五、SSD原理（Single Shot MultiBox Detector）

主流的算法主要分为两个类型：

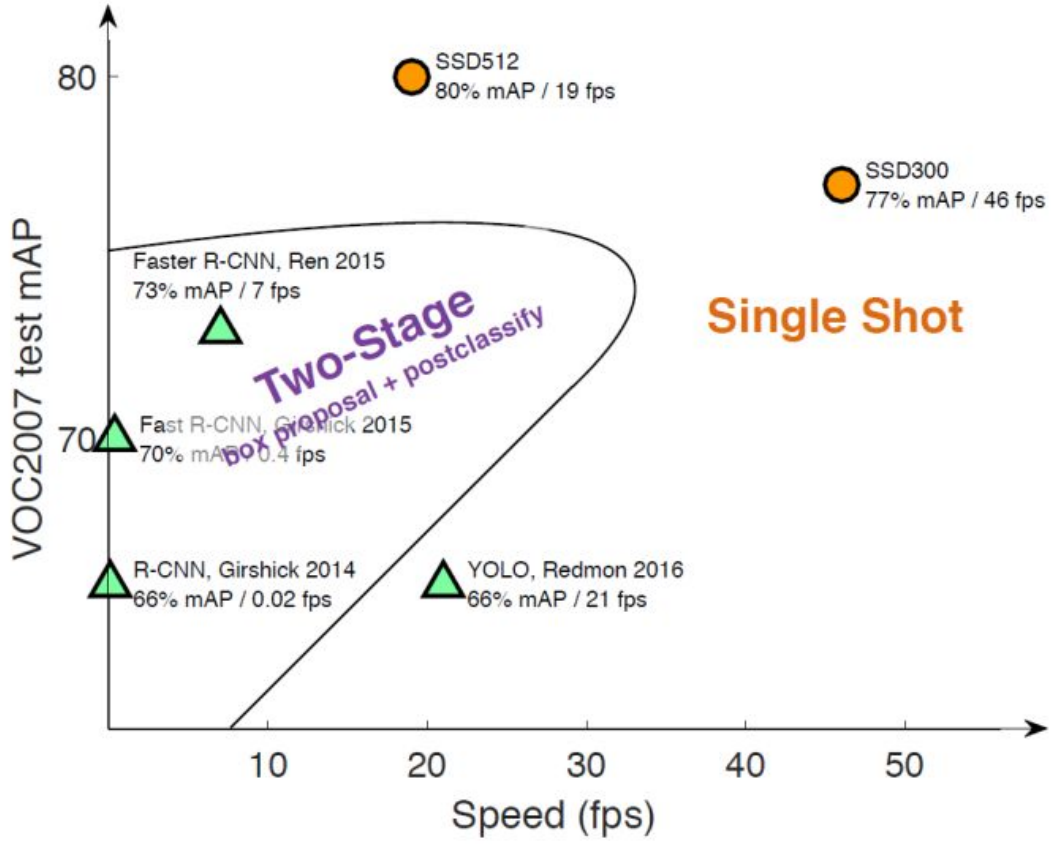
（1）tow-stage

R-CNN系列算法，其主要思路是先通过启发式方法（**selective search**）或者CNN网络（**RPN**）产生一些列稀疏的候选框，然后对这些候选框进行分类和回归。two-stage方法的优势是准确度高。

（2）one-stage

如YOLO和SSD，主要思路是均匀的在图片的不同位置进行**密集抽样**，抽样时可以采用不同尺度和长宽比，然后利用CNN提取特征后直接进行分类和回归，整个过程只需要一部，所以其优势是速度快。

