 Fast R-CNN 的性能方面非常有效。

Unfortunately, this combination doesn’t benefit from the speed of YOLO since we run each model seperately and then combine the results. However, since YOLO is so fast it doesn’t add any significant computational time compared to Fast R-CNN.

遗憾的是，这个组合并没有从 YOLO 的速度中受益，因为我们分别运行每个模型，然后将结果组合起来。但是，由于 YOLO 速度如此之快，与 Fast R-CNN 相比，不会增加任何显著的计算时间。

4.4. VOC 2012 Results

On the VOC 2012 test set, YOLO scores 57.9% mAP. This is lower than the current state of the art, closer to the original R-CNN using VGG-16, see Table 3. Our system struggles with small objects compared to its closest competitors. On categories like bottle, sheep, and tv/monitor YOLO scores 8−10% lower than R-CNN or Feature Edit. However, on other categories like cat and train YOLO achieves higher performance.



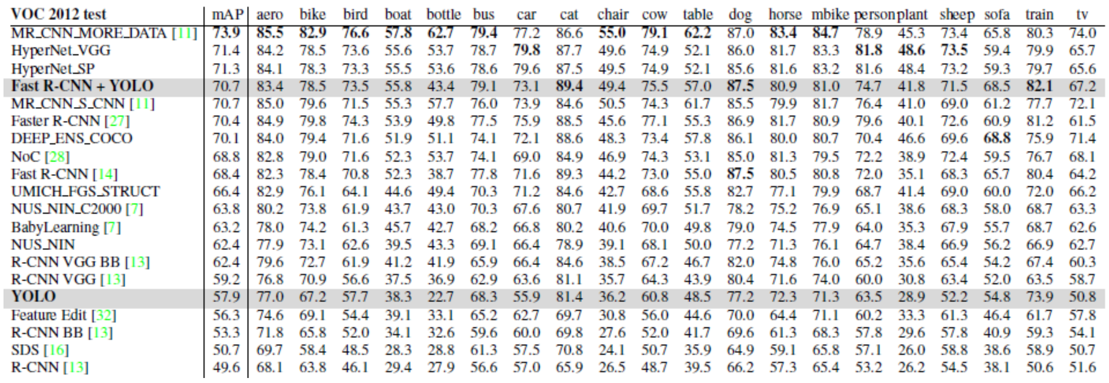
VOC 2012 test	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
MR_CNN_MORE_DATA [11]	**73.9**	**85.5**	**82.9**	**76.6**	**57.8**	**62.7**	**79.4**	77.2	86.6	**55.0**	**79.1**	**62.2**	87.0	**83.4**	**84.7**	78.9	45.3	73.4	65.8	80.3	74.0
HyperNet_VGG	71.4	84.2	78.5	73.6	55.6	53.7	78.7	**79.8**	87.7	49.6	74.9	52.1	86.0	81.7	83.3	**81.8**	**48.6**	**73.5**	59.4	79.9	65.7
HyperNet_SP	71.3	84.1	78.3	73.3	55.5	53.6	78.6	79.6	87.5	49.5	74.9	52.1	85.6	81.6	83.2	81.6	48.4	73.2	59.3	79.7	65.6
Fast R-CNN + YOLO	70.7	83.4	78.5	73.5	55.8	43.4	79.1	73.1	**89.4**	49.4	75.5	57.0	**87.5**	80.9	81.0	74.7	41.8	71.5	68.5	**82.1**	67.2
MR_CNN_S_CNN [11]	70.7	85.0	79.6	71.5	55.3	57.7	76.0	73.9	84.6	50.5	74.3	61.7	85.5	79.9	81.7	76.4	41.0	69.0	61.2	77.7	72.1
Faster R-CNN [27]	70.4	84.9	79.8	74.3	53.9	49.8	77.5	75.9	88.5	45.6	77.1	55.3	86.9	81.7	80.9	79.6	40.1	72.6	60.9	81.2	61.5
DEEP_ENS_COCO	70.1	84.0	79.4	71.6	51.9	51.1	74.1	72.1	88.6	48.3	73.4	57.8	86.1	80.0	80.7	70.4	46.6	69.6	**68.8**	75.9	71.4
