Object Detection Summary

1. SPP-Net

Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition [Paper] [Code]: Kaiming He et.al

Net structure

Main structure: ZF + Spatial Pyramid Pooling layer

Spatial Pyramid Pooling Layer: SPP是BOW（Bag of Words）模型的扩展，**将图像从粗粒度到细粒度进行划分，然后将局部特征进行整合（在CNN流行之前很常用）。**

SPP-net将最后一层conv层之后的pooling层用SPP层进行替换，在每种粒度输出kM维向量（k：# filter； M：该粒度包含的bin的数量）。其中最粗的粒度只输出一个bin，也就是global pooling, 而global average pooling可以用用于减小模型尺寸，降低过拟合，提高精度。

特点：可以接受不同长宽比，不同大小的输入图片，输出固定大小的向量。

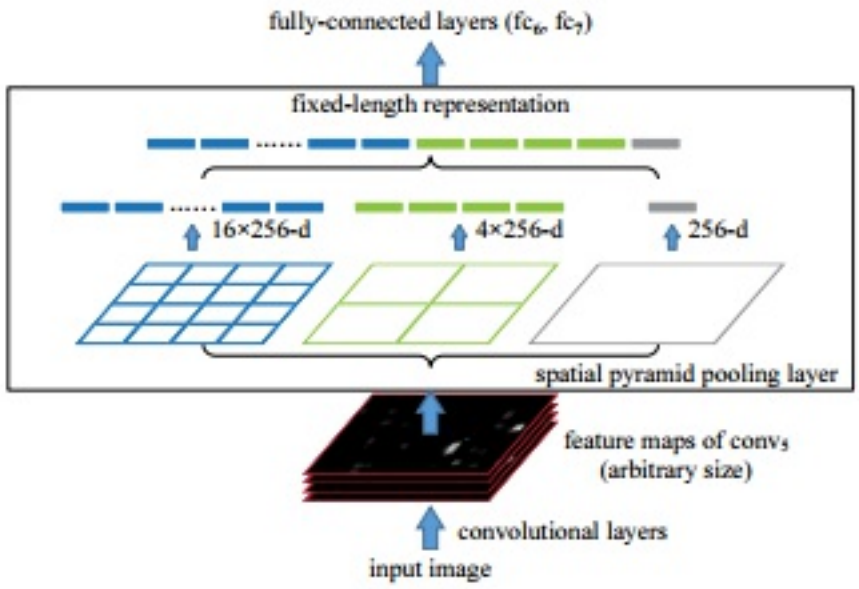


Figure 3: A network structure with a **spatial pyramid pooling layer**. Here 256 is the filter number of the $conv_5$ layer, and $conv_5$ is the last convolutional layer.

Model Training

Single size training
输入：固定224 x 224大小
conv5 feature map输出：a x a (eg: 13 x 13)

SPP: n x n bins, 滑动窗池化，win=ceiling(a/n) stride=floor(a/n)

Multi-size training
输入： 224 x 224和180 x 180 (224 resize 产生)
由于SPP层输出的大小只与粒度有关与输入无关，所以224和180两种图片输入的模型的各层权重可以完全共享，因此在训练的时候用一个网络训练一个epoch之后保留权重用另一个网络训练一个epoch，如此迭代。

SPP Net for object detection

与RCNN相比，不同与RCNN的proposal选择在输入图片上进行，SPP Net在featurem map上进行操作，不需要对每个proposal进行重新的卷积计算，速度提高100倍+（操作与fast RCNN类似）。

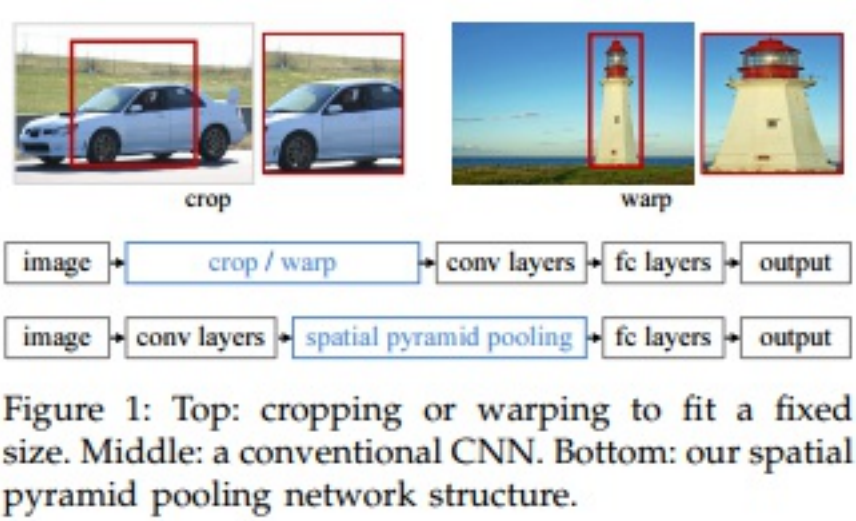


Figure 1: Top: cropping or warping to fit a fixed size. Middle: a conventional CNN. Bottom: our spatial pyramid pooling network structure.

Selective Search 提取2000个proposal，resize图像min(w, h) = s ∈ S = {480， 576， 688， 864， 1200}后提取特征，对每个候选proposal用4层SP（1x1,2x2,3x3,6x6, total 50 bins）pool，输出256*50=12800d向量输入到fc层，最后用SVM分类。正例为bbox，负例为与正例IoU<0.3的box（去掉与其他样本iou>0.7的负例）

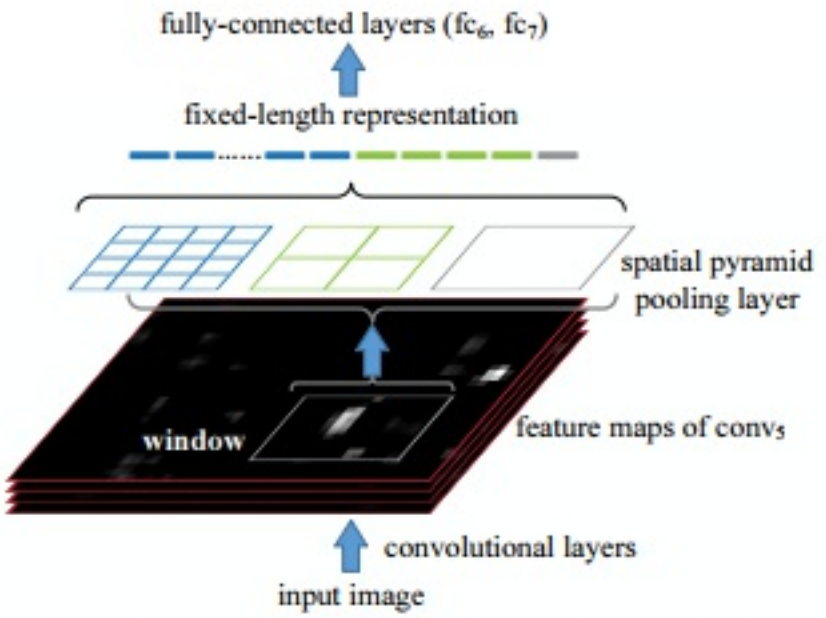


Figure 5: Pooling features from arbitrary windows on feature maps. The feature maps are computed from the entire image. The pooling is performed in candidate windows.

---

2. DeepID-Net

DeepID-Net: Deformable Deep Convolutional Neural Networks for Object Detection [Paper]: Xiaoou training
An extension of R-CNN: box pre-training, cascade on region proposals, deformation layers and context representations

Network structure

method: SS提取proposal; RCNN剔除可能是背景的proposal；crop bbox的图片输入到DeepID-Net输出各类的confidence（200类）；用计算全图的分类score（1000类）作为contextual信息来优化前面bbox的分类score；平均多个deep model的输出来提高Detection精度；RCNN进行bbox regression。

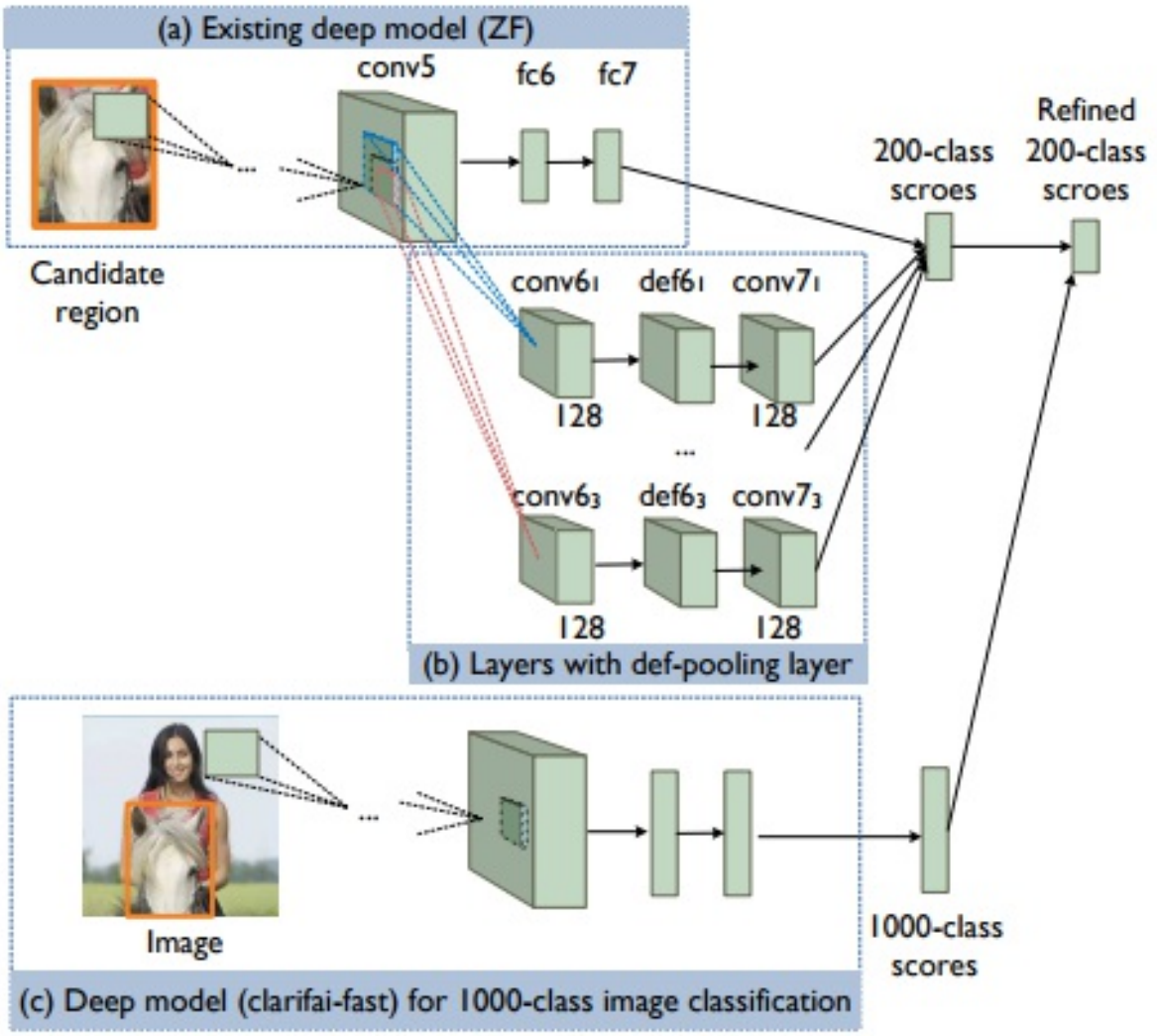


Figure 3. Architecture of DeepID-Net with three parts: (a) baseline deep model, which is ZF [54] in our best-performing single-model detector; (b) layers of part filters with variable sizes and def-pooling layers; (c) deep model to obtain 1000-class image classification scores. The 1000-class image classification scores are used to refine the 200-class bounding box classification scores.