均匀的密集采样的一个重要缺点是**训练比较困难**，这主要是因为正样本与负样本极其不平衡，导致模型准确度稍低，不同算法的性能如图：



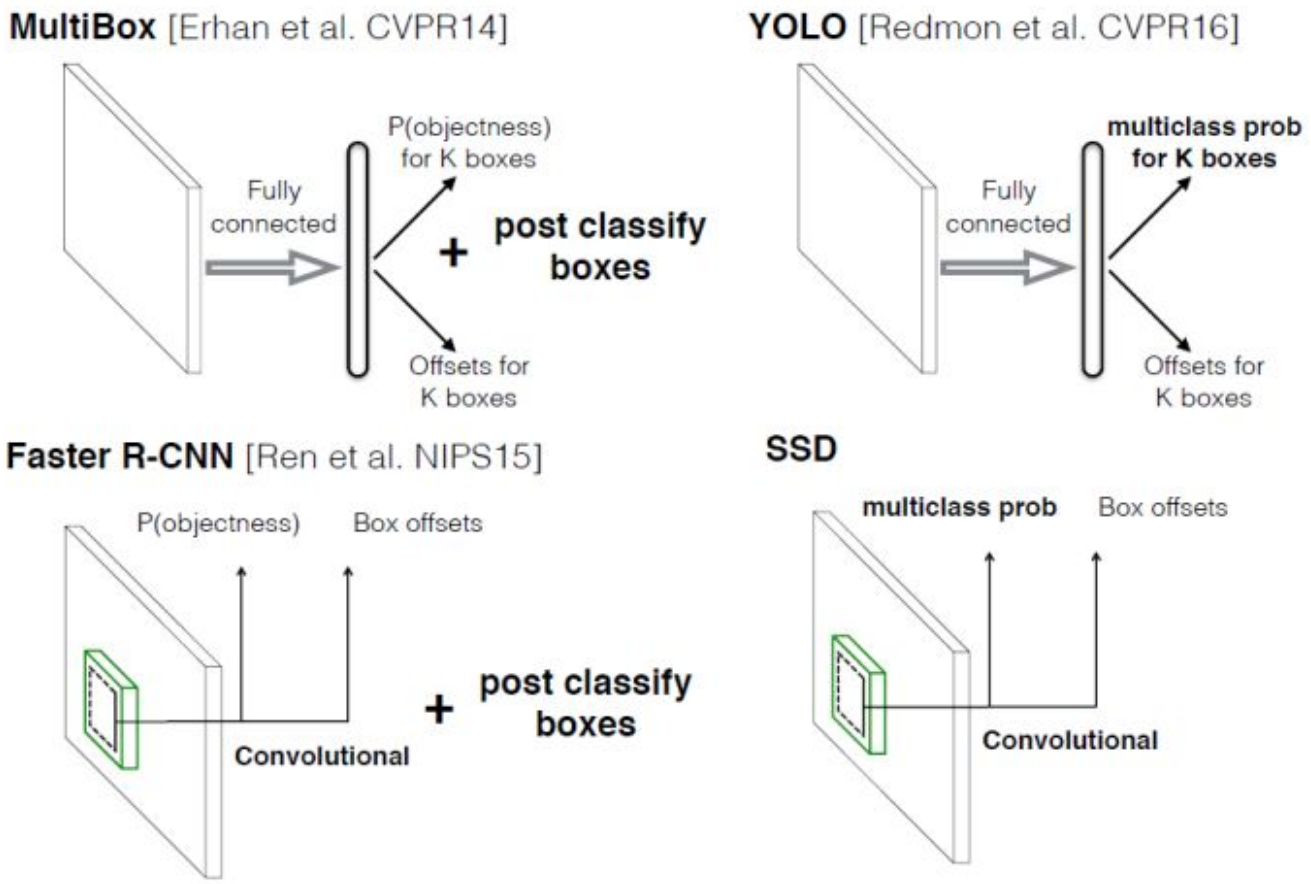
SSD英文名是（Single Shot MultiBox Detector），single shot指的是SSD算法属于**one-stage**方法，MultiBox说明SSD是**多框预测**。

上图可以看出SSD在准确度和速度（除了SSD512）上都比YOLO要好很多。

下图是不同算法的基本框架图，对于Faster R-CNN，其先通过CNN得到候选框，然后再进行分类和回归，而Yolo与SSD可以一步到位完成检测。相比于YOLO，SSD采用CNN来直接进行检测，而不是像YOLO那样在全连接层之后再做检测。

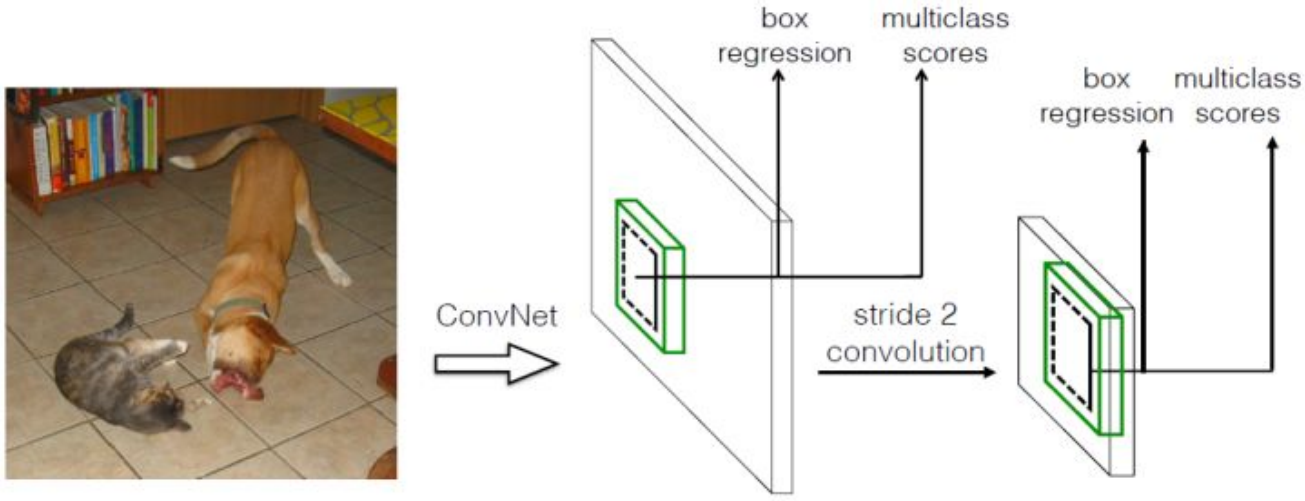
其实采用**卷积直接做检测**只是SSD相比于YOLO的其中一个不同点，另外还有两个重要的改变，一是**SSD提取不同尺度的特征图来做检测**，大尺度特征图（较靠前的特征图）用来检测小物体，小尺度特征图（**较靠后的特征图，感受野大**）用来检测大物体；二是SSD采用了**不同尺度和长宽比的先验框（Prior boxes, Default boxes**，在Faster R-CNN中叫做锚，