NoC [28]	68.8	82.8	79.0	71.6	52.3	53.7	74.1	69.0	84.9	46.9	74.3	53.1	85.0	81.3	79.5	72.2	38.9	72.4	59.5	76.7	68.1
Fast R-CNN [14]	68.4	82.3	78.4	70.8	52.3	38.7	77.8	71.6	89.3	44.2	73.0	55.0	**87.5**	80.5	80.8	72.0	35.1	68.3	65.7	80.4	64.2
UMICH_FGS_STRUCT	66.4	82.9	76.1	64.1	44.6	49.4	70.3	71.2	84.6	42.7	68.6	55.8	82.7	77.1	79.9	68.7	41.4	69.0	60.0	72.0	66.2
NUS_NIN_C2000 [7]	63.8	80.2	73.8	61.9	43.7	43.0	70.3	67.6	80.7	41.9	69.7	51.7	78.2	75.2	76.9	65.1	38.6	68.3	58.0	68.7	63.3
BabyLearning [7]	63.2	78.0	74.2	61.3	45.7	42.7	68.2	66.8	80.2	40.6	70.0	49.8	79.0	74.5	77.9	64.0	35.3	67.9	55.7	68.7	62.6
NUS_NIN	62.4	77.9	73.1	62.6	39.5	43.3	69.1	66.4	78.9	39.1	68.1	50.0	77.2	71.3	76.1	64.7	38.4	66.9	56.2	66.9	62.7
R-CNN VGG BB [13]	62.4	79.6	72.7	61.9	41.2	41.9	65.9	66.4	84.6	38.5	67.2	46.7	82.0	74.8	76.0	65.2	35.6	65.4	54.2	67.4	60.3
R-CNN VGG [13]	59.2	76.8	70.9	56.6	37.5	36.9	62.9	63.6	81.1	35.7	64.3	43.9	80.4	71.6	74.0	60.0	30.8	63.4	52.0	63.5	58.7
YOLO	57.9	77.0	67.2	57.7	38.3	22.7	68.3	55.9	81.4	36.2	60.8	48.5	77.2	72.3	71.3	63.5	28.9	52.2	54.8	73.9	50.8
Feature Edit [32]	56.3	74.6	69.1	54.4	39.1	33.1	65.2	62.7	69.7	30.8	56.0	44.6	70.0	64.4	71.1	60.2	33.3	61.3	46.4	61.7	57.8
R-CNN BB [13]	53.3	71.8	65.8	52.0	34.1	32.6	59.6	60.0	69.8	27.6	52.0	41.7	69.6	61.3	68.3	57.8	29.6	57.8	40.9	59.3	54.1
SDS [16]	50.7	69.7	58.4	48.5	28.3	28.8	61.3	57.5	70.8	24.1	50.7	35.9	64.9	59.1	65.8	57.1	26.0	58.8	38.6	58.9	50.7
R-CNN [13]	49.6	68.1	63.8	46.1	29.4	27.9	56.6	57.0	65.9	26.5	48.7	39.5	66.2	57.3	65.4	53.2	26.2	54.5	38.1	50.6	51.6

Table 3: PASCAL VOC 2012 Leaderboard. YOLO compared with the full comp4 (outside data allowed) public leaderboard as of November 6th, 2015. Mean average precision and per-class average precision are shown for a variety of detection methods. YOLO is the only real-time detector. Fast R-CNN + YOLO is the forth highest scoring method, with a 2.3% boost over Fast R-CNN.

