计算机视觉论文阅读集合

wanghao

2019 年 5 月 31 日

目录

1 图像分类

2 目标检测

2.1 Rich feature hierarchies for accurate object detection and semantic segmentation

2.1.1 模型框架

R-CNN 分为三个模块。

提出候选框。文章采用 selective search 方法，对一张图片计算出若干个边界框。

特征抽取。将边界框在原图中对应的区域都 resize 成 227*227 大小的，然后将提取出来的区域输入到 CNN 特征提取网络中，最终输出 4096 维的特征向量。

SVM 线性分类。SVMs 的输入为 4096 维的特征向量，输出为 N 维的分数向量。N 表示可识别物体的种类，分数表示图像是该类物体的可能性。该 SVMs 由 N 个二分类 SVM 构成，每个 SVM 都对应了一个类别。

虽然文章中宣称只有三个模块，但是实际上还有一个模块。

边界框回归。每类物体都有一组回归器用于修正预测框的位置。当候选框经过 SVMs 打分后，再通过对应类的回归器，修正候选框位置。

2.1.2 模型训练

模型采用分步的训练方法。

CNN 特征提取网络。

将已经训练好的用于图像分类的 CNN 网络的分类层修改为 N+1(N 表示物体种类，1 表示背景类)路，然后进行微调。与真实框有 >=0.5 IoU 的候选框作为这个框类别的正样本，其余的作为负样本。注意如果一个候选框包含了两个的真实框，则取最大 IoU 值的真实框作为改候选框的正样本。每次 SGD 迭代微调，都采样 32 个正样本，96 个负样本作为一个 mini-batch。

SVM 线性分类

对每类物体都训练一个 SVM 二分类器。对每一类的 SVM 分类器来说，将真实框作为正样本，将候选框与所有该类真实框的 IoU 都 $\leq \theta$ 的作为负样本。最优的 θ 值用搜索的方法找出来 (实验结果为 0.3)。

边界框回归

训练集由 $\{P^i, G^i, \phi(P^i)\}$ 组成,$P^i = (P^i_x, P^i_y, P^i_w, P^i_h), G^i = (G^i_x, G^i_y, G^i_w, G^i_h)$ 。G^i 是距离 P^i 最近的真实框，并且他们的 IoU$\geq$ 0.6。如果不存在这样的真实框，则该候选框会被丢弃。每一组训练器都由四个线性回归器组成，$d_*(P) = \{d_x(P), d_y(P), d_w(P), d_h(P)\}$，$d_*(P) = w_*^T \phi(P)$，$\phi(P)$ 表示 CNN 网络提取出来的特征。预测的 $\hat{G}_*$ 由以下方程构造出来:

$$\hat{G}^x_i = P^w_i d_x(P) + P^x_i \tag{1}$$

$$\hat{G}^y_i = P^h_i d_y(P) + P^y_i \tag{2}$$

$$\hat{G}^w_i = P^w_i \exp(d_w(P)) \tag{3}$$

$$\hat{G}^h_i = P^h_i \exp(dh(P)) \tag{4}$$

定义回归器 $d_*(P)$ 的目标为 t_*, 则：

$$t_x = (G_x - P_x)/P_w \tag{5}$$

$$t_y = (G_y - P_y)/P_h \tag{6}$$

$$t_w = \log(G_w/P_w) \tag{7}$$

$$t_h = \log(G_h/P_h) \tag{8}$$

目标函数为：

$$\mathbf{w}_* = \arg\min_{\hat{\mathbf{w}}_*} \sum_i^N (t_*^i - \hat{\mathbf{w}}_*^{\mathrm{T}} \phi_5(P^i)) + \lambda ||\hat{\mathbf{w}}_*|| \tag{9}$$