Anchors）。Yolo算法的缺点是难以检测小目标，而且定位不准，但是这几点重要的改进使得SSD在一定程度上克服这些缺点。



设计理念

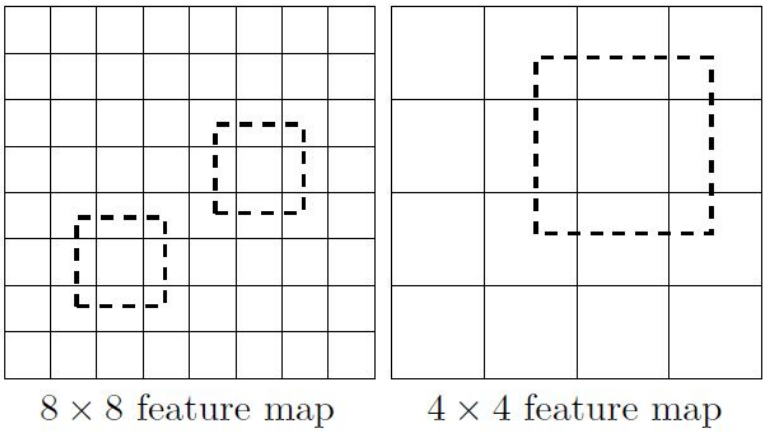
SSD和YOLO都是采用一个CNN网络来进行检测，但是却采用了多尺度的特征图，其基本架构如下图，下面将SSD核心设计理念总结为以下三点：



（1）采用多尺度特征图用于检测

所谓多尺度采用大小不同的特征图，CNN网络一般前面的特征图比较大，后面会逐渐采用**stride=2**的卷积或者**pool**来降低特征图大小，下图所示，一个比较大的特征图和一个比较小的特征图，他们**都用来做检测**。这样做的好处是**比较大的特**

征图用来检测相对较小的目标，而小的特征图负责检测大目标，8x8的特征图可以划分更多的单元，但是其每个单元的default box尺度比较小。



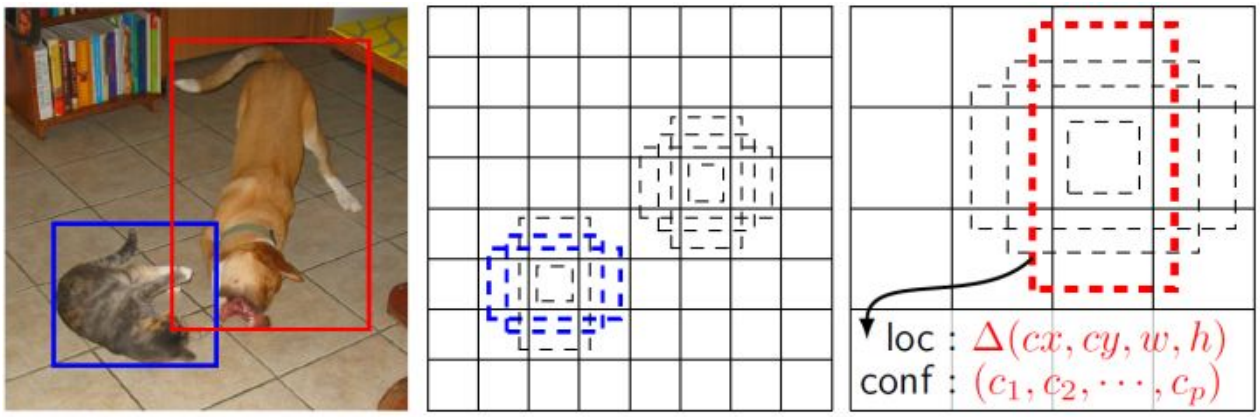
（2）采用卷积进行检测

与Yolo最后采用全连接层不同，SSD**直接采用卷积**对不同的特征图来**进行提取**检测结果。对于形状为$m \times n \times p$的特征图，只需要采用$3 \times 3 \times p$这样比较小的卷积核得到检测值。

（3）设置先验框default boxes

在YOLO中，每个单元预测多个边界框，但是其都是相对这个单元本身（正方块），但是真实目标的形状是多变的，Yolo需要在训练过程中自适应目标的形状。而SSD借鉴了Faster R-CNN中anchor的理念，**每个单元设置尺度或者长宽比不同的default boxes**，**预测的边界框（bounding boxes）是以这些先验框为基准的，在一定程度上减少训练难度。**

一般情况下，每个单元会设置多个default boxes，其尺度和长宽比存在差异，如下图所示，可以看到**每个单元使用了4个不同的default boxes**，图片中猫和狗**分别采用最适合它们形状的先验框**来进行训练，后面会详细讲解训练过程中的先验框匹配原则。



SSD的检测值也与YOLO不太一样，**对于每个单元cell的每个先验框default box，其都输出一套独立的检测值**，对于一个bouding box，主要分为两个部分：

- 第一个部分是**各个类别的置信度或者评分**
  - 值得注意的是**SSD将背景也当做了一个特殊的类别**，如果检测目标共有**c个类别**，SSD其实需要**预测c+1个置信度值**，其中第一个置信度是不含目标或者**属于背景的评分**。后面当我们说c个类别置信度时，请记住里面包含背景那个特殊的类别，即真实的检测类别只有c-1个。
  - 在预测过程中，置信度最高的那个类别就是边界框所属的类别。**特别的，当第一个置信度最高时，表示边界框中并不包含目标。**

- 第二个部分就是**边界框的location**，包含4个值(cx, cy, w, h)，分别表示边界框的中心坐标以及宽高。但是**真实预测其实只是预测边界框相对于先验框的转换值（offset）**。
  - 先验框（anchor）位置用$d = (d^{cx}, d^{cy}, d^w, d^h)$表示，其对应**预测的边界框**用$b = (b^{cx}, b^{cy}, b^w, b^h)$表示，那么**边界框的预测值l**其实是b相对于d的转换值：
  - $l^{cx} = (b^{cx} - d^{cx})/d^w,\ l^{cy} = (b^{cy} - d^{cy})/d^h$
  - $l^w = \log(b^w/d^w),\ l^h = \log(b^h/d^h)$