def-pooling layer:conv层输出大小为W x H，C个part detection maps，记Mc为第c个part detection map，Mc的第(i,j)个元素是$m_c^{(i,j)}$。def-pooling层从Mc取一个大小为(2R+1)x(2R+1)，中心为（$s_x.x, s_y.y$）的区域，输出如下，大小为（$W/s_x, H/s_y$）：

$$b_c^{(x,y)} = \max_{\delta_x,\delta_y \in \{-R,\cdots,R\}} \{m_c^{\mathbf{z}_{\delta_x,\delta_y}} - \sum_{n=1}^{N} a_{c,n} d_{c,n}^{\delta_x,\delta_y}\}, \quad (1)$$

$$\text{where } \mathbf{z}_{\delta_x,\delta_y} = (s_x \cdot x + \delta_x, s_y \cdot y + \delta_y).$$

其中$m_c^{z_{\delta_x,\delta_y}}$是把第c个part放到deformed position z_{δ_x,δ_y}的visual score；$a_{c,n}$和$d_{c,n}^{\delta_x,\delta_y}$通过BP学习；

$\Sigma_{n=1}^{N} a_{c,n} d_{c,n}^{\delta_x,\delta_y}$是把part从假定的anchor处（$s_x.x$, $s_y.y$放到z_{δ_x,δ_y}的惩罚。

而当N=1，a_n=1, $d_1^{\delta_x,\delta_y}$=0 for $|\delta x|, |\delta y| \le k$，$d_1^{\delta_x,\delta_y} = \infty$ for $|\delta x|, |\delta y| > k$，此时def-pooling layer相当于max-pooling with kernel size k。

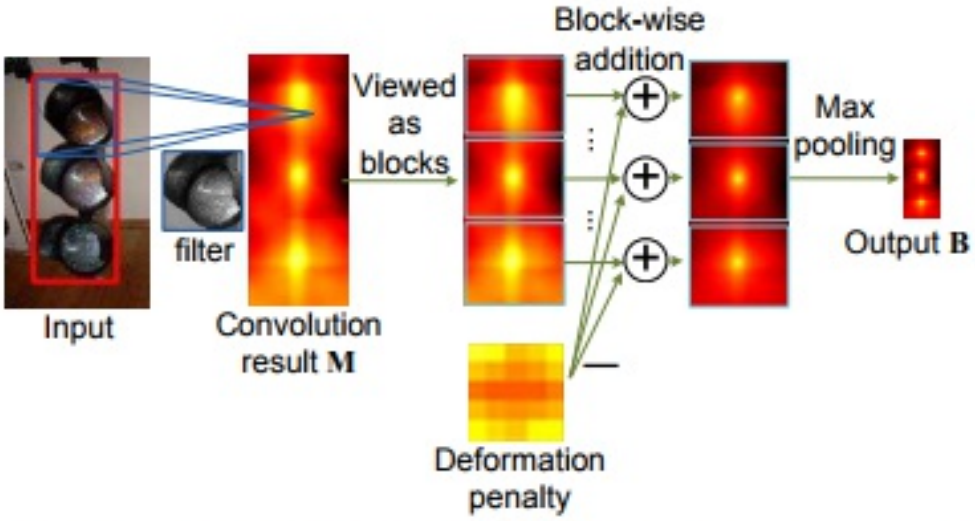


Figure 4. Def-pooling layer. The part detection map and the deformation penalty are summed up. Block-wise max pooling is then performed on the summed map to obtain the output **B** of size $\frac{H}{s_y} \times \frac{W}{s_x}$ (3×1 in this example).

优势：1. 可以替换conv层，学习不同size和semantic meaning的deformation parts；比如高级用大的parts（上肢），中级的用中间级别的part（人头），低级的用小的part（嘴）。2. def-pooling layer可以同时检测多个有相同visual cur的deformable parts（如上图所示）；通过def-pooling layer学习的visual pattern可以在多个class共享。

contextual modeling: concatenate 1000-class score as contextual score with 200-class score to form a 1200d feature vector.

multiple deep models: models trained in different structures, pretraining schemes, loss functions (hinge loss), adding def-pooling layer or not, doing bbox rejection or not.

---

3. YOLO

You Only Look Once: Unified, Real-Time Object Detection [Paper] [Code]: RGB et.al
Other references: TensorFlow YOLO object detection on Android

特点：从全图提取特征来同时预测所有类的所有bbox，允许end-to-end training，检测实时性高。

Network structure

原理：首先将图片切分成S x S个格子，如果物体的中心在一个格子内，那么这个格子就需要对检测的该物体

负责。每个格子预测B个bbox以及这些bbox的confidence score - Pr(Object)*IOU_{pred}^{truth}即反映了格子包含物体的可能性以及位置坐标的精度。每个bbox预测值有5个：（x, y, w, h, confidence）；而每个格子预测C个条件类概率（$Pr(Class_i|Object)$。只有当格子包含物体的时候才计算这个概率，并且**计算class的概率每个格子只做一次，与bbox的数量(B)无关**。在test的时候将条件类概率与之前的confidence相乘即可得到每个bbox的每个类的检测概率（这个值包含了预得的测类以及坐标的准确性，如下）。

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

在Pascal数据集下S=7， B=2, C=20，所以最后的预测为7x7x(5x2+20)的Tensor。

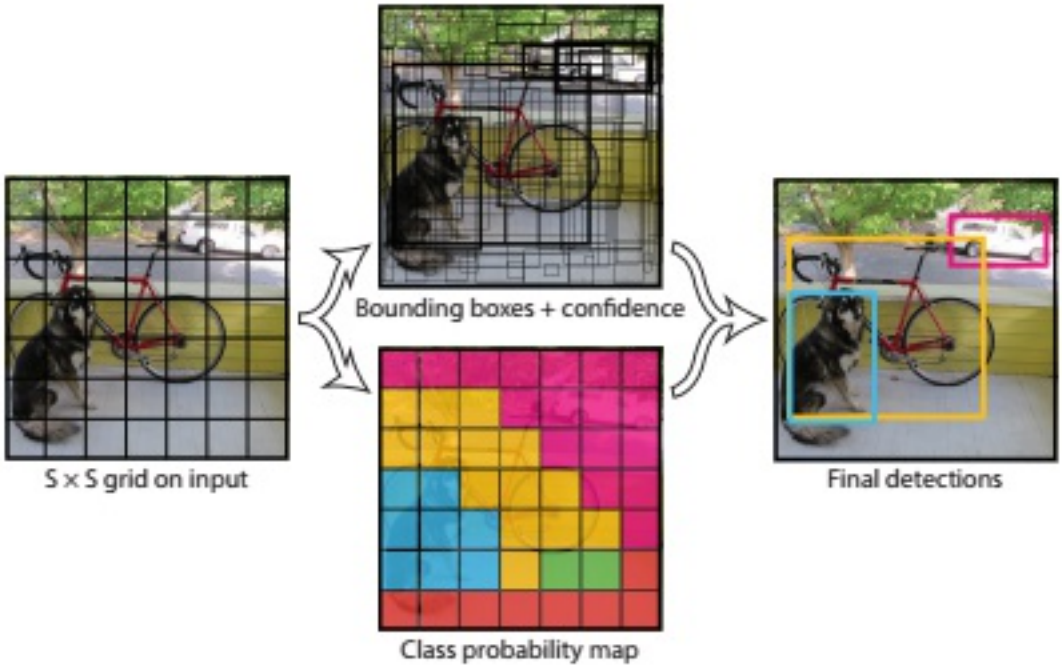


Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

网络结构： 24 conv + 2 fc; fast version为9 conv + 2 fc。

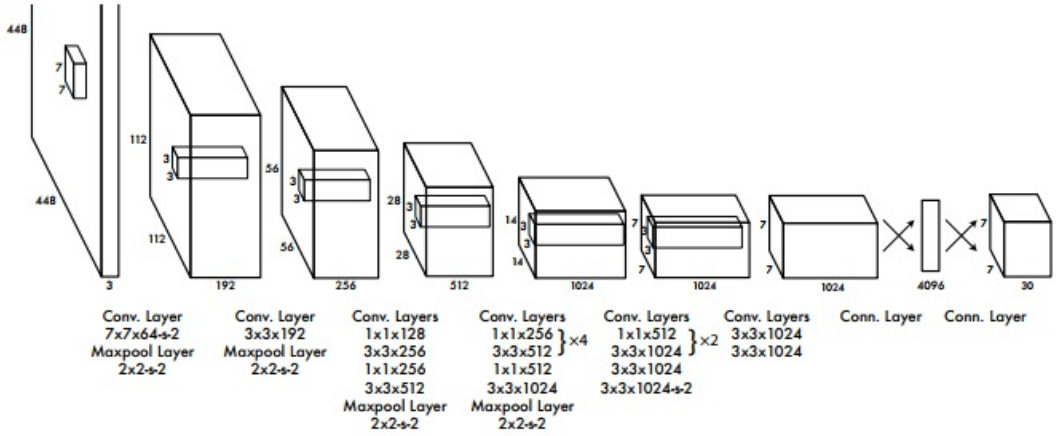


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224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