注意，CNN 和 SVM 的正负样本的定义是不同的。SVMs 是重新训练的，不需要微调。所以正负样本的定义不需要跟 CNN 训练时相同。作者发现用这个正负样本定义比用微调的正负样本定义的效果要好。作者认为正负样本的定义不同并不重要，关键是用于微调的数据是有限的。如果只把真实框作为正样本，可能会导致微调网络时过拟合。通过将那些 IoU 在 [0.5,1) 之间的候选框 (jittered examples) 作为正样本，可以把正样本的数据量扩大 30 倍。值得注意的是，把这些抖动的样本作为正样本可能不是一个最好的选择，因为没有用一个精确的位置去微调网络。

为什么在网络最后使用 SVMs 进行分类呢？作者发现使用 SVMs 的效果比使用原始的全连接层效果要好。作者猜测是正样本的设置策略导致了这个问题。作者认为如果有足够的检测训练数据，整个训练过程的正负样本就可以简单地统一为 SVMs 的样本设置策略，并且可以把 SVM 变为全连接层。这样就能简化整个 R-CNN 的训练过程。

2.1.3 模型测试

ss 生成 2000 个候选框，然后将框中的图像 reisze 成 227*227 的 RGB 图像，进入 CNN 网络提取出每个候选框对应的特征，然后将特征送入 SVMs 分类器，得到属于每个类的得分，判断出该候选框属于哪个类，最后再使用 NMS 算法，生成最后的边界框集合。

2.2 Fast R-CNN

Fast R-CNN 融合了 SPPnet 和 R-CNN 的特点，并做了一些改进。作者在 R-CNN 的基础上，根据 SPPnet 提出了 RoI Layer，然后合并了框的回归与分类计算，最终得到了 Fast R-CNN。与 SPPnet 和 R-CNN 相比具有以下优点：

1. 对比 SPPnet,R-CNN 有更高的检测质量。
2. 训练是单阶段，不需要分为很多步，使用了多任务损失。
3. 训练步骤可以更新所有的网络层参数，对比 SPPnet。
4. 不需要硬盘保存特征。

2.2.1 模型框架

Fast R-CNN 主要分为四个模块。

提出候选框。使用 ss 方法，与 R-CNN 中的提出候选框模块相同。

特征图计算。直接将原始图片输入 CNN 特征提取网络，获取整个图片对应的特征映射图。

候选框特征提取。将在原图上的候选框 (x, y, w, h) 映射到特征图 $(M \times N)$ 上去，然后截取框内的特征，池化后输出固定维度的特征, 将该层称为 RoI pooling layer。假设 (x, y, w, h) 为候选框, 我们需要计算出该候选框在 feature map 上的位置 $(x^{'}, y^{'}, w^{'}, h^{'})$。假设候选框 (x, y, w, h) 由左上角 (x_1, y_1) 和右下角 (x_2, y_2) 两个点构成，其中心点为 $(\bar{x}, \bar{y})$。我们只需要分别对 $(x_1, y_1), (x_2, y_2)$ 找到在 feature map 上能感受到他们的点 $(x_1^{'}, y_1^{'})$,$(x_2^{'}, y_2^{'})$。$(x_1^{'}, y_1^{'})$ 和 $(x_2^{'}, y_2^{'})$ 即可确定 $(x^{'}, y^{'}, w^{'}, h^{'})$。feature map 上的每个点都能映射到原始图上去，可看成该映射点为中心的窗口做卷积得到。由于原始图上的点 (x_i, y_i) 可能被 feature map 上的多个点 $(x_j^{'}, y_j^{'})$ 感受到，我们需要选择一个点作为它的感受点，并且选择的这个点的映射点离 (x_i, y_i) 最近。$(x_1^{'}$ 和 $y_1^{'})$ 和 $(x_2^{'}, y_2^{'})$ 都按照上述方法确定。