习惯上，我们称上面这个过程为**边界框的编码（encode）**，预测时，你需要反向这个过程，即进行**解码（decode）**，从预测值l中得到边界框的真实值b：

$$b^{cx} = d^w l^{cx} + d^{cx},\ b^{cy} = d^y l^{cy} + d^{cy}$$

$$b^w = d^w \exp(l^w),\ b^h = d^h \exp(l^h)$$

然而，在SSD的**caffe源码**实现中还有trick，那就是**设置variance超参数来调整检测值**，通过bool参数variance_encoded_in_target来控制两种模式。当其为true时，表示variance（方差）被包含在预测值中，就是上面那种情况，如果是false（大部分采用这种方式，训练更容易），就需要手动设置超参数variance，用来对l的4个值进行放缩，此时边界框需要这样解码：

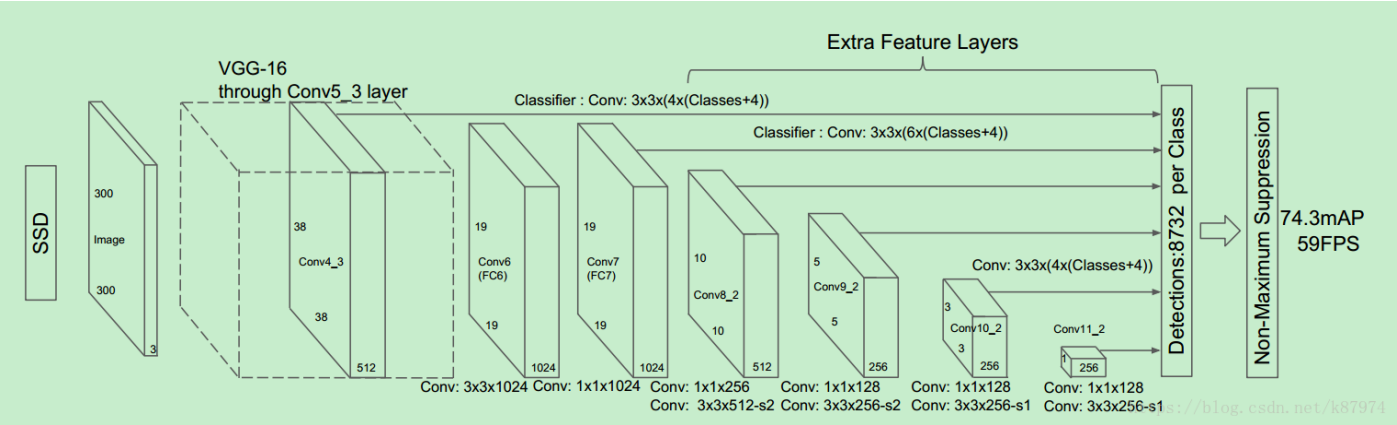
$$b^{cx} = d^w(variance[0] * l^{cx}) + d^{cx},\ b^{cy} = d^y(variance[1] * l^{cy}) + d^{cy}$$

$$b^w = d^w \exp(variance[2] * l^w),\ b^h = d^h \exp(variance[3] * l^h)$$

综上所述，对于一个大小$m \times n$的特征图，共有m x n 个单元，**每个单元设置的先验框数目记为k**，那么每个单元共需要$(c+4)k$个预测值，所有的单元共需要$(c+4)kmn$个预测值，**由于SSD采用卷积做检测，所以就需要**$(c+4)k$**个卷积核来完成这个特征图的检测过程。（卷积核参数共享）。**

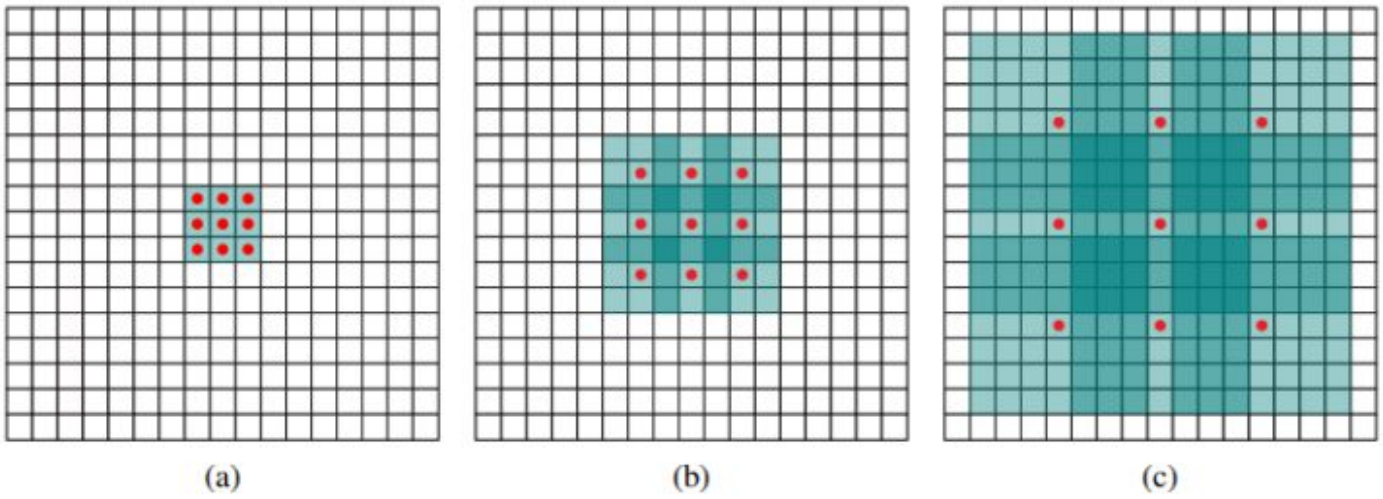
网络结构

SSD采用VGG16作为基础模型，然后在VGG16的基础上新增了卷几层来获得更多的特征图以用于检测。SSD网络结构如图所示：



很明显可以看出SSD利用了**多尺度的特征图**做检测，模型的输入图片大小是300×300（还可以是512×512），其与前者网络结构没有差别，只是最后新增一个卷积层。