Model Training

首先用ImageNet 1000 class + 上图前20层+average pooling+fc预训练（top5:88%）**分类**,输入的图片大小为224x224。然后在这基础上加4conv+2fc（weights随机初始化）并且图片输入大小提高到448x448训练 **检测**。其中 **预处理** 根据输入图片的大小\normalize bbox的宽和高到0-1，而x,y则基于特定格子参数化到0-1。

最后一层的激活函数是linear activation，其他层为Leaky ReLU:

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

每个格子都会预测多个bbox，但是训练的时候我们对应每个物体只想要一个bbox predictor。如果某个predictor的预测值与某个物体真实值之间存在最大IOU，那么这个predictor就是负责这个预测这个物体的。

Loss function

loss function:

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

其中，λ_{coord}=5, λ_{noobj}=0.5用于增大bbox预测坐标的loss，减少没有包含物体格子的预测confidence的loss（因为大部分格子木有包含物体）；而为了使小的bbox预测坐标时的偏差比大的bbox时影响更大，对w和h进行了开根处理。I_i^{obj}表示格子i含有是否物体，有就是1；I_{ij}^{obj}表示i个格子中使用第j个bbox predictor是负责预测该物体的（分类误差只有在格子有物体的时候计算，坐标误差只有在predictor是负责预测这个物体的时候计算）。

数据集：VOC2007+VOC2012

Inference

98(7x7x2) bbox + NMS

Limitation

- 每个格子只有2个bbox，以及只能属于1类，因此集群的小物体很难检
- 预测新的尺寸或者异常长宽比的bbox的坐标能力较差，相比与faster R-CNN，定位误差较大。
- 将大小不同个的bbox的error按一样的方式处理（小bbox的误差对IoU影响较大）

---

4. YOLOv2(YOLO9000)

YOLO9000: Better, Faster, Stronger [Paper] [Code]

Improvement

1). 每层conv后添加 **BN** (Batch Normalization)；

2). **分类** 模型也用448x448的输入图片训练（提高训练输入图像的分辨率），然后fine-tune用于检测模型；

3).**Anchor boxes**
使用anchor boxes代替conv层后的全连接层预测的bbox并用anchor box来预测class（在yolo中是由每个格子进行预测）和objectiveness（98 bbox in YOLO --> >1000(13x13x9) bbox in YOLOv2)

用k-means clustering代替手动预先确定anchor box的大小(聚类的距离函数为d(box, centroid)=1−IOU(box, centroid)而非euclidean distance)

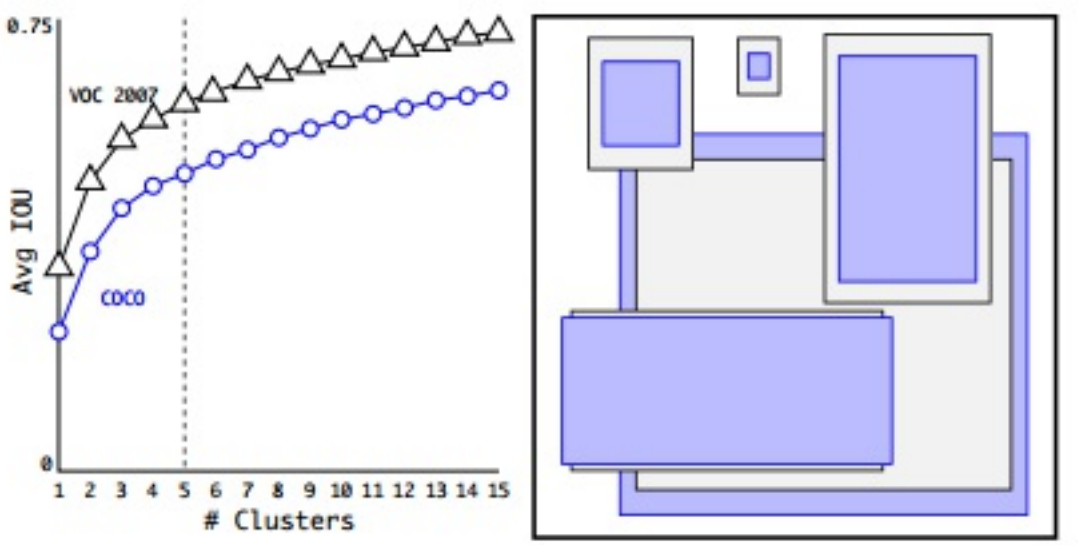


Figure 2: Clustering box dimensions on VOC and COCO. We run k-means clustering on the dimensions of bounding boxes to get good priors for our model. The left image shows the average IOU we get with various choices for k. We find that $k = 5$ gives a good tradeoff for recall vs. complexity of the model.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

4). 使用相对grid cell的位置来预测坐标：将真实bbox bound to 0-1，而将预测值用logistics activation也限制到0-1。其中pw和ph是预设的anchor box的宽和高。

$$
\begin{aligned}
b_x &= \sigma(t_x) + c_x \\
b_y &= \sigma(t_y) + c_y \\
b_w &= p_w e^{t_w} \\
b_h &= p_h e^{t_h} \\
Pr(\text{object}) * IOU(b, \text{object}) &= \sigma(t_o)
\end{aligned}
$$

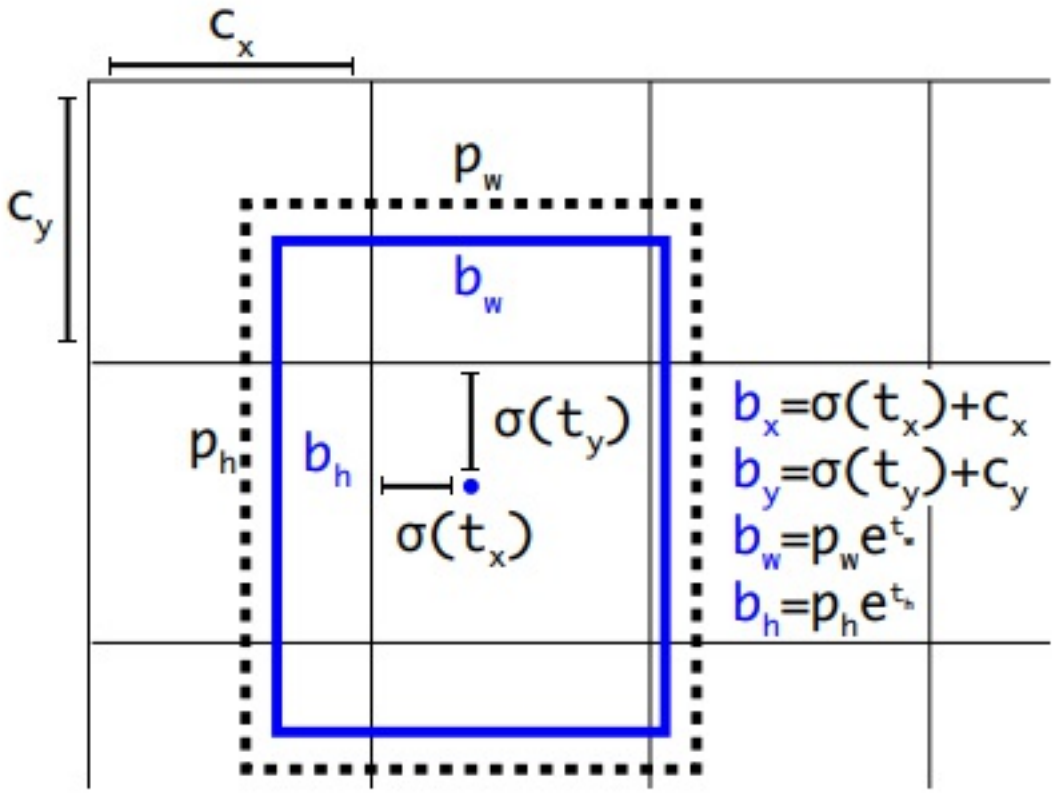


Figure 3: Bounding boxes with dimension priors and location prediction. We predict the width and height of the box as offsets from cluster centroids. We predict the center coordinates of the box relative to the location of filter application using a sigmoid function.

5). 更细粒度的feature来捕获小物体（ **passthrough层** ）：使用最后一层feature map(13x13)的前一层(26x26x512)，将其转成13x13x2048（类似与resnet中identity mapping）并与最后一层concatenate。

6). 由于YOLOv2只包含conv层和pooling层，没有fc层，因此输入 **任意大小** 的图像。每10个batch选择一个新的输入图像大小 - {320, 352, ..., 608}。

7). 用 **Darknet** 代替VGG作为分类模型作为YOLOv2的基础，减少计算量提高速度。在检测的时候用3x3 conv层（1024 filter）+ 1x1 conv层代替Darknet最后的conv层。加一层passthrough层将最后的3x3x512层（feature map:14x14）与最后的3x3x1024层(feature map: 7x7)相连。

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

8). Hierarchical classification 使用WordTree来预测每个节点的条件概率：比如如果一个物体标记为terrier，那么同时也被标注为dog和mammal。在训练/预测的时候将所有WordTree的中间节点也加入预测的label（VOC：1000-> 1369），然后计算联合概率。

$$
\begin{aligned}
Pr(\text{Norfolk terrier}) &= Pr(\text{Norfolk terrier}|\text{terrier}) \\
&\quad * Pr(\text{terrier}|\text{hunting dog}) \\
&\quad\quad * \ldots * \\
&\quad * Pr(\text{mammal}|Pr(\text{animal}) \\
&\quad * Pr(\text{animal}|\text{physical object})
\end{aligned}
$$