4.4. VOC 2012 上的结果

在 VOC 2012 测试集上，YOLO 获得了 57.9%的 mAP。这低于现有的最好技术，如表 3 所示其接近于使用 VGG-16 的原始 R-CNN。我们的系统与其最接近的竞争对手相比，需要改善在小目标上的检测。在水瓶、绵羊和电视/显示器等类别上，YOLO 的得分比 R-CNN 或

Feature Edit 低 8−10%。然而，在猫和火车等其它类别上 YOLO 实现了更高的性能。



VOC 2012 test	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
MR_CNN_MORE_DATA [11]	**73.9**	**85.5**	**82.9**	**76.6**	**57.8**	**62.7**	**79.4**	77.2	86.6	**55.0**	**79.1**	**62.2**	87.0	**83.4**	**84.7**	78.9	45.3	73.4	65.8	80.3	74.0
HyperNet_VGG	71.4	84.2	78.5	73.6	55.6	53.7	78.7	**79.8**	87.7	49.6	74.9	52.1	86.0	81.7	83.3	**81.8**	**48.6**	**73.5**	59.4	79.9	65.7
HyperNet_SP	71.3	84.1	78.3	73.3	55.5	53.6	78.6	79.6	87.5	49.5	74.9	52.1	85.6	81.6	83.2	81.6	48.4	73.2	59.3	79.7	65.6
Fast R-CNN + YOLO	70.7	83.4	78.5	73.5	55.8	43.4	79.1	73.1	**89.4**	49.4	75.5	57.0	**87.5**	80.9	81.0	74.7	41.8	71.5	68.5	**82.1**	67.2
MR_CNN_S_CNN [11]	70.7	85.0	79.6	71.5	55.3	57.7	76.0	73.9	84.6	50.5	74.3	61.7	85.5	79.9	81.7	76.4	41.0	69.0	61.2	77.7	72.1
Faster R-CNN [27]	70.4	84.9	79.8	74.3	53.9	49.8	77.5	75.9	88.5	45.6	77.1	55.3	86.9	81.7	80.9	79.6	40.1	72.6	60.9	81.2	61.5
DEEP_ENS_COCO	70.1	84.0	79.4	71.6	51.9	51.1	74.1	72.1	88.6	48.3	73.4	57.8	86.1	80.0	80.7	70.4	46.6	69.6	**68.8**	75.9	71.4
NoC [28]	68.8	82.8	79.0	71.6	52.3	53.7	74.1	69.0	84.9	46.9	74.3	53.1	85.0	81.3	79.5	72.2	38.9	72.4	59.5	76.7	68.1
Fast R-CNN [14]	68.4	82.3	78.4	70.8	52.3	38.7	77.8	71.6	89.3	44.2	73.0	55.0	**87.5**	80.5	80.8	72.0	35.1	68.3	65.7	80.4	64.2
UMICH_FGS_STRUCT	66.4	82.9	76.1	64.1	44.6	49.4	70.3	71.2	84.6	42.7	68.6	55.8	82.7	77.1	79.9	68.7	41.4	69.0	60.0	72.0	66.2
NUS_NIN_C2000 [7]	63.8	80.2	73.8	61.9	43.7	43.0	70.3	67.6	80.7	41.9	69.7	51.7	78.2	75.2	76.9	65.1	38.6	68.3	58.0	68.7	63.3
BabyLearning [7]	63.2	78.0	74.2	61.3	45.7	42.7	68.2	66.8	80.2	40.6	70.0	49.8	79.0	74.5	77.9	64.0	35.3	67.9	55.7	68.7	62.6
NUS_NIN	62.4	77.9	73.1	62.6	39.5	43.3	69.1	66.4	78.9	39.1	68.1	50.0	77.2	71.3	76.1	64.7	38.4	66.9	56.2	66.9	62.7
R-CNN VGG BB [13]	62.4	79.6	72.7	61.9	41.2	41.9	65.9	66.4	84.6	38.5	67.2	46.7	82.0	74.8	76.0	65.2	35.6	65.4	54.2	67.4	60.3
R-CNN VGG [13]	59.2	76.8	70.9	56.6	37.5	36.9	62.9	63.6	81.1	35.7	64.3	43.9	80.4	71.6	74.0	60.0	30.8	63.4	52.0	63.5	58.7
YOLO	57.9	77.0	67.2	57.7	38.3	22.7	68.3	55.9	81.4	36.2	60.8	48.5	77.2	72.3	71.3	63.5	28.9	52.2	54.8	73.9	50.8
Feature Edit [32]	56.3	74.6	69.1	54.4	39.1	33.1	65.2	62.7	69.7	30.8	56.0	44.6	70.0	64.4	71.1	60.2	33.3	61.3	46.4	61.7	57.8
R-CNN BB [13]	53.3	71.8	65.8	52.0	34.1	32.6	59.6	60.0	69.8	27.6	52.0	41.7	69.6	61.3	68.3	57.8	29.6	57.8	40.9	59.3	54.1
SDS [16]	50.7	69.7	58.4	48.5	28.3	28.8	61.3	57.5	70.8	24.1	50.7	35.9	64.9	59.1	65.8	57.1	26.0	58.8	38.6	58.9	50.7
R-CNN [13]	49.6	68.1	63.8	46.1	29.4	27.9	56.6	57.0	65.9	26.5	48.7	39.5	66.2	57.3	65.4	53.2	26.2	54.5	38.1	50.6	51.6