假设核大小为 $k \times k$, 步长为 s, 填充为 p,k 为奇数。

feature map 到原图映射点的计算方法。设第 $i+1$ 层上的点为 $(x^{'}, y^{'})$，其在第 i 层的映射点为 (x, y)，则:

$$x = x^{'} * s + \lfloor k/2 \rfloor - p \tag{10}$$

$$y = y^{'} * s + \lfloor k/2 \rfloor - p \tag{11}$$

原图到 feature map 感受点的计算方法。设第 i 层的点为 (x, y)，其在第 $i+1$ 层上的感受点为 $(x^{'}, y^{'})$，则:

$$x^{'} = (x + p - \lfloor k/2 \rfloor)/s \tag{12}$$

$$y^{'} = (y + p - \lfloor k/2 \rfloor)/s \tag{13}$$

感受点的计算与映射点的计算是互逆的，但是映射点一定能得到整数值，感受点可能计算出浮点数。而浮点坐标是在这里是不存在的，再对浮点坐标取整即可。

在Fast R-CNN中，作者强行令 $p = \lfloor k/2 \rfloor$，最终得到：

$$x^{'} = x/s \tag{14}$$

$$y^{'} = y/s \tag{15}$$

假设在 feature map 前有 n 个卷积层和池化层，每层的步长为 s_i，feature map 上的点为 $(x^{'}, y^{'})$ 则：

$$x^{'} = x/S = x/\prod_{i=1}^{n} s_i \tag{16}$$

$$y^{'} = y/S = x/\prod_{i=1}^{n} s_i \tag{17}$$

由于可能得到浮点数，作者向候选框的中心做了偏移，即得到：

$$x_1^{'} = \lfloor x_1/S \rfloor + 1 \tag{18}$$

$$y_1^{'} = \lfloor y_1/S \rfloor + 1 \tag{19}$$

$$x_2^{'} = \lfloor x_2/S \rfloor - 1 \tag{20}$$

$$y_2^{'} = \lfloor y_2/S \rfloor - 1 \tag{21}$$

由于 $k \geq 1$, 所以上述计算得到的点一定可以感受到要感受的点。窃以为用 $x_1^{'} = \lceil x/S \rceil, x_2^{'} = \lfloor x_2/S \rfloor$ 更好 (:。

找到在 feature map 上的特征框 $(x^{'}, y^{'}, w^{'}, h^{'})$ 后, 进行最大池化。对任意大小的特征框都生成固定大小 $(H \times W)$ 的特征向量 (在 VGG16 中，$H = W = 7$)。

分类与边界框回归。该模块对应了两个任务，包括框的分类和回归。固定维度的特征经过一系列全连接层后分为两个分支，一个分支经过 softmax 输出每个类别 (加上背景) 的概率，一个分支对每一类输出 4 个回归值 (t_x, t_y, t_h, t_w)，t_* 的定义与 R-CNN 中是相同的, 利用候选框与真实框计算出 t_*。

2.2.2 模型训练

Fast R-CNN 将 R-CNN 的三个训练部分合成了一个整体。可以更新所有层的参数 (R-CNN 和 SPP-net 无法根据 SVM 的 loss, 更新卷积层的参数)，可以实现端到端的训练。不需要将特征缓存在硬盘中。

数据生成。每个 mini-batch, 作者随机取出 2 张图片，然后在 2 张图片上共采样出 128 个候选框，每张图片采样出 64 个候选框。将 $IoU \geq 0.5$ 的作为正样本，将 IoU 在 [0.1) 的作为负样本，每个 mini-batch 生成的正负样本比例为 1:3。训练时，图片以 0.5 的概率随机水平翻转。

损失函数。Fast R-CNN 使用了多任务损失函数 $L(p, u, t^u, v)$：

$$L(p, u, t^u, v) = L_{cls}(p, u) + \lambda[u \geq 1] L_{loc}(t^u, v) \tag{22}$$

$$L_{cls}(p, u) = -\log p_u \tag{23}$$

$$L_{loc}(t^u, v) = \sum_{i \in x,y,w,h} \text{smooth}_{L1}(t_i^u - v_i) \tag{24}$$

$$\text{smooth}_{L1}(x) = \begin{cases} 0.5x^2 \\ |x| - 0.5 \end{cases} \tag{25}$$

p 是 softmax 输出的 $K + 1$ 维类别概率矩阵,u 表示该框的真实类别,v 表示框的真实偏移,t^u 表示框的预测偏移 (定义与2.1.2中相同)。损失函数由类别损失和位置损失两部分构成，当真实类别为前景 $(u \geq 1)$ 时，位置损失有效，当真实类别为背景 $(u = 0)$ 时，位置损失无效。