采用VGG16做基础模型，首先VGG16是在ILSVRC CLS-LOC数据集预训练。然后借鉴了DeepLab-LargeFOV，分别**将VGG16的全连接层fc6和fc7转换成3×3卷积层 conv6和1×1卷积层conv7**，同时将**池化层pool5由原来的stride=2的2×2变成stride=1的3×3（猜想是不想reduce特征图大小）**，为了配合这种变化，采用了一种**Atrous Algorithm**，其实就是conv6采用扩展卷积或带孔卷积（Dilation Conv），其在不增加参数与模型复杂度的条件下**指数级扩大卷积的视野**，其使用**扩张率(dilation rate)参数**，来表示扩张的大小，如下图6所示，(a)是普通的3×3卷积，其视野就是3×3，(b)是扩张率为1，此时视野变成7×7，(c)扩张率为3时，视野扩大为15×15，但是视野的特征更稀疏了。Conv6采用3×3大小但dilation rate=6的扩展卷积。



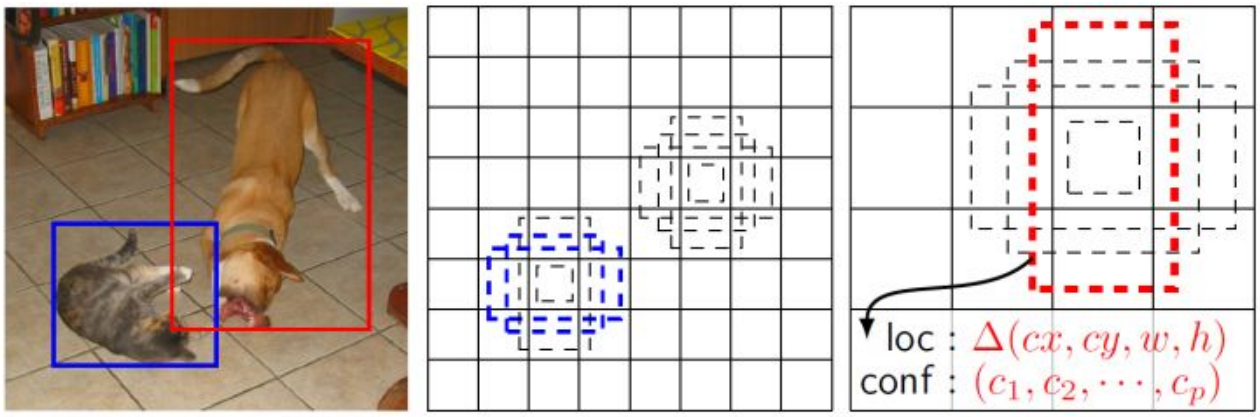
然后**移除dropout层和fc8层，并新增一系列卷积层，在检测数据集上做finetuing。**

其中VGG16中的**Conv4_3层**将作为用于检测的**第一个特征图**。conv4_3层特征图大小是38×38，但是该层比较靠前，其norm（范数）较大，所以在其后面增加了一个**L2 Normalization层**（参见ParseNet），以保证和后面的检测层差异不是很大，这个和Batch Normalization层不太一样，其**仅仅是对每个像素点在channle维度做归一化**，而**Batch Normalization层是在[batch_size, width, height]三个维度上做归一化。**归一化后一般设置一个可训练的放缩变量**gamma**，使用TF可以这样简单实现：

```
1 # l2norm (not bacth norm, spatial normalization)
2 def l2norm(x, scale, trainable=True, scope="L2Normalization"):
3     n_channels = x.get_shape().as_list()[-1]
4     l2_norm = tf.nn.l2_normalize(x, [3], epsilon=1e-12)
5     with tf.variable_scope(scope):
6         gamma = tf.get_variable("gamma", shape=[n_channels, ],
dtype=tf.float32,
7
initializer=tf.constant_initializer(scale),
```

```
8                                       trainable=trainable)
9          return l2_norm * gamma
```

从后面新增的卷积层中提取**Conv7，Conv8_2，Conv9_2，Conv10_2，Conv11_2作为检测所用的特征图**，加上**Conv4_3层**，共提取了**6个特征图**，其大小分别是 $(38,38),(19,19),(10,10),(5,5),(3,3),(1,1)$，但是**不同特征图单元cell设置的先验框数目不同（同一个特征图上每个单元设置的先验框是相同的，这里的数目指的是一个单元的先验框数目）**。先验框的设置，包括尺度（或者说大小）和长宽比两个方面。对于先验框的尺度，其遵守一个线性递增规则：**随着特征图大小降低，先验框尺度线性增加**