表 3: PASCAL VOC 2012 排行榜。截至 2015 年 11 月 6 日，YOLO 与完整 comp4（允许外部数据）公开排行榜进行了比较。显示了各种检测方法的 mAP 和每类的平均精度。YOLO 是唯一达到实时的检测器。Fast R-CNN + YOLO 是评分第四高的方法，比 Fast R-CNN 提升了 2.3%。

Our combined Fast R-CNN + YOLO model is one of the highest performing detection methods. Fast R-CNN gets a 2.3% improvement from the combination with YOLO, boosting it 5 spots up on the public leaderboard.

我们联合的 Fast R-CNN + YOLO 模型是性能最高的检测方法之一。Fast R-CNN 从与 YOLO 的组合中获得了 2.3%的提高，在公开排行榜上提升了 5 位。

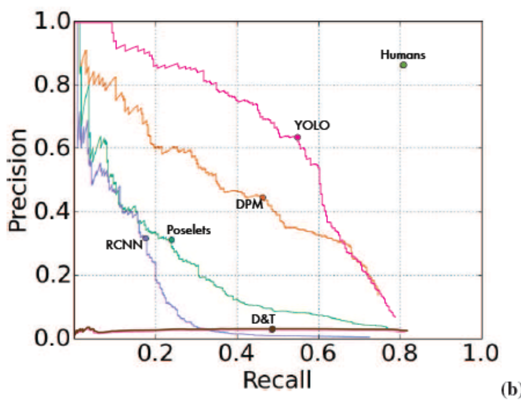
4.5. Generalizability: Person Detection in Artwork

Academic datasets for object detection draw the training and testing data from the same distribution. In real-world applications it is hard to predict all possible use cases and the test data can diverge from what the system has seen before [3]. We compare YOLO to other detection systems on the Picasso Dataset [12] and the People-Art Dataset [3], two datasets for testing person detection on artwork.

4.5. 泛化能力：艺术品中的行人检测

用于目标检测的学术数据集以相同分布获取训练和测试数据。在现实世界的应用中，很难预测所有可能的样本，而且测试数据可能与系统之前看到的不同[3]。（译者注：也就是说测试数据与模型训练的数据可能在风格、模式、目标表现形式等方面有很大的区别，例如模型训练时的数据是用相机拍摄的图像，而测试时用油画作品图像，此时由于油画作品比较抽象，模型就可能无法很好地识别其中的目标）我们在 Picasso 数据集上[12]和 People-Art 数据集[3]上将 YOLO 与其它的检测系统进行比较，这两个数据集常用于测试艺术品中的行人检测。

Figure 5 shows comparative performance between YOLO and other detection methods. For reference, we give VOC 2007 detection AP on person where all models are trained only on VOC 2007 data. On Picasso models are trained on VOC 2012 while on People-Art they are trained on VOC 2010.