注意。1. Fast R-CNN 中的三个部分是统一训练的，正负样本的策略是相同的，这与 R-CNN 中有很大不同。2. Fast R-CNN 中类别分类使用的是 softmax, 而 R-CNN 中使用的是 SVM。3.Fast R-CNN 采用的是联合训练方式，而 R-CNN 中 CNN 网络微调、框的分类和回归的训练是分开的。作者通过实验证明联合训练的方式效果比分步训练好，softmax 的效果始终比 SVM 好一点。使用更严格负样本策略的 SVM 在 S 和 M 的模式的 Fast-RCNN 的效果好，而在 L 模式下，联合训练的效果比使用更严格负样本策略的 SVM 好。

2.2.3 模型测试

ss 生成大概 2000 个候选框，经过整个网络输出每个框的类别和位置，再使用 NMS 输出结果。

2.3 Faster R-CNN-Towards Real-Time Object

作者在 Fast R-CNN 的基础上，用 RPN 网络代替了 ss 算法。

2.3.1 模型框架

Fast R-CNN 网络。除去 ss 算法部分的 Fast R-CNN 结构。

RPN 网络。RPN 网络主要用于生成候选框，与 ss 的作用是相同相同的。网络结构组成：feature map 以前的网络结构部分 $+n \times n$ 的卷积层 + 两个分离的 1×1 卷积层 (分别用于 anchors 的分类和回

归)。RPN 网络实际上以 feature map 层上的每个点为中心，将这些点映射到原图上去，每个点都有大小相同、形式相同的多个框。将这些框作为默认框，进行分类和回归，生成效果更好的候选框。

Anchors。所谓的 anchors 实际上就是在原图上画的一些默认框，这些默认框可以看成滑动窗口在一些点上移动构成的，而这些点是从 feature map 上映射下来的。假设 feature map 上每个点对应了 k 个框，文章中默认使用了 3 种大小，3 种长宽比，则每个点产生 $k = 9$ 个点。假设 feature map 的大小为 $W * H$，则共产生 WHk 个 anchors。

Fast R-CNN 和 RPN 共享了 feature map 层之前的网络结构。

2.3.2 模型训练

相较 Fast R-CNN，Faster R-CNN 增加了一个 RPN 网络的训练。

4 步联合训练方法。1. 训练 RPN 网络，用预训练好的模型初始化参数。2. 训练一个分开的 Fast R-CNN 网络，使用 RPN 网络生成的框，用预训练好模型初始化参数。此时，两个网络没有共享参数。3. 再次训练 RPN 网络，用第 2 步训练好的 Fast R-CNN 的参数初始化 RPN 共享部分的网络参数，然后固定共享部分的参数，只微调 RPN 独有的网络部分。此时，RPN 和 Fast R-CNN 共享了 feature map 前网络的参数。4. 再次训练 Fast R-CNN 网络，固定共享部分的参数，微调 Fast R-CNN 独有的网络部分。

RPN 网络训练。RPN 的训练是端到端的。每个 mini-batch 的训练数据都是从一张图片生成的。一张图片的所有 anchorts 包含了大量的负样本和少量的负样本，因此在训练的时候，我们不能使用所有的 anchors，否则会导致损失函数偏向负样本。每次 mini-batch 采样 256 个 anchors，正负样本比例设置为 $1:1$, 若正样本不足，则用负样本填充。正样本规则：1. 和一个真实框具有最高的 IoU，2. 和任意一个真实框的 IoU≥ 0.7。负样本规则：和所有框的 IoU 都 ≤ 0.3。用 $(0, 0.01)$ 的高斯分布初始化 RPN 独有的层，用预训练好的模型初始化其他的。损失函数如下：

$$L(\{p_i\}, \{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*) \tag{26}$$