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

每一个feature map中的每一个小格子（cell）都包含多个default box，同时每个box对应loc（位置坐标）和conf（每个种类的得分）。

default box长宽比例默认有**四个**和**六个：**

四个default box是长宽比（**aspect ratios**）为（1:1）、（2:1）、（1:2）、（1:1）；六个则是添加了（1:3）、（3:1）

为什么会有两个（1:1）呢。这时候就要讲下论文中**Choosing scales and aspect ratios for default boxes**这段内容了。作者认为**不同的feature map应该有不同的比例**（一个大框一个小框，长宽比相同，大框是指不同feature map 相对于原图的尺寸比例不同），这是什么意思呢，代表的是**default box中这个1在原图中的尺寸是多大的**，计算公式如下所示：

$$s_k = s_{min} + \frac{s_{max} - s_{min}}{m-1}(k-1), k \in [1, m]$$

> S_k即代表在300*300输入中的比例，表示先验框大小相对于图片的比例
>
> m为当前的feature map是第几层； **m=5，因为一共有6个feature map，但是第一层（Conv4_3层）是单独设置的**
>
> k代表的是一共有多少层的feature map
>
> S_{min}和S_{max}代表的是第一层和最后一层所占的比例，比例的最小值和最大值，在**ssd300中为0.2-0.9。**
>
> 计算：
>
> 第一个feature map 是 conv4_3：默认设置比例为0.2/2=0.1， 此时k=1

第二个feature map 是 conv7： k=2，s = 0.2 +(0.7/4) x (2-1) =0.375 ，最后300x0.375 = 112.5，这个就是在这个feature map中比例为1的这个default box 的尺寸相对于原图300x300 的大小。

为什么default box的size有两个1吗?

作者在这有引入了一个$s'_k = \sqrt{s_k s_{k+1}}$，也就是每个特征图都设置了两个长宽比为1大小不同的正方形default box。有的小伙伴可能会有疑问，这有了**Sk+1**则需要多出来一部分的S_k啊，是的没错，最后一个特征图需要参考$s_{m+1} = 300 \times 105/100 = 315$来计算$s'_m$，因此每个特征图（的每个cell）都有6个default box $\{1, 2, 3, \frac{1}{2}, \frac{1}{3}, 1'\}$（**aspect ratios**），但是在实现时， Conv4_3，Conv10_2，Conv11_2仅仅使用4个先验框（default box），不使用长宽比为3,1/3的先验框（default box）。作者的代码中就添加了两层，**第一层取0.1**，**最后一层取1**。

注：对于第一个特征图，先验框（default box）的尺度比例一般$s_{min}/2 = 0.1$，则尺度为300x0.1=30。

对于后面的特征图，先验框尺度比例按照上面公式线性增加，先将尺度比例放大100倍，然后再计算得到Sk，然后再将Sk除以100，再乘以图片大小，就可以得到各个特征图的先验框的size$30, 60, 111, 162, 213, 264$

那么S怎么用呢？按如下方式计算先验框的宽高（这里的Sk是上面求得的各个特征图的先验框的实际size，不再是尺度比例）：

$$w_k^a = s_k\sqrt{a_r},\ h_k^a = s_k/\sqrt{a_r}$$

a_r代表的是之前提到的default box（**aspect ratios**）比例，即$a_r \in \{1, 2, 3, \frac{1}{2}, \frac{1}{3}\}$

对于default box中心点的值取值为：

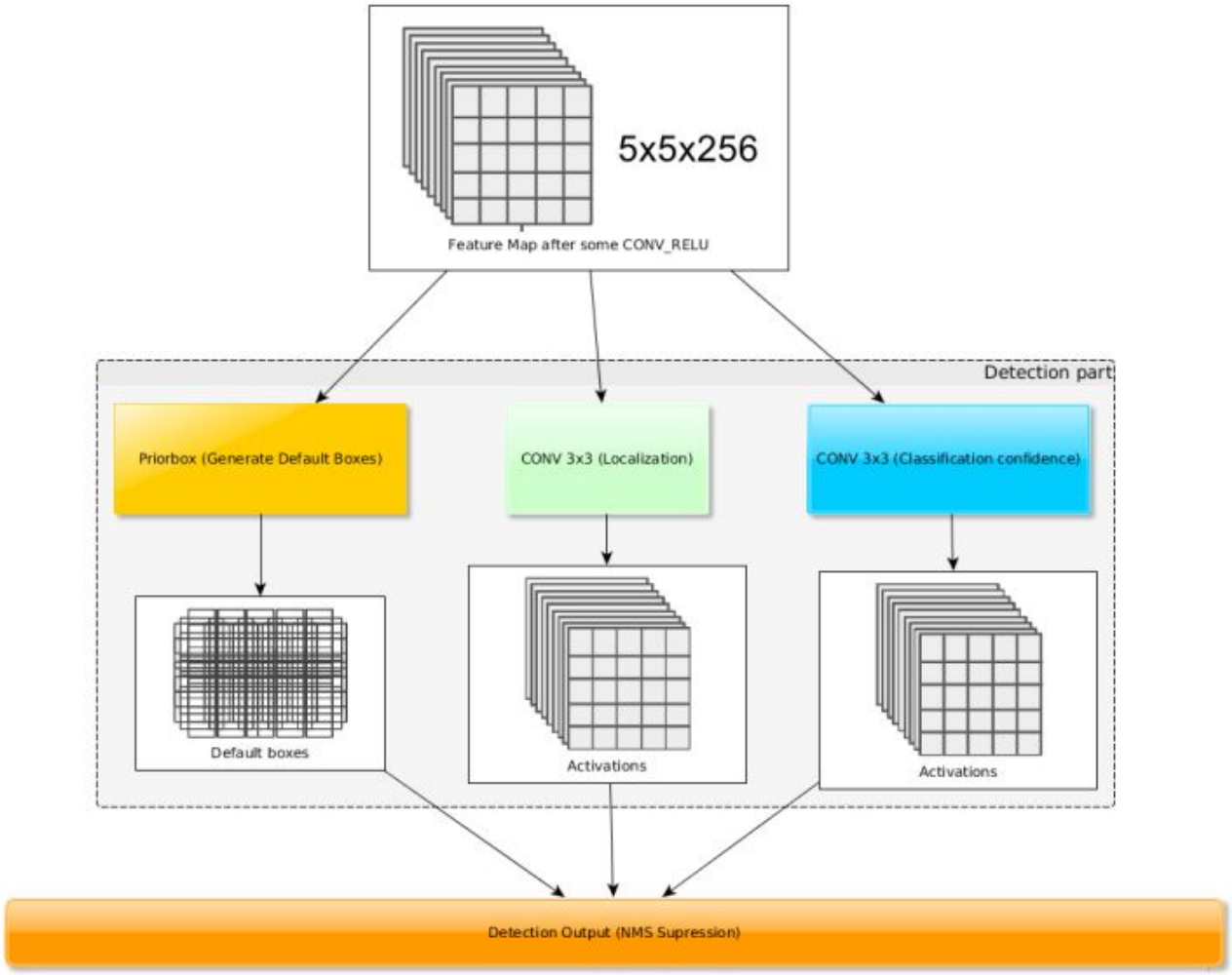
$$(\frac{i+0.5}{|f_k|}, \frac{j+0.5}{|f_k|}), i, j \in [0, |f_k|)$$