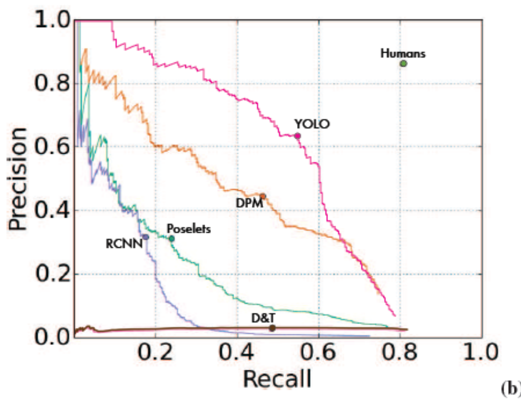
(a) Picasso Dataset precision-recall curves.

	VOC 2007	Picasso		People-Art
	AP	AP	Best F_1	AP
YOLO	**59.2**	**53.3**	**0.590**	**45**
R-CNN	54.2	10.4	0.226	26
DPM	43.2	37.8	0.458	32
Poselets [2]	36.5	17.8	0.271	
D&T [4]	-	1.9	0.051	

(b) Quantitative results on the VOC 2007, Picasso, and People-Art Datasets. The Picasso Dataset evaluates on both AP and best F_1 score.

Figure 5: Generalization results on Picasso and People-Art datasets.

图 5 所示为 YOLO 和其它检测方法之间性能比较的结果。作为参考，我们在 person 上提供 VOC 2007 的检测 AP，其中所有模型仅在 VOC 2007 数据上训练。在 Picasso 数据集上测试的模型使用 VOC 2012 进行了训练，而 People-Art 数据集测试的模型使用 VOC 2010 进行了训练。



(a) Picasso Dataset precision-recall curves.

	VOC 2007	Picasso		People-Art
	AP	AP	Best F_1	AP
YOLO	**59.2**	**53.3**	**0.590**	**45**
R-CNN	54.2	10.4	0.226	26
DPM	43.2	37.8	0.458	32
Poselets [2]	36.5	17.8	0.271	
D&T [4]	-	1.9	0.051	

(b) Quantitative results on the VOC 2007, Picasso, and People-Art Datasets. The Picasso Dataset evaluates on both AP and best F_1 score.

图 5：Picasso 和 People-Art 数据集上的泛化结果。

R-CNN has high AP on VOC 2007. However, R-CNN drops off considerably when applied to artwork. R-CNN uses Selective Search for bounding box proposals which is tuned for natural images. The classifier step in R-CNN only sees small regions and needs good proposals.

R-CNN 在 VOC 2007 上的 AP 较高。然而，当应用于艺术品时，R-CNN 明显性能下降。R-CNN 使用 Selective Search 来调整自然图像的边界框 proposals。R-CNN 中的分类器步骤只能看到小区域，并且需要很好的边界框 proposals。

DPM maintains its AP well when applied to artwork. Prior work theorizes that DPM performs well because it has strong spatial models of the shape and layout of objects. Though DPM doesn’t degrade as much as R-CNN, it starts from a lower AP.

DPM 在应用于艺术品时保持了其 AP。之前的工作认为 DPM 表现良好，因为它具有目标形状和布局的强大空间模型。虽然 DPM 不会像 R-CNN 那样 AP 下降很厉害，但它的 AP 一开始就较低（译者注：DPM AP 本来就低，已经没有下降空间了，哈哈哈）。

YOLO has good performance on VOC 2007 and its AP degrades less than other methods when applied to artwork. Like DPM, YOLO models the size and shape of objects, as well as relationships between objects and where objects commonly appear. Artwork and natural images are very different on a pixel level but they are similar in terms of the size and shape of objects, thus YOLO can still predict good bounding boxes and detections.