$L_{cls}(p_i, p_i^*)$ 和 $L_{reg}(t_i, t_i^*)$ 跟2.2.2中是类似的，分别代表对数损失和回归损失，λ 设为 10。

Fast R-CNN 网络训练。与 2.2.2是类似的。不同的地方在于 RPN 网络可能产生的候选框比 ss 算法的多，而 RPN 中的候选框相互之间可能有很高的 IoU。训练时，为了减少冗余计算，我们对 RPN 的候选框做了一个 NMS(阈值设为 0.7)，最终大概 2000 个候选框。RPN 网络可能生成了超出图片边界的框，对于这种框直接忽略。

2.3.3 模型测试

与 Fast R-CNN 类似。注意 RPN 生成的候选框可能超出了边界，对于这种框需要通过去除超出边界的部分，修正框的位置。

2.4 You Only Look Once-Unified Real-Time Object Dectection

2.4.1 模型框架

2.4.2 模型训练

2.4.3 模型测试

2.5 SSD:Single Shot MultiBox Detector

2.5.1 模型框架

2.5.2 模型训练

2.5.3 模型测试

3 目标追踪

目前在目标追踪领域识别效果最好的是 Siamese Networks 系列的模型，并且它的运行速度远远超过了人眼所需的帧率。

3.1 Fully-Convolutional Siamese Networks

本文提出了一个新的全卷积孪生网络，在 ILSVRC15 上达到了最优。

3.1.1 模型框架

该模型由一个 Siamese network 和一个 cross-correlation 层组成。最终输出一个 score map，在 score map 中取最大的值对应的位置，即为目标位置。

Siamese network。实际上就是一个全卷积神经网络，用于提取图像的特征。该网络输入两张图像，同时经过同一个卷积神经网络，输出两个特征图，将该网络抽象为函数 φ。两张图像一个作为 exemplar image z, 一个作为 candidate image X，最终生成 $\varphi(z)$ 和 $\varphi(X)$。

Cross-correlation。实质上就是一个卷积层，将 z 视为卷积核，步长为 1，在 X 上做卷积，最终生成一个 score map，其上的每个点表示是该物体的得分值。

模型框架如图 1 所示。

3.1.2 模型训练

数据生成。每次从一段视频中抽取出两帧图像，两张图像均包含目标物体。z 的大小是 127×127，X 的大小是 255×255。z 和 X 以相同的方式生成，假设边界框大小为 (w, h), 以目标物体为中心扩增上下文 (就是原图像) 边距 $p = (w + h)/4$，然后再对 $(w + 2p, h + 2p)$ 用各通道均值做一个 padding，使其达到 z 或 X 需要的大小。对 score map 上的每个点都要生成一个 label, 当该点与目标中心的距离不大于 R 时，设为正值 1, 否则为 -1。

损失函数。对最终生成的 score map 计算损失值。

$$L(y, v) = \frac{1}{|\mathcal{D}|} \sum_{u \in \mathcal{D}} l(y[u], v[u]) \tag{27}$$

y 表示真实值，v 表示计算出来的值,u 表示 score map 上的点，$\mathcal{D}$ 表示 score map，L 表示损失。

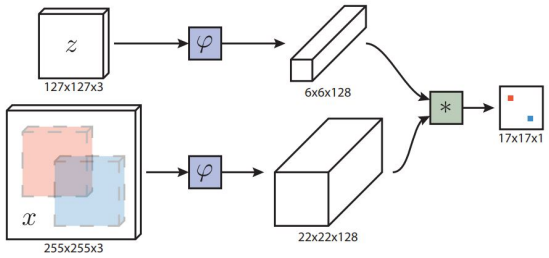

Fig. 1. Fully-convolutional Siamese architecture. Our architecture is fully-convolutional with respect to the search image x. The output is a scalar-valued score map whose dimension depends on the size of the search image. This enables the similarity function to be computed for all translated sub-windows within the search image in one evaluation. In this example, the red and blue pixels in the score map contain the similarities for the corresponding sub-windows. Best viewed in colour.