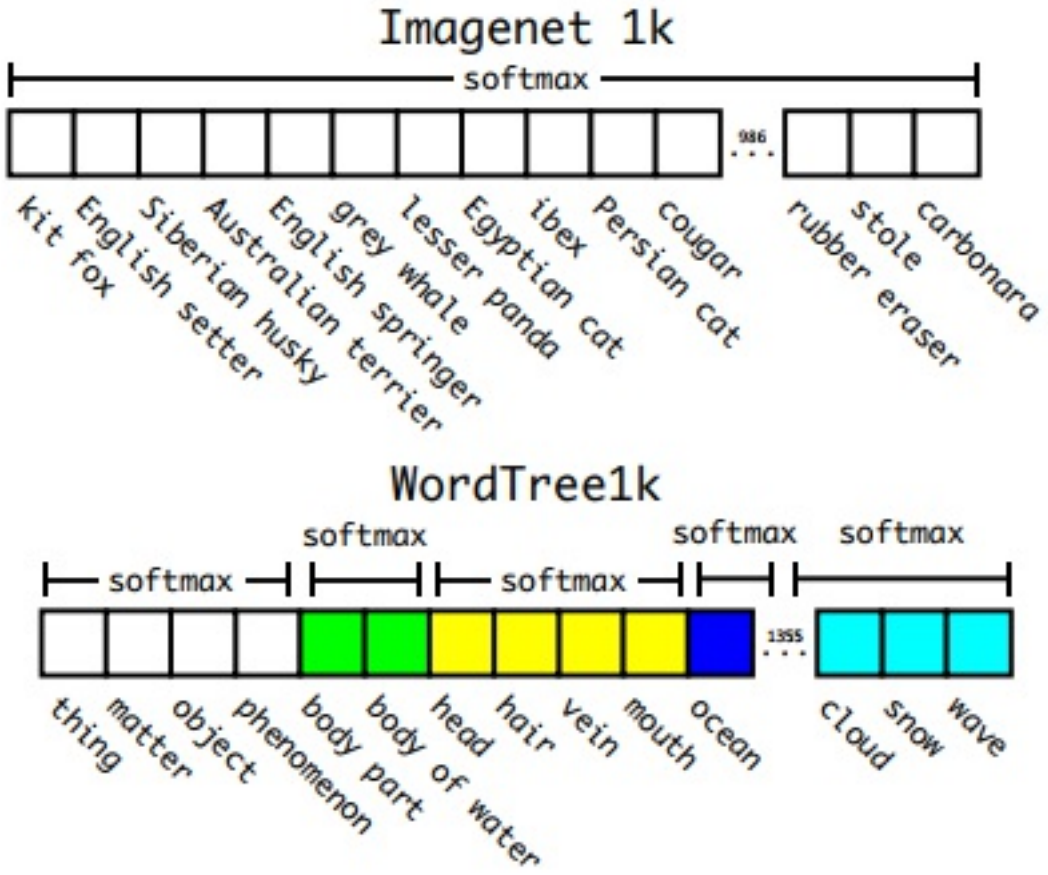


Figure 5: Prediction on ImageNet vs WordTree. Most ImageNet models use one large softmax to predict a probability distribution. Using WordTree we perform multiple softmax operations over co-hyponyms.

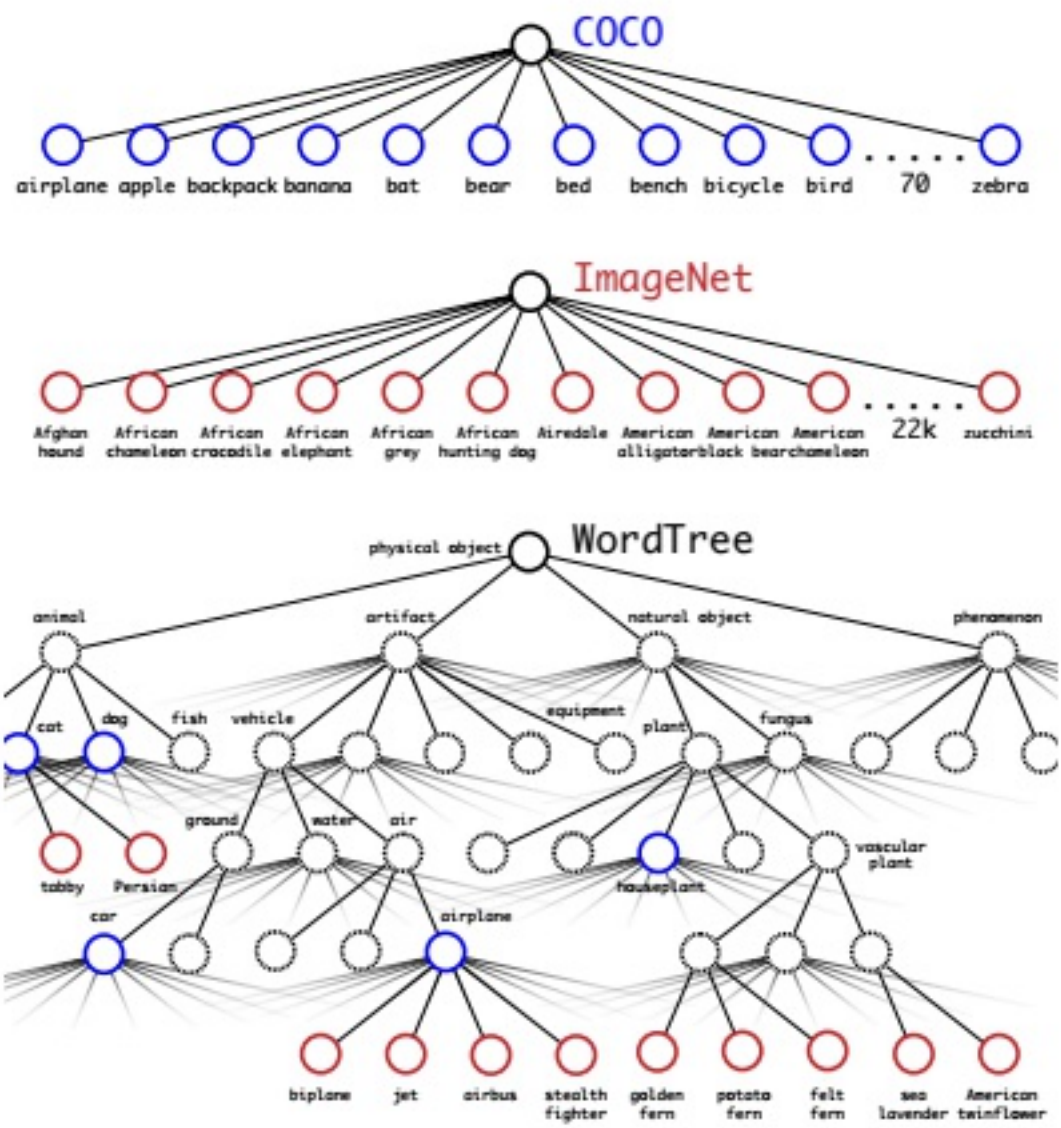


Figure 6: Combining datasets using WordTree hierarchy. Using the WordNet concept graph we build a hierarchical tree of visual concepts. Then we can merge datasets together by mapping the classes in the dataset to synsets in the tree. This is a simplified view of WordTree for illustration purposes.

---

5. AttentionNet

AttentionNet: Aggregating Weak Directions for Accurate Object Detection [Paper]

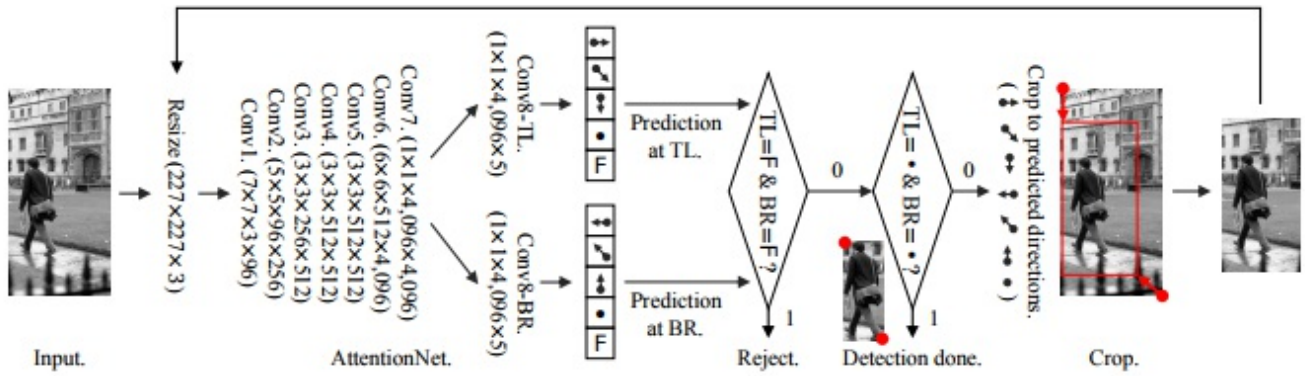


Figure 2. A pipeline of our detection framework. AttentionNet is composed of two final layers for top-left (TL) and bottom-right (BR) of the input image domain. Each of them outputs a direction (→ ↘ ↓ for TL, ← ↖ ↑ for BR) where each corner of the image should go to for the next step, or a "stop" sign (•), or "non-human" sign (F). When AttentionNet outputs "non-human" in both layers, the image is rejected. The image is cropped according to the weak directions and fed to AttentionNet again, until it meets "stop" in both layers.

6. DenseBox

DenseBox: Unifying Landmark Localization with End to End Object Detection [Paper] : Baidu

model structure

VGG19 前12层conv，conv4-4后面添加up-sampling层，与conv3-4连接叉出两个branch，分别用于预测bbox的left top和bottom right点和是否包含物体的概率：$t_i = \{s, dx_t=x_i-x_t, dy_t=y_i-y_t, dx_b=x_i-x_b, dy_b=y_i-y_b\}_i$(第一个是confidence score，后面的是输出的坐标与真实bbox的距离)。

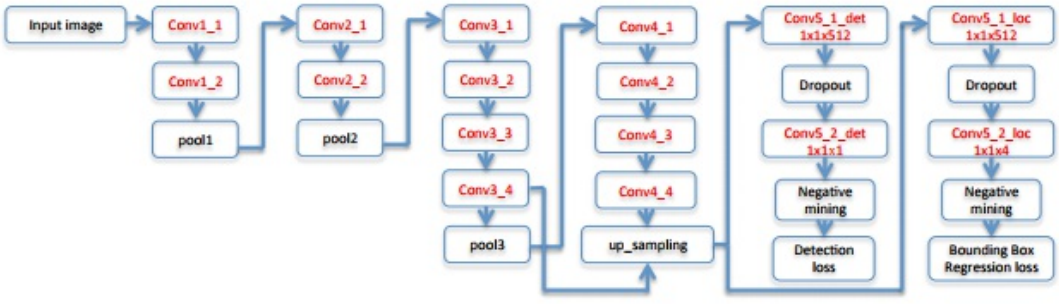


Figure 3: **Network architecture of DenseBox.** The rectangles with red names contain learnable parameters.

loss function

$$\mathcal{L}_{cls}(\hat{y}, y^*) = \|\hat{y} - y^*\|^2$$

$$\mathcal{L}_{loc}(\hat{d}, d^*) = \sum_{i \in \{tx,ty,bx,by\}} \left\|\hat{d}_i - d_i^*\right\|^2$$