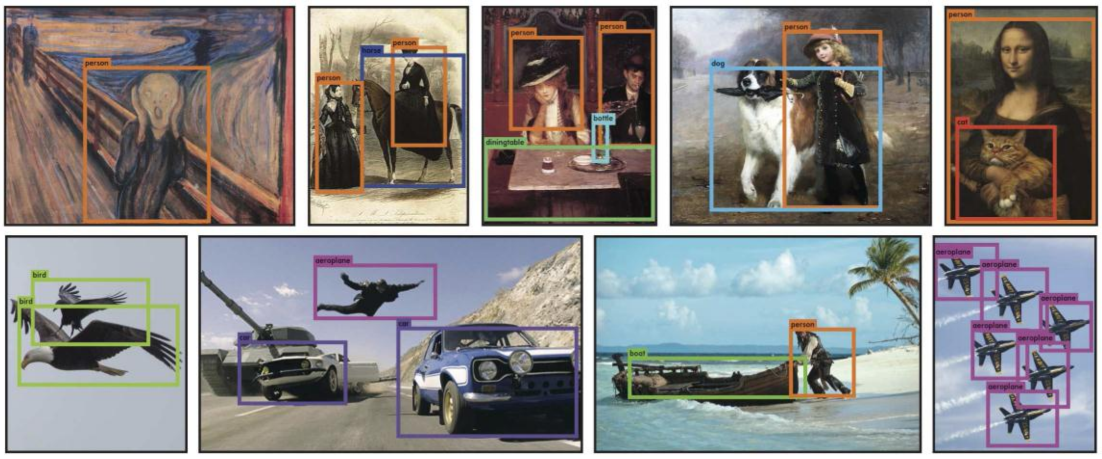


Figure 6: Qualitative Results. YOLO running on sample artwork and natural images from the internet. It is mostly accurate although it does think one person is an airplane.

YOLO 在 VOC 2007 上有很好的性能，在应用于艺术品时其 AP 下降幅度低于其它方法。像 DPM 一样，YOLO 建模了目标的大小和

形状以及目标和目标通常出现的位置之间的关系。艺术品和自然图像在像素级别上有很大不同，但是它们在目标的大小和形状方面是相似的，因此 YOLO 仍然可以预测好的边界框和检测结果。

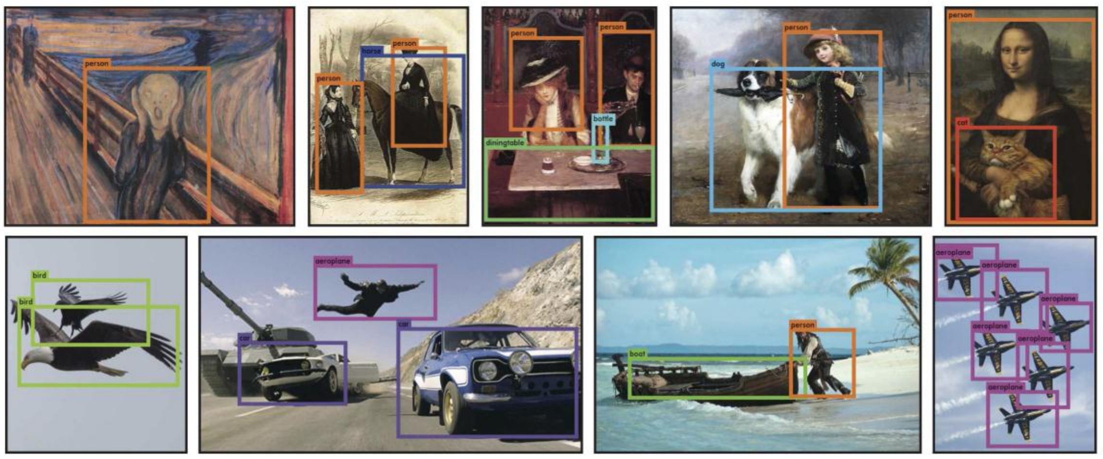


图 6：定性结果。YOLO 在源于网络的艺术品和自然图像上的运行结果。虽然它将人误检成了飞机，但它大部分上是准确的。

5. Real-Time Detection In The Wild

YOLO is a fast, accurate object detector, making it ideal for computer vision applications. We connect YOLO to a webcam and verify that it maintains real-time performance, including the time to fetch images from the camera and display the detections.