图 1: siamFC 框架

单个点的损失函数：

$$l(y, v) = \log(1 + \exp(-yv)) \tag{28}$$

3.1.3 模型测试

测试时输入一段视频和第一帧中目标的位置。用上述的方法抽取出第一帧的 z, 只算一次 $\varphi(z)$。每次的搜索区域都是以上一帧目标中心为中心，大小为上一帧目标大小的四倍 (实际上就固定是 255×255 了，因为能检测到的目标大小只能是 127×127)。最后生成的 score map 大小始终为 17×17，然后对这个 socre map 做一个三线性插值，生成 272×272 大小的 score map，然后再取最大值，至于为什么要这么干，当然是作者测出来这么干效果好了 (:。此外，作者假设在相邻的帧中，物体的位置，形状，大小等都是相似的，对于较大的变化都有一个惩罚，为了应对物体的尺度多样性，作者用五种尺度去搜索，。

3.2 High Performance Visual Tracking with Siamese Region Proposal Network

该论文实际上是 siamFC 模型和 RPN 网络的合体版本。

3.2.1 模型框架

该模型由一个孪生神经网络和 RPN 网络组成，输出每个默认框的位置修正值和正负概率。

Siamese network。一个全连接神经网络，用于提取图像特征，与 3.1中类似。

RPN。与2.2.2中的 RPN 网络类似，此时将 $\varphi(X)$ 作为输入的 feature map，然后分为两路一路用于分类，一路用于回归。$\varphi(z)$ 作为卷积核，同样分为两路一路用于分类，一路用于回归。在分类和回归两个分支中，$\varphi(X)$ 和 $\varphi(z)$ 都先经过一个卷积层，然后两者再做互相关操作 (互相关此时为卷积操作与3.1相同)。

模型框架如图 2 所示。

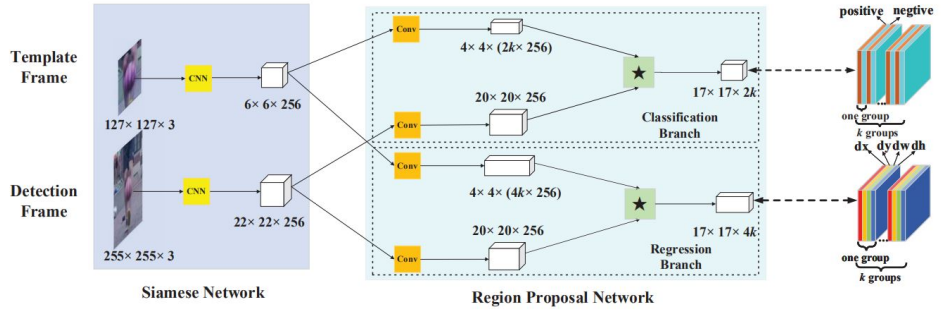

Figure 2: Main framework of Siamese-RPN: left side is Siamese subnetwork for feature extraction. Region proposal subnetwork lies in the middle, which has two branches, one for classification and the other for regression. Pair-wise correlation is adopted to obtain the output of two branches. Details of these two output feature maps are in the right side. In classification branch, the output feature map has $2k$ channels which corresponding to foreground and background of k anchors. In regression branch, the output feature map has $4k$ channels which corresponding to four coordinates used for proposal refinement of k anchors. In the figure, $\star$ denotes correlation operator.

图 2: SiamRPN 框架

3.2.2 模型训练

3.2.3 模型测试

3.3 Distractor-aware Siamese Networks for Visual

3.3.1 模型框架

3.3.2 模型训练

3.3.3 模型测试

4 实例分割

4.1 Mask R-CNN

5 对抗攻击

6 其他