（貌似这个公式是错误的）

$$M(\hat{t}_i) = \begin{cases} 0 & f^i_{ign} = 1 \text{ or } f^i_{sel} = 0 \\ 1 & \text{otherwise} \end{cases}$$

$$\mathcal{L}_{det}(\theta) = \sum_i \left(M(\hat{t}_i)\mathcal{L}_{cls}(\hat{y}_i, y^*_i) + \lambda_{loc}[y^*_i > 0]M(\hat{t}_i)\mathcal{L}_{loc}(\hat{d}_i, d^*_i) \right)$$

landmark Localization

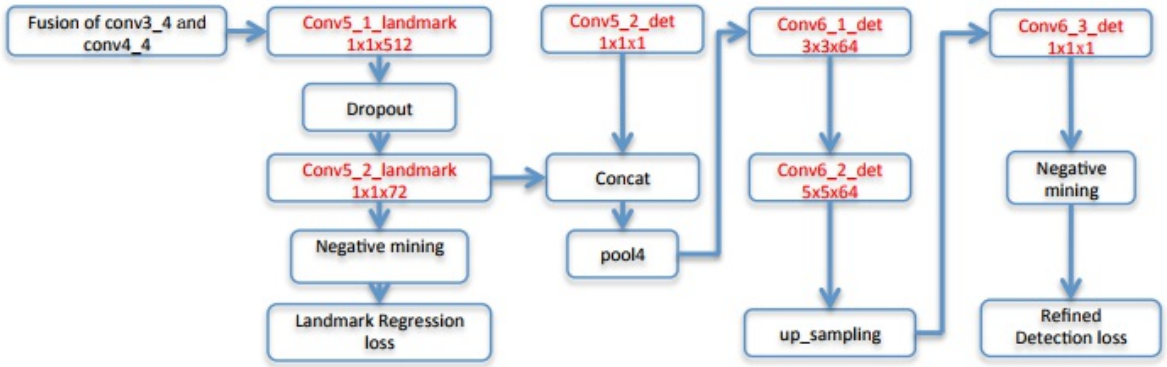


Figure 4: **Top:** The pipeline of DenseBox with landmark localization. **Bottom:** The network structure for landmark localization.

$$\mathcal{L}_{full}(\theta) = \lambda_{det}\mathcal{L}_{det}(\theta) + \lambda_{lm}\mathcal{L}_{lm}(\theta) + \mathcal{L}_{rf}(\theta)$$

Llm: L2 loss of predicted value and labels; Lrf与Lcls相同

---

7. SSD

SSD: Single Shot MultiBox Detector [Paper] [Code]

特点：用一套预设大小和长宽比的bbox来预测各类的score和box offset（类似于Faster RCNN的anchor box）；**结合不同分辨率的feature map用于处理不同大小的物体**；完全 **没有proposal generation** 和接下去的feature resampling，易于训练。速度快59fps on Titan X with mAP 74.3% on 300x 300 image in VOC2007

Model structure

以VGG16作为base，然后添加几层逐渐变小的conv层作为不同尺寸的feature map用于检测模型（**低层感受野比较小适合检测小物体，高层感受野比较大适合检测大物体**）。在各层m x n x p 大小的feature map用3 x 3 x p的kernel来生成类的score或者相对与预设的bbox的坐标offset。

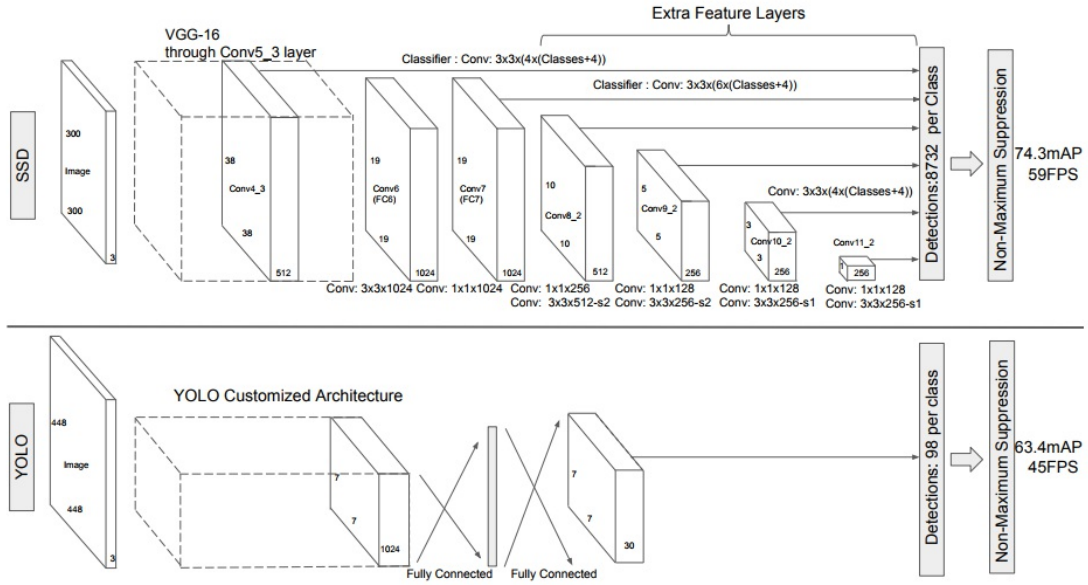
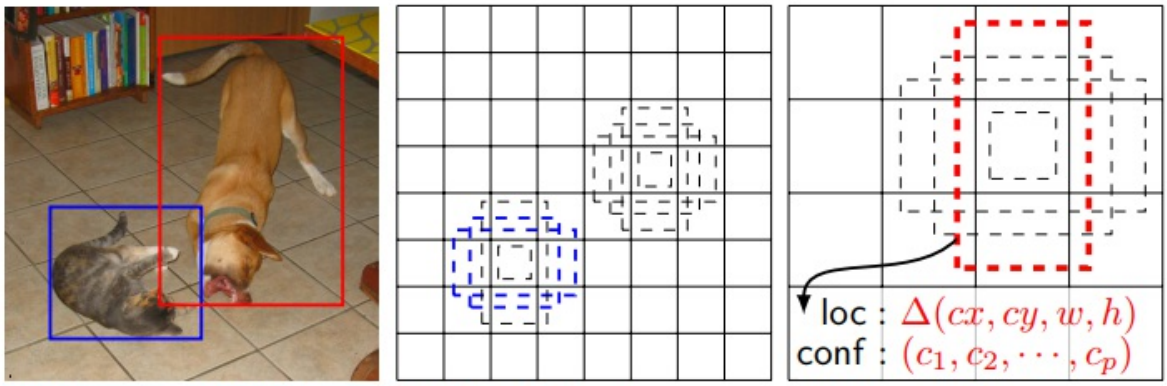


Fig. 2: A comparison between two single shot detection models: SSD and YOLO [5]. Our SSD model adds several feature layers to the end of a base network, which predict the offsets to default boxes of different scales and aspect ratios and their associated confidences. SSD with a 300×300 input size significantly outperforms its 448×448 YOLO counterpart in accuracy on VOC2007 `test` while also improving the speed.

bbox的默认大小和长宽比

对于feature map的每个位置有(c+4)*k个filter，其中c为class数，k为每个点预设的bbox数，4为box shape offset，所以每个feature map共有(c+4)kmn个输出。



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

Fig. 1: **SSD framework.** (a) SSD only needs an input image and ground truth boxes for each object during training. In a convolutional fashion, we evaluate a small set (e.g. 4) of default boxes of different aspect ratios at each location in several feature maps with different scales (e.g. 8×8 and 4×4 in (b) and (c)). For each default box, we predict both the shape offsets and the confidences for all object categories ($(c_1, c_2, \cdots, c_p)$). At training time, we first match these default boxes to the ground truth boxes. For example, we have matched two default boxes with the cat and one with the dog, which are treated as positives and the rest as negatives. The model loss is a weighted sum between localization loss (e.g. Smooth L1 [6]) and confidence loss (e.g. Softmax).

Model training

1). 与YOLO和faster RCNN类似的，SSD训练的时候需要首先将ground bbox投射到 **每层** feature map上。 首先需要确定哪些预设的box对ground truth的检测负责：根据预设box挑出IoU（jaccard overlap）最高的ground bbox，然后如果预设的box与其他ground bbox之间IoU大于0.5也进行配对，那么网络就可以对多个重叠的预设bbox都检车出高的分值而非只有IoU最高的那一个。

Loss：

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

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

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

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

$$L_{conf}(x, c) = -\sum_{i \in Pos}^{N} x_{ij}^p log(\hat{c}_i^p) - \sum_{i \in Neg} log(\hat{c}_i^0) \quad \text{where} \quad \hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)} \quad (3)$$

其中N是匹配上的预设bbox的个数。权重α设为1，而x_{ij}^k={1,0}为第i个预设box与第j个ground bbox在物体k上匹配。

预设的bbox不一定需要与每一层的实际感受野对应，**通过设计一系列预设的bbox使特定的feature map与特定大小的物体相对应。**
首先每一层feature map的预设bbox的大小计算如下：

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

其中，Smin=0.2， Smax=0.9， m指共有m层freature map，即最底层的大小为0.2，最高层的大小为0.9。

而长宽比为$a_r \in \{1, 2, 3, 1/2, 1/3\}$，bbox的宽$w_k^a$=Sk*sqrt($a_r$),bbox的高$h_k^a$=Sk/sqrt($a_r$)。 而对ar=1，加一个bbox的Sk'=sqrt(Sk*Sk+1)；而每个bbox的中心为((i+0.5)/|fk|, (j+0.5)/|fk|)，其中fk是第k个feature map的size。

通过这种设计，图1中4x4feature map有匹配上够的预设box，但是在8x8的feature map上就没有。

2). hard negative mining

3). data augmentation

- 使用全图
- 与物体box的IoU ∈ {0.1， 0.3， 0.5， 0.7， 0.9}的局部图
- 随机path

挑选的patch大小从原图的0.1-1，长宽比从0.5-2。然后resize到固定大小，0.5的几率随机水平翻转，以及其他一些photo-metric distortions。

---

8. DSSD

DSSD : Deconvolutional Single Shot Detector [Paper]

特点：SSD+resnet101 with deconvolutional layer (**获取高等级context**)