5. 现实环境下的实时检测

YOLO 是一种快速、精确的目标检测器，非常适合计算机视觉应用。我们将 YOLO 连接到网络摄像头，并验证它是否能保持实时性能，包括从摄像头获取图像并显示检测结果的时间。

The resulting system is interactive and engaging. While YOLO processes images individually, when attached to a webcam it functions like a tracking system, detecting objects as they move around and change in appearance. A demo of the system and the source code can be found on our project website: http://pjreddie.com/yolo/.

最终的系统是交互式的并且是参与式的。虽然 YOLO 单独地处理图像，但当连接到网络摄像头时，其功能类似于跟踪系统，可在目标移动和外观变化时检测目标。系统演示和源代码可以在我们的项目网站上找到：http://pjreddie.com/yolo/。

6. Conclusion

We introduce YOLO, a unified model for object detection. Our model is simple to construct and can be trained directly on full images. Unlike classifier-based approaches, YOLO is trained on a loss function that directly corresponds to detection performance and the entire model is trained jointly.

6. 结论

我们介绍了 YOLO，一种统一的目标检测模型。我们的模型构建简单，可以直接在整张图像上进行训练。与基于分类器的方法不同，YOLO 直接在对应检测性能的损失函数上训练，并且整个模型统一训练。

Fast YOLO is the fastest general-purpose object detector in the literature and YOLO pushes the state-of-the-art in real-time object detection. YOLO also generalizes well to new domains making it ideal for applications that rely on fast, robust object detection.

快速 YOLO 是文献中最快的通用目的的目标检测器，YOLO 推动了实时目标检测的最新技术。YOLO 还很好地泛化到新领域，使其成为要求快速、强大目标检测应用的理想选择。

Acknowledgements: This work is partially supported by ONR N00014-13-1-0720, NSF IIS-1338054, and The Allen Distinguished Investigator Award.

致谢：这项工作得到了 ONR N00014-13-1-0720、NSF IIS-1338054 和 The Allen Distinguished Investigator Award 的部分支持。

References

参考文献

[1] M. B. Blaschko and C. H. Lampert. Learning to localize objects with structured output regression. In Computer Vision–ECCV 2008, pages 2–15. Springer, 2008. 4
[2] L. Bourdev and J. Malik. Poselets: Body part detectors trained using 3d human pose annotations. In International Conference on Computer Vision (ICCV), 2009. 8
[3] H. Cai, Q. Wu, T. Corradi, and P. Hall. The cross-depiction problem: Computer vision algorithms for recognising objects in artwork and in photographs. arXiv preprint arXiv:1505.00110, 2015. 7
[4] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, volume 1, pages 886–893. IEEE, 2005. 4, 8
[5] T. Dean, M. Ruzon, M. Segal, J. Shlens, S. Vijaya-narasimhan, J. Yagnik, et al. Fast, accurate detection of 100,000 object classes on a single machine. In Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on, pages 1814–1821. IEEE, 2013. 5
[6] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. arXiv preprint arXiv:1310.1531, 2013. 4
[7] J. Dong, Q. Chen, S. Yan, and A. Yuille. Towards unified object detection and semantic segmentation. In Computer Vision–ECCV 2014, pages 299–314. Springer, 2014. 7
[8] D.Erhan, C.Szegedy, A.Toshev, and D.Anguelov. Scalable object detection using deep neural networks. In Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on, pages 2155–2162. IEEE, 2014. 5, 6