其中i，j代表在feature map中的水平和垂直的第几格

f_k代表的是feature map的size

每个单元的先验框中心点分布在各单元的中心

得到特征图后，需要对特征图进行卷积得到检测结果，下图给出了一个5x5大小的特征图检测过程：



Priorbox是得到先验框，生成规则前面已经讲了。

检测值包含两个部分：类别置信度和边界框位置，各采用一次3x3卷积来进行完成。

n_k是该特征图所采用的先验框数目，那么类别置信度需要的卷积核数量为$n_k \times c$，而边界框位置需要的卷积核数量为$n_k \times 4$。由于**每个先验框都会预测一个边界框**，所以SSD300一共可以预测

Conv4_3 得到的feature map大小为38*38：38*38*4 = 5776

Conv7 得到的feature map大小为19*19：19*19*6 = 2166

Conv8_2 得到的feature map大小为10*10：10*10*6 = 600

Conv9_2 得到的feature map大小为5 * 5 ： 5 * 5 * 6 = 150

Conv10_2得到的feature map大小为3 * 3 ： 3 * 3 * 4 = 36

Conv11_2得到的feature map大小为1 * 1 ： 1 * 1 * 4 = 4

最后结果为：8732

$38 \times 38 \times 4 + 19 \times 19 \times 6 + 10 \times 10 \times 6 + 5 \times 5 \times 6 + 3 \times 3 \times 4 + 1 \times 1 \times 4 = 8732$个边界框，这是一个相当庞大的数字，所以说SSD本质上是密集采样。

训练过程

（1）先验框匹配

在训练过程中，首先要确定训练图片中的ground truth（真实目标）与哪个先验框来进行匹配，与之**匹配的先验框所对应的边界框将负责预测它**。

Yolo中，ground truth的中心落在哪个单元格，该单元格中与其IOU最大的边界框负责预测它。

SSD中，先验框与ground truth的匹配原则又两点：

> 1、每个ground truth找到与其IOU最大的先验框，互相匹配。该先验框称为正样本（先验框对应的预测box）
>
> 若有个先验框没有与ground truth匹配，就只能与背景匹配，就是负样本。（一个图片中ground truth少，但先验框多，这样匹配，很多先验框会是负样本，正负样本不均衡）。
>
> 2、对剩余未匹配先验框，若某个ground truth的IOU大于某个阈值（一般是0.5），那么该先验框也与这个ground truth进行匹配。
>
> 这样ground truth可能与多个先验框匹配
>
> FP：负样本　　TP：正样本

尽管一个ground truth可以与多个先验框匹配，但是ground truth相对于先验框还是太少了，所以负样本会很多。为保证正负样本尽量均衡，SSD采用了**hard negative mining**，先将每一个物体位置上对应 predictions（default boxes）是 negative 的 boxes 进行排序，按照 default boxes 的 confidence 的大小。 选择最高的几个，保证最后 negatives、positives 的比例接近3:1

（2）损失函数

损失函数定义为位置误差（locatization loss， loc）与置信度误差（confidence loss, conf）的加权和：

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

> 其中 N 是先验框的正样本数量。
>
> 这里 $x_{ij}^{p}\in\{1,0\}$ 为一个指示参数，当 $x_{ij}^{p}=1$ 时表示第 i 个先验框与第 j 个ground truth匹配，并且ground truth的类别为 p 。
>
> c 为类别置信度预测值。
>
> l 为先验框的所对应边界框的位置预测值
>
> g 是ground truth的位置参数

对于**位置误差**，其采用**Smooth L1 loss**，定义如下：

$$L_{loc}(x,l,g)=\sum_{i\in Pos}^{N}\sum_{m\in\{cx,cy,w,h\}}x_{ij}^{k}\text{smooth}_{\text{L1}}(l_i^m-\hat{g}_j^m)$$

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

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

$$\text{smooth}_{L_1}(x)=\begin{cases}0.5x^2 & \text{if } |x|<1\\ |x|-0.5 & \text{otherwise,}\end{cases}$$