Model structure

只用resnet101替换vgg16，mAP反而从77.5%降到76.4%，但是进过调整之后的精度能显著提升。

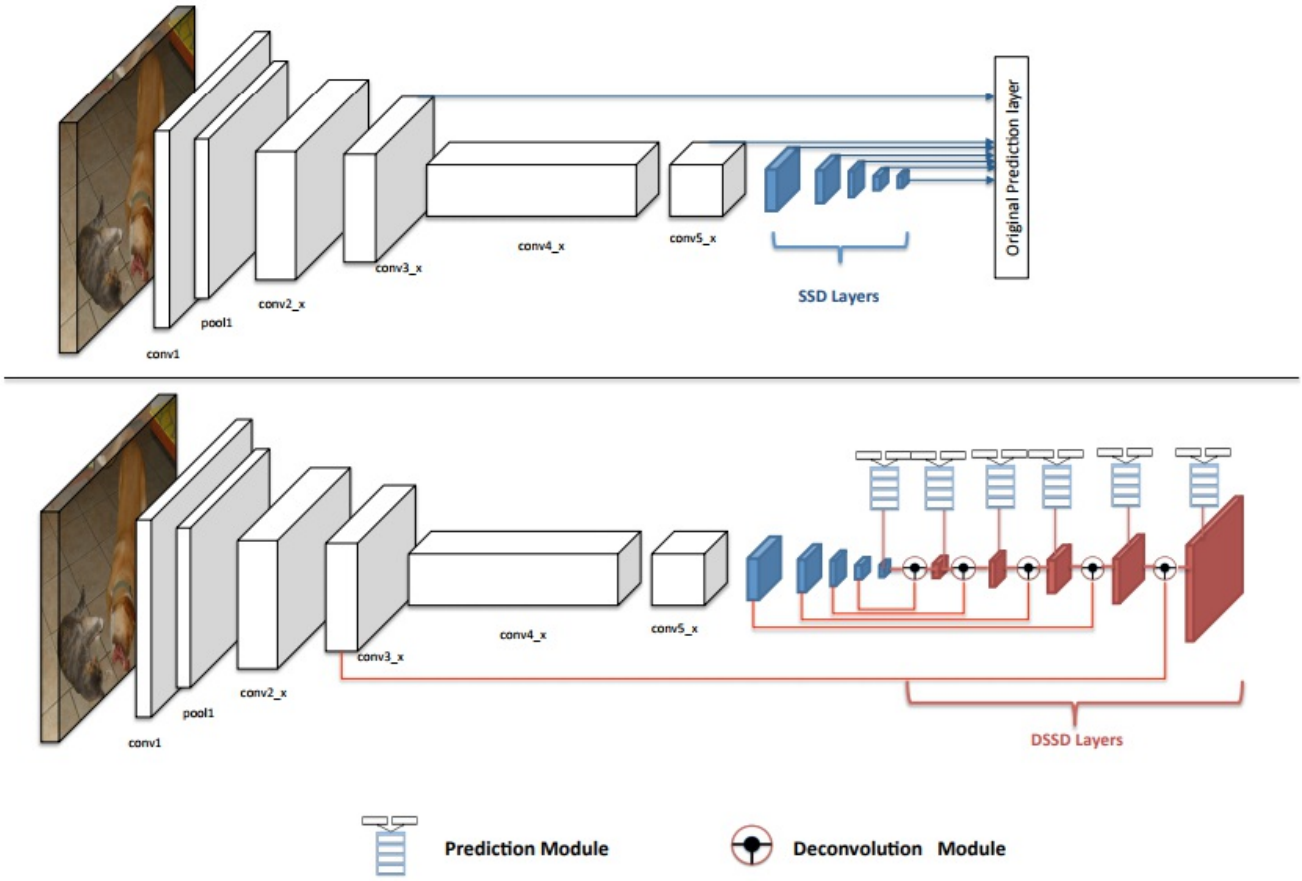


Figure 1: **Networks of SSD and DSSD on residual network.** The blue modules are the layers added in SSD framework, and we call them SSD Layers. In the bottom figure, the red layers are DSSD layers.

prediction module:在feature map之后使用resnet block后再进行预测，而非SSD中直接在conv层之后添加L2 normalization层进行预测，这样做的检测精度就明显超过VGG16了。

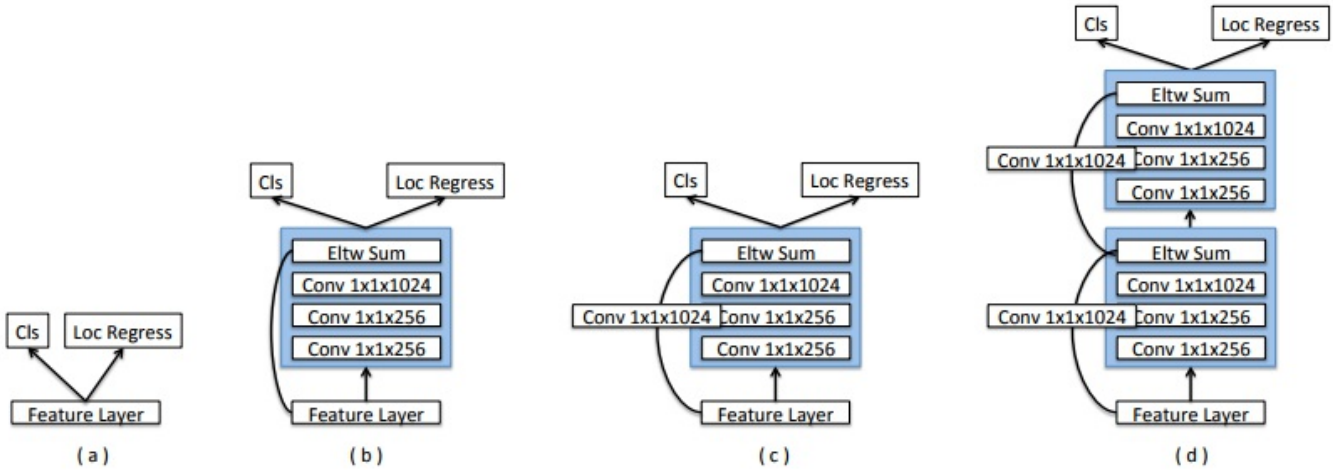


Figure 2: Variants of the prediction module

Deconvolutional SSD:为了获取高级的context，在SSD基础上添加了deconv层。（**缺点:训练预测时的处理速度会变慢，没有预训练模型可用**）

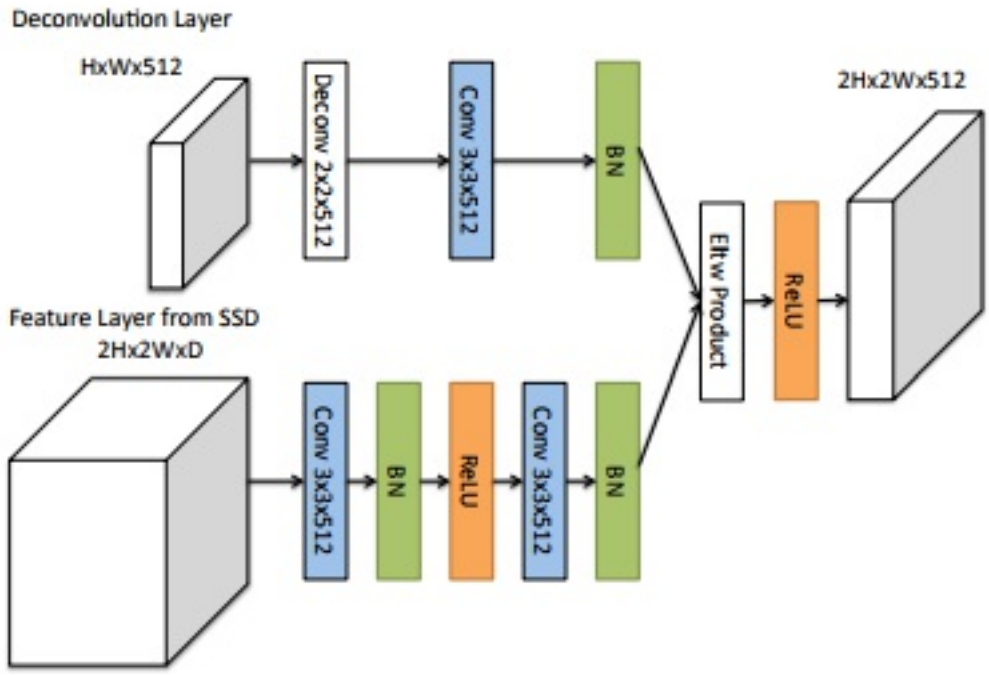


Figure 3: Deconvolution module

其中，elw product值element-wise product，用于连接原始的conv层和相应的deconv层。

Model training

Loss: joint localization loss (smooth L1) + confidence loss (softmax)。

由于没有类似RCNN的resampling过程，需要数据增强。

使用kmeans clustering来确定预设box的长宽比（**与YOLO2类似**），发现7个类最合适，添加了比例1.6（1/1.6）。

FPS

Method	network	mAP	With BN layers		BN layers removed		# Proposals	GPU	Input resolution
			FPS	batch size	FPS	batch size			
Faster R-CNN [24]	VGG16	73.2	7	1	-	-	6000	Titan X	$\sim 1000 \times 600$
Faster R-CNN [14]	Residual-101	76.4	2.4	1	-	-	300	K40	$\sim 1000 \times 600$
R-FCN [3]	Residual-101	80.5	9	1	-	-	300	Titan X	$\sim 1000 \times 600$
SSD300*[18]	VGG16	77.5	46	1	-	-	8732	Titan X	300×300
SSD512*[18]	VGG16	79.5	19	1	-	-	24564	Titan X	512×512
SSD321	Residual-101	77.1	11.2	1	16.4	1	17080	Titan X	321×321
SSD321	Residual-101	77.1	18.9	15	22.1	44	17080	Titan X	321×321
DSSD321	Residual-101	78.6	9.5	1	11.8	1	17080	Titan X	321×321
DSSD321	Residual-101	78.6	13.6	12	15.3	36	17080	Titan X	321×321
SSD513	Residual-101	80.6	6.8	1	8.0	1	43688	Titan X	513×513
SSD513	Residual-101	80.6	8.7	5	11.0	16	43688	Titan X	513×513
DSSD513	Residual-101	81.5	5.5	1	6.4	1	43688	Titan X	513×513
DSSD513	Residual-101	81.5	6.6	4	6.3	12	43688	Titan X	513×513

Table 7: **Comparison of Speed & Accuracy on PASCAL VOC2007 `test`.**

9. Inside-Outside Net (ION)

Inside-Outside Net: Detecting Objects in Context with Skip Pooling and Recurrent Neural Networks
[Paper]: Microsoft RGB

特点:预测的时候使用通过spatial RNN整合了contextual information（ION结构）；在ROI区域用skip pooling提取不同尺度的特征；将这两者连接作为网络的输入。

Figure 2. **Challenging detections** on COCO 2015 test-dev using our model trained on COCO "2014 train."