[9] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. International Journal of Computer Vision, 111(1):98–136, Jan. 2015. 2
[10] P.F.Felzenszwalb, R.B.Girshick, D.McAllester, and D.Ramanan. Object detection with discriminatively trained part based models. IEEE Transactions on Pattern Analysis and Machine Intelligence, 32(9):1627–1645, 2010. 1, 4
[11] S. Gidaris and N. Komodakis. Object detection via a multi-region & semantic segmentation-aware CNN model. CoRR, abs/1505.01749, 2015. 7
[12] S. Ginosar, D. Haas, T. Brown, and J. Malik. Detecting people in cubist art. In Computer Vision-ECCV 2014 Workshops, pages 101–116. Springer, 2014. 7
[13] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on, pages 580–587. IEEE, 2014. 1, 4, 7
[14] R. B. Girshick. Fast R-CNN. CoRR, abs/1504.08083, 2015. 2, 5, 6, 7
[15] S. Gould, T. Gao, and D. Koller. Region-based segmentation and object detection. In Advances in neural information processing systems, pages 655–663, 2009. 4
[16] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Simultaneous detection and segmentation. In Computer Vision–ECCV 2014, pages 297–312. Springer, 2014. 7
[17] K.He, X.Zhang, S.Ren, and J.Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. arXiv preprint arXiv:1406.4729, 2014. 5
[18] G.E.Hinton, N.Srivastava, A.Krizhevsky, I.Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580, 2012. 4
[19] D.Hoiem, Y.Chodpathumwan, and Q.Dai. Diagnosing error in object detectors. In Computer Vision–ECCV 2012, pages 340–353. Springer, 2012. 6
[20] K. Lenc and A. Vedaldi. R-cnn minus r. arXiv preprint arXiv:1506.06981, 2015. 5, 6
[21] R. Lienhart and J. Maydt. An extended set of haar-like features for rapid object detection. In Image Processing. 2002. Proceedings. 2002
International Conference on, volume 1, pages I–900. IEEE, 2002. 4
[22] M. Lin, Q. Chen, and S. Yan. Network in network. CoRR, abs/1312.4400, 2013. 2
[23] D. G. Lowe. Object recognition from local scale-invariant features. In Computer vision, 1999. The proceedings of the seventh IEEE international conference on, volume 2, pages 1150–1157. Ieee, 1999. 4
[24] D. Mishkin. Models accuracy on imagenet 2012 val. https://github.com/BVLC/caffe/wiki/ Models-accuracy-on-ImageNet-2012-val. Accessed: 2015-10-2. 3
[25] C. P. Papageorgiou, M. Oren, and T. Poggio. A general framework for object detection. In Computer vision, 1998. sixth international conference on, pages 555–562. IEEE, 1998. 4
[26] J. Redmon. Darknet: Open source neural networks in c. http://pjreddie.com/darknet/, 2013–2016. 3
[27] J.Redmon and A.Angelova. Real-time grasp detection using convolutional neural networks. CoRR, abs/1412.3128, 2014. 5

[28] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. arXiv preprint arXiv:1506.01497, 2015. 5, 6, 7

[29] S. Ren, K. He, R. B. Girshick, X. Zhang, and J. Sun. Object detection networks on convolutional feature maps. CoRR, abs/1504.06066, 2015. 3, 7

[30] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV), 2015. 3

[31] M. A. Sadeghi and D. Forsyth. 30hz object detection with dpm v5. In Computer Vision–ECCV 2014, pages 65–79. Springer, 2014. 5, 6

[32] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. CoRR, abs/1312.6229, 2013. 4, 5

[33] Z.Shen and X.Xue. Do more dropouts in pool5 feature maps for better object detection. arXiv preprint arXiv:1409.6911, 2014. 7

[34] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. CoRR, abs/1409.4842, 2014. 2

[35] J. R. Uijlings, K. E. van de Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. International journal of computer vision, 104(2):154–171, 2013. 4, 5

[36] P. Viola and M. Jones. Robust real-time object detection. International Journal of Computer Vision, 4:34–47, 2001. 4

[37] P. Viola and M. J. Jones. Robust real-time face detection. International journal of computer vision, 57(2):137–154, 2004. 5

[38] J. Yan, Z. Lei, L. Wen, and S. Z. Li. The fastest deformable part model for object detection. In Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on, pages 2497–2504. IEEE, 2014. 5, 6

[39] C.L.Zitnick and P.Dollár.Edgeboxes:Locating object proposals from edges. In Computer Vision–ECCV 2014, pages 391–405. Springer, 2014. 4