> 由于 x_{ij}^p 的存在，所以**位置误差仅针对正样本进行计算**。值得注意的是，**要先对ground truth的 g 进行编码得到 $\hat{g}$ ，因为预测值 l 也是编码值，若设置variance_encoded_in_target=True，编码时要加上variance：**
>
> $$\hat{g}_j^{cx}=(g_j^{cx}-d_i^{cx})/d_i^w/variance[0],\hat{g}_j^{cy}=(g_j^{cy}-d_i^{cy})/d_i^h/variance[1]$$
>
> $$\hat{g}_j^{w}=\log(g_j^w/d_i^w)/variance[2],\ \hat{g}_j^{h}=\log(g_j^h/d_i^h)/variance[3]$$

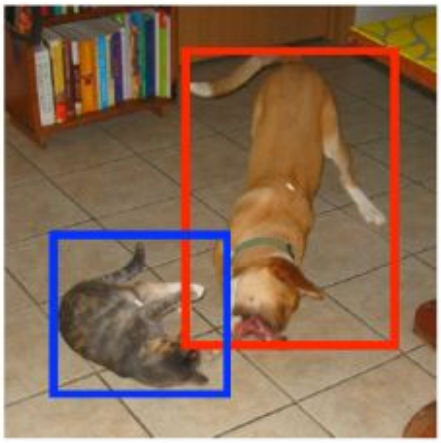
对于**置信度误差**，其采用**softmax loss**:

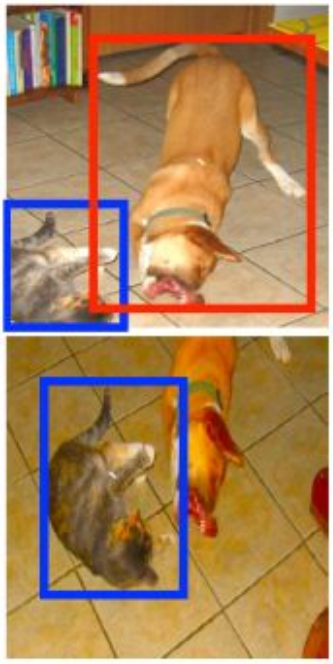
$$L_{conf}(x,c)=-\sum_{i\in Pos}^{N}x_{ij}^p log(\hat{c}_i^p)-\sum_{i\in Neg}log(\hat{c}_i^0) \quad \text{where} \quad \hat{c}_i^p=\frac{\exp(c_i^p)}{\sum_p\exp(c_i^p)}$$

权重系数 α 通过交叉验证设置为1。

（3）数据扩增

采用数据扩增（**Data Augmentation**）可以提升SSD的性能，主要采用的技术有**水平翻转（horizontal flip）**，**随机裁剪加颜色扭曲（random crop & color distortion）**，**随机采集块域（Randomly sample a patch）**（获取小目标训练样本），如下图所示：





预测过程

确定预测框类别（置信度最大者）与置信度值，并且过滤掉属于背景的预测框，过滤掉置信度阈值较低的预测框；

对留下的预测框进行编码，得到真实的位置参数（解码后还需要clip，防止预测框位置超出图片）；

解码之后，根据置信度进行降序排列，保留top-k个预测框；

进行NMS算法，过滤掉那些重叠度比较大的预测框，最后剩余的预测框就是检测结果了。

性能评估

首先整体看一下SSD在VOC2007，VOC2012及COCO数据集上的性能，如表1所示。相比之下，SSD512的性能会更好一些。加*的表示使用了image expansion data augmentation（通过zoom out来创造小的训练样本）技巧来提升SSD在小目标上的检测效果，所以性能会有所提升。

Method	VOC2007 test		VOC2012 test		COCO test-dev2015 trainval35k		
	07+12	07+12+COCO	07++12	07++12+COCO			
	0.5	0.5	0.5	0.5	0.5:0.95	0.5	0.75
SSD300	74.3	79.6	72.4	77.5	23.2	41.2	23.4
SSD512	76.8	81.6	74.9	80.0	26.8	46.5	27.8
SSD300*	77.2	81.2	75.8	79.3	25.1	43.1	25.8
SSD512*	**79.8**	**83.2**	**78.5**	**82.2**	**28.8**	**48.5**	**30.3**

SSD与其它检测算法的对比结果（在VOC2007数据集）如表2所示，基本可以看到，SSD与Faster R-CNN有同样的准确度，并且与Yolo具有同样较快地检测速度。

Method	mAP	FPS	batch size	# Boxes	Input resolution
Faster R-CNN (VGG16)	73.2	7	1	~ 6000	~ 1000 × 600
Fast YOLO	52.7	155	1	98	448 × 448
YOLO (VGG16)	66.4	21	1	98	448 × 448
SSD300	74.3	46	1	8732	300 × 300
SSD512	76.8	19	1	24564	512 × 512
SSD300	74.3	59	8	8732	300 × 300
SSD512	76.8	22	8	24564	512 × 512

文章还对SSD的各个trick做了更为细致的分析，表3为不同的trick组合对SSD的性能影响，从表中可以得出如下结论：

- 数据扩增技术很重要，对于mAP的提升很大；
- 使用不同长宽比的先验框可以得到更好的结果；

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

同样的，采用多尺度的特征图用于检测也是至关重要的，这可以从表4中看出：

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

补充

1、L2 normalization

2、hard nevigating mining

分类: 待删除

keepgoing18
关注 - 0
粉丝 - 35
+加关注

« 上一篇： 1x1卷积核作用
» 下一篇： SSD-2（代码部分介绍）

posted on 2019-02-06 19:21 keepgoing18 阅读(6169) 评论(0) 编辑 收藏