Model Structure

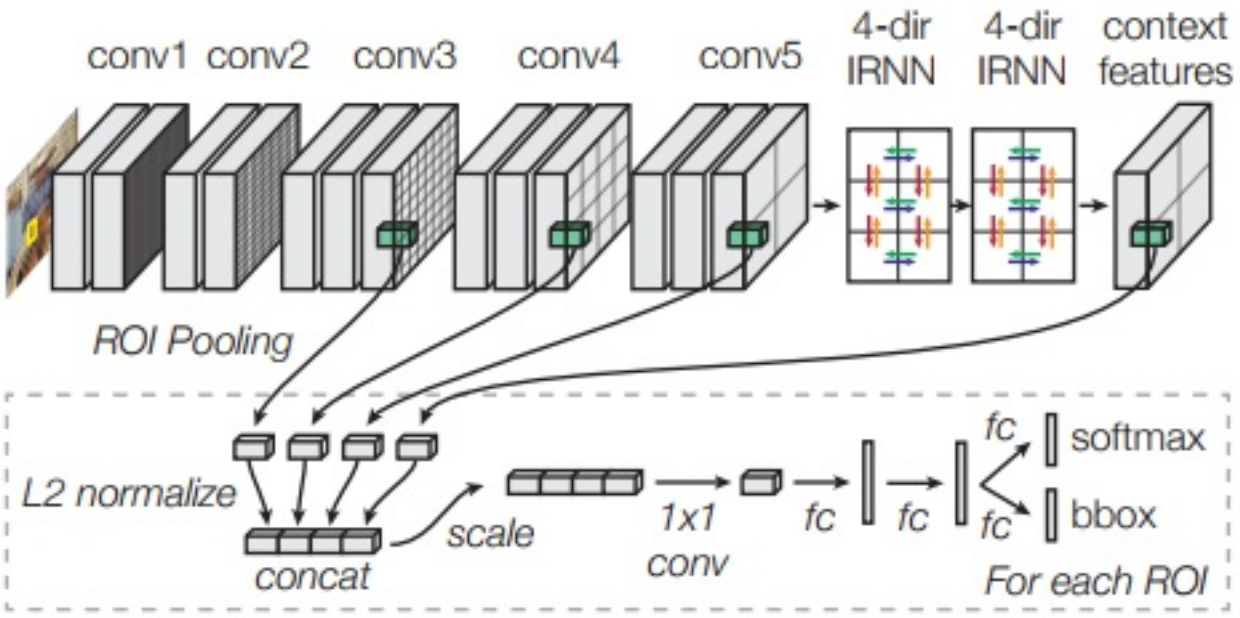


Figure 1. **Inside-Outside Net (ION).** In a single pass, we extract VGG16 [35] features and evaluate 2000 proposed regions of interest (ROI). For each proposal, we extract a fixed-size descriptor from several layers using ROI pooling [8]. Each descriptor is L2-normalized, concatenated, scaled, and dimension-reduced (1x1 convolution) to produce a fixed-length feature descriptor for each proposal of size 512x7x7. Two fully-connected (fc) layers process each descriptor and produce two outputs: a one-of-K class prediction ("softmax"), and an adjustment to the bounding box ("bbox").

2k ROI from raw image (RCNN) --> conv3,4,5+context feature --> L2 normalization， concatenate， re-scale, dimension reduced (1x1 conv) --> 512x7x7 (**由vgg16决定**)

此外，由于需要结合多层的feature，需要减少1x1conv层初始weight的大小，使用 **Xavier initialization**

Context feature with IRNNs

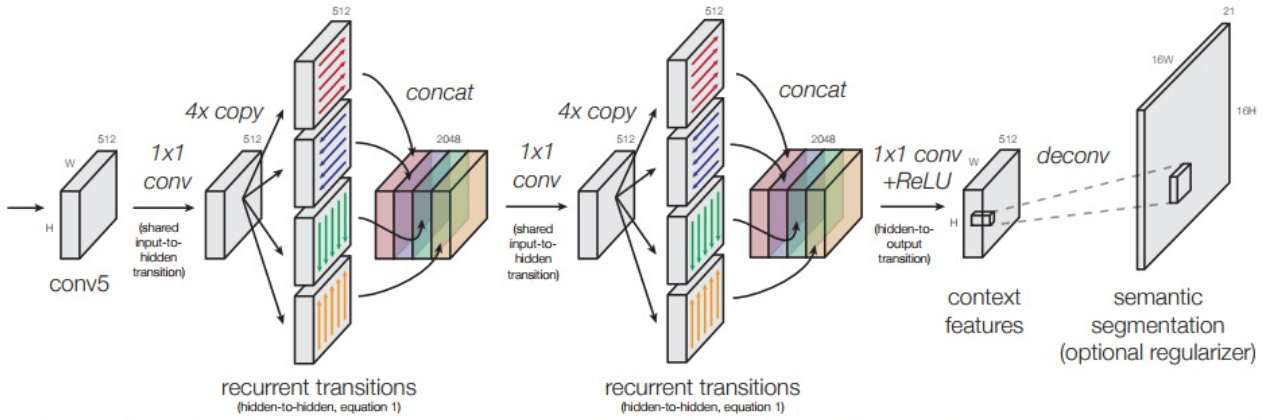


Figure 3. **Four-directional IRNN architecture.** We use "IRNN" units [21] which are RNNs with ReLU recurrent transitions, initialized to the identity. All transitions to/from the hidden state are computed with 1x1 convolutions, which allows us to compute the recurrence more efficiently (Eq. 1). When computing the context features, the spatial resolution remains the same throughout (same as conv5). The semantic segmentation regularizer has a 16x higher resolution; it is optional and gives a small improvement of around +1 mAP point.

RNN输入序列的的4个读取方向：上下左右。这里采用的RNN为 **ReLU RNN**（recurrent weight matrix初始化为 **identity matrix**，梯度可以完全通过BP传递。效果可以和LSMTM一样好，但是内存，计算速度可以提升很多）。（常用的RNN结构为GRU， LSTM， plain tanh recurrent nn）

使用 **1x1 conv** 作为input-to-hidden transition，因为可以被各个方向共享。bias也可以通过这个方式共享然后整合到1x1 conv层中。然后IRNN的输出以4个方向的隐含层依次连接。

IRNN从左到右的update如下，其他方向类似：

$$h_{i,j}^{\text{right}} \leftarrow \max\left(\mathbf{W}_{hh}^{\text{right}} h_{i,j-1}^{\text{right}} + h_{i,j}^{\text{right}}, 0\right).$$

而当将hiden transition matrix固定为单位阵的时候，上式就可以简化为：

$$h_{i,j}^{\text{right}} \leftarrow \max\left(h_{i,j-1}^{\text{right}} + h_{i,j}^{\text{right}}, 0\right).$$

在每个方向，平行计算所有独立的行和列而非每次计算一个RNN cell；使用semantic label来regularize IRNN输出，这个时候需要添加一层deconv层（32x32 kernel放大16倍）并crop；经过测试在连接各层输出的时候 **不需要dropout**；在训练RNN的时候bias也是不需要的。

第一个4向IRNN输出的feature map就汇总了每个cell周边的feature，随后的1x1 conv层将这些信息混合并降维。在第二个IRNN后，每个cell的输出与输入与输入相关，因此此时的context feature包含了 **global和local** 的信息，feature随位置而改变。

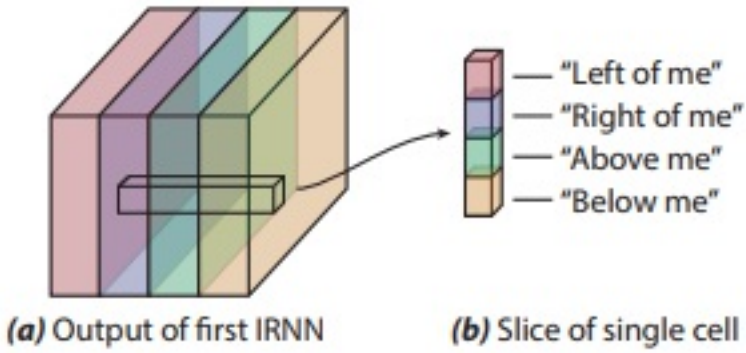


Figure 4. Interpretation of the first IRNN output. Each cell in the output summarizes the features to the left/right/top/bottom.

10. R-FCN

R-FCN: Object Detection via Region-based Fully Convolutional Networks [Paper] [Code] : Kaiming He

特点：fully convolutional, computation shared on entire image, position-sensitive score map

Table 1: Methodologies of *region-based* detectors using **ResNet-101** [9].

	R-CNN [7]	Faster R-CNN [19, 9]	R-FCN [ours]
depth of shared convolutional subnetwork	0	91	101
depth of RoI-wise subnetwork	101	10	**0**

Model structure

region proposal (RPN) + region classification。最后一层feature map对每个类+背景生成一个k^2 **position-sensitive** （top-left, ..., bottom right） score map，因此总共有$k^2(C+1)$个channel。RFCN最后一层是position-sensitive ROI pooling层，合并conv层的结果输出每个ROI的score，但是这里用的是selective pooling（与fast rcnn中不同），每一个kxk的bin都只是来自kxk层score map的其中一层。

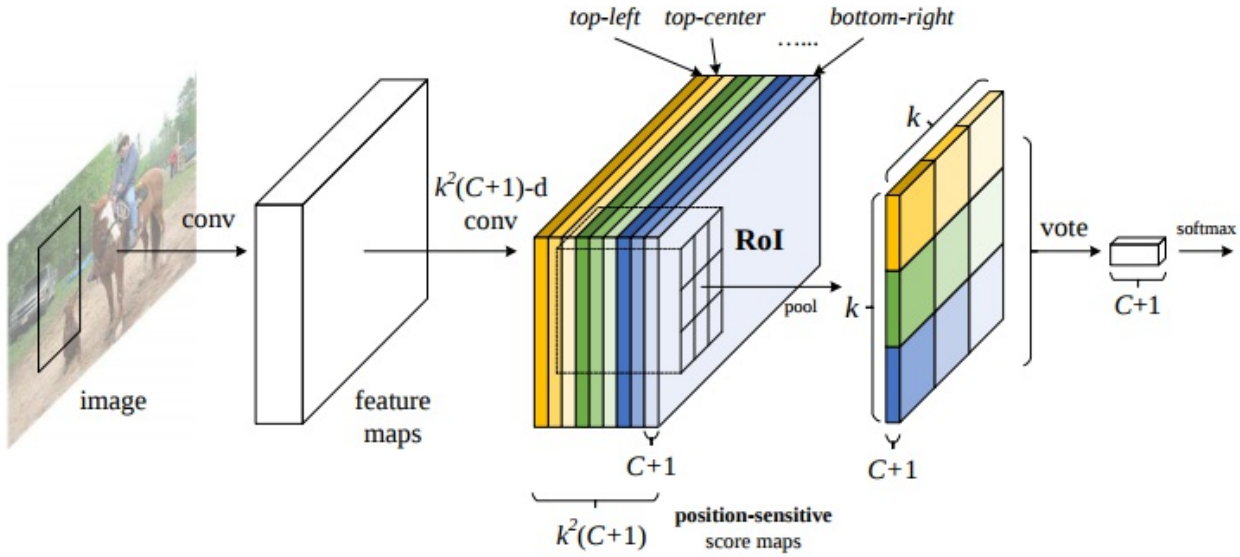


Figure 1: Key idea of **R-FCN** for object detection. In this illustration, there are $k \times k = 3 \times 3$ position-sensitive score maps generated by a fully convolutional network. For each of the $k \times k$ bins in an RoI, pooling is only performed on one of the k^2 maps (marked by different colors).

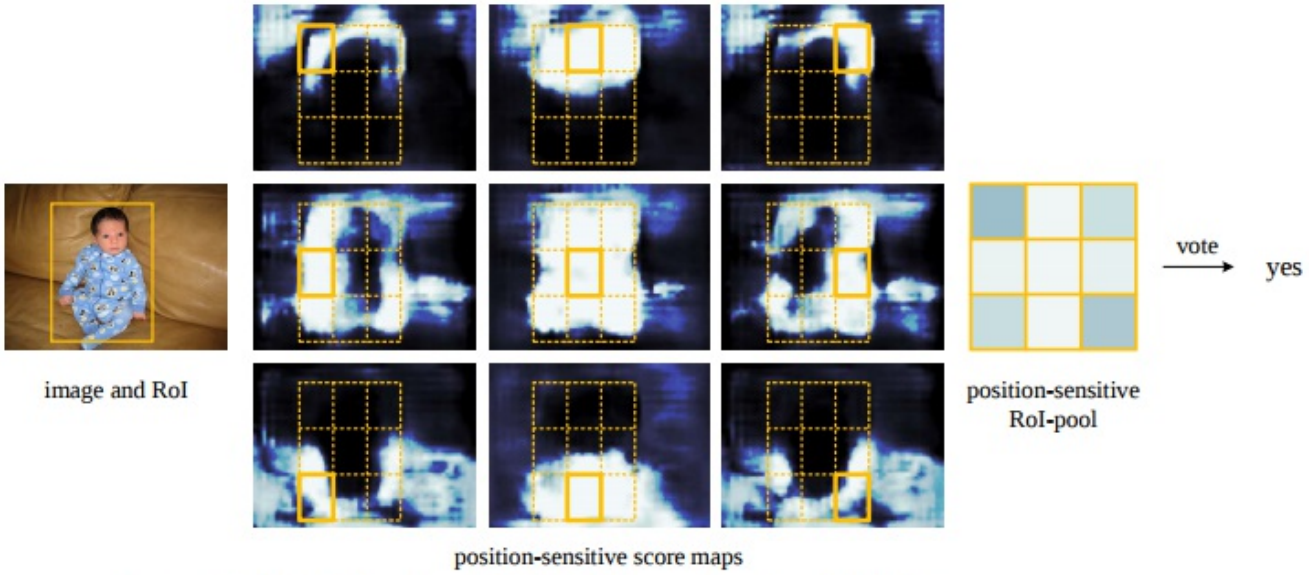


Figure 3: Visualization of R-FCN ($k \times k = 3 \times 3$) for the *person* category.

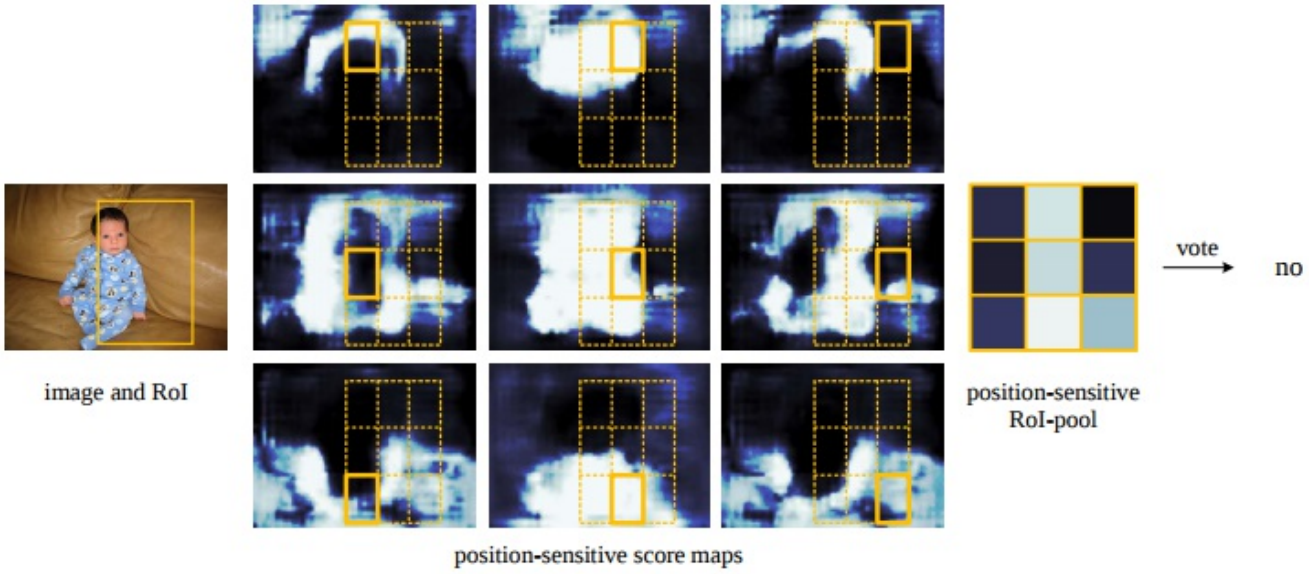


Figure 4: Visualization when an RoI does not correctly overlap the object.

Base Net: ResNet101 + 1x1 conv (2048d->1024d) + k^2(C+1)-channel conv层

position-sensitive score map/pooling
将ROI分成k x k个格子，每个格子w x h；然后对(i,j)格子的score map进行pooling（**position-sensitive ROI pooling**）。

$$r_c(i,j \mid \Theta) = \sum_{(x,y) \in \text{bin}(i,j)} z_{i,j,c}(x + x_0, y + y_0 \mid \Theta)/n.$$

其中r是第(i,j)格子的对于类型c的pooled response，z是k^2(C+1)个score map的其中一个，n是bin中包含的pixel数（所以是average pooling）

然后这个k^2 position-sensitive score对这个ROI使用average scroe进行投票，生成一个C+1维vector，然后计

算softmax（分类score）。

$$r_c(\Theta) = \sum_{i,j} r_c(i,j \mid \Theta).$$

$$s_c(\Theta) = e^{r_c(\Theta)} / \sum_{c'=0}^{C} e^{r_{c'}(\Theta)}$$

类似的，在上述$k^2(C+1)$d的conv层添加 **$4k^2$d** 的conv层用于bbox coordinate regression，输出向量为$4k^2$向量，然后通过average voting整合到4d向量（t=(tx,ty,tw,th)）。**注意，这里为了简化计算用的是class agnostic进行box回归，如果需要用class-specifi，那么添加的conv层为$4k^2C$。**（ROI层以后不需要学习，所以训练时候的计算量可以忽略）

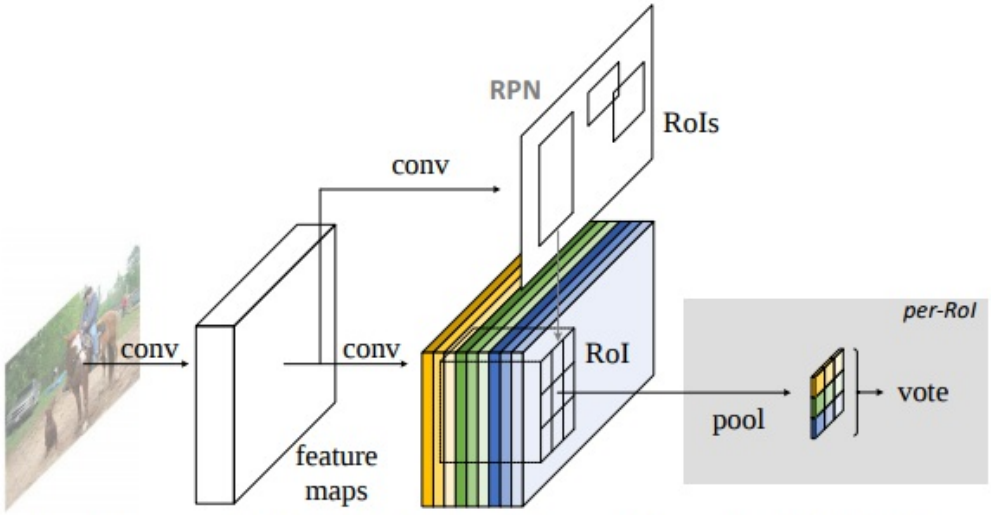


Figure 2: Overall architecture of R-FCN. A Region Proposal Network (RPN) [18] proposes candidate RoIs, which are then applied on the score maps. All learnable weight layers are convolutional and are computed on the entire image; the per-RoI computational cost is negligible.

training

Loss: IoU > 0.5为正例

$$L(s, t_{x,y,w,h}) = L_{cls}(s_{c^*}) + \lambda[c^* > 0]L_{reg}(t, t^*).$$

OHEM

input image: 600 (短边)

Algorithme à trous: improve mAP 2.6%

inference

300 ROI， NMS with IoU = 0.3