



尺度不变的全卷积神经网络：高效的人手检测

Scale Invariant Fully Convolutional Network: Detecting Hands Efficiently

刘丹¹, 独大为², 张立波^{3,*}, 罗铁坚¹, 武延军³, 黄飞跃⁴, Siwei Lyu²

¹中国科学院大学, 中国 ²纽约州立大学奥尔巴尼分校, 美国

³智能软件研究中心, 中国科学院软件研究所, 中国 ⁴腾讯YouTu实验室, 中国

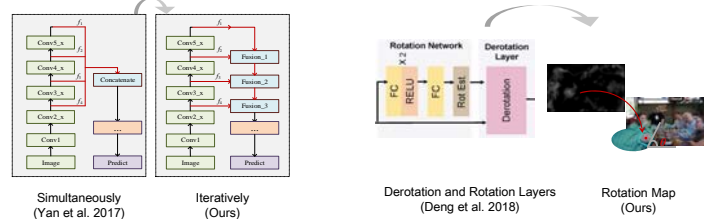
会议: AAI 2019
通讯作者: 张立波
联系方式:
18655882017
libo@iscas.ac.cn

问题: 高效的人手检测
应用: 虚拟现实、人机交互、驾驶监控等

动机

	代表性方法	不足
传统方法: 人工提取特征 + 分类器	皮肤特征 (Dardas and Georganas 2011) 手的形状和背景信息 (Mittal, Zisserman, and Torr 2011) 方向梯度直方图 (HOG) (Betancourt et al. 2015)	<ul style="list-style-type: none"> 特征提取的时间和精力开销大 非端到端优化过程 特征的局限性
基于深度学习的 方法: CNNs	候选区域 + CNN (Bambach et al. 2015) FCN + 分类 (Le et al. 2017) 旋转 & 解旋转 (Deng et al. 2018)	<ul style="list-style-type: none"> 小的手部区域漏检率大 网络结构复杂, 训练和测试时间长, 不能达到实时检测

贡献



- 提出尺度不变的全卷积神经网络, 补充加权特征融合模块 (CWF block) 可以学习每个尺度特征与其他尺度的特征, 并以一种迭代的方式融合多个尺度的特征进行最后的预测
- 提出多尺度损失函数, 对网络的中间层也加入了监督, 加快了模型收敛的速度
- 与当前最好的方法相比, 在保证精度的同时检测速度更加快

尺度不变的全卷积神经网络

1. 特征提取

- 取VGG16网络pooling-2到pooling-5的特征图; ResNet50网络conv2_1, conv3_1, conv4_1, conv5_1的特征图
- 特征大小分别是输入图像的 $(\frac{1}{4})^2$, $(\frac{1}{8})^2$, $(\frac{1}{16})^2$, $(\frac{1}{32})^2$

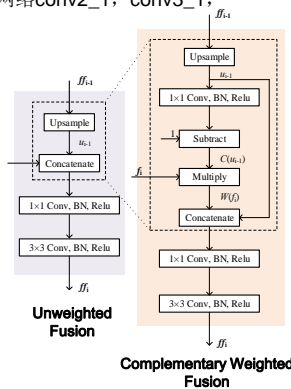
2. 特征融合

- 在unpooling层, 上一层 (较高层次) 特征图上采样到当前特征图相同大小, 记为 u_{s-1}
- (CWF) 对当前特征图 f_s 进行加权:

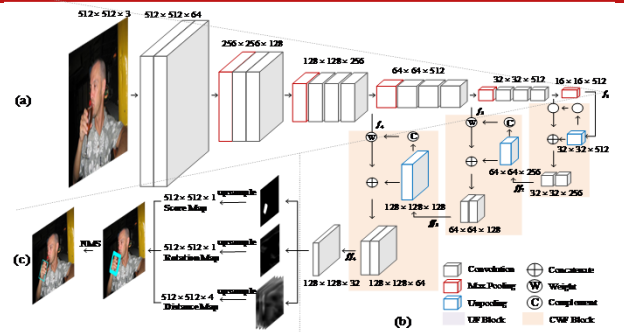
$$\begin{cases} W(f_s) = f_s * C(u_{s-1}), \\ C(u_{s-1}) = 1 - Conv_{1 \times 1}(u_{s-1}). \end{cases}$$
- 将上面得到的两层的特征图在通道维度上进行连接
 - 经过 1×1 卷积减少输出通道数
 - 经过 3×3 卷积融合不同层次的特征
- 融合后的特征图作为下一个融合模块 (UF或CWF) 的基本特征图进行下一步的迭代融合

3. 输出

- 输出包括三部分: 得分图, 角度图和距离图, 分别记录预测手型区域的置信度、旋转角度和几何信息。



- s : the current scale
- f_s : the current scale feature maps
- $W(f_s)$: the weighted feature maps
- $C(u_{s-1})$: the complementary feature maps
- $*$: element-wise multiplication



以VGG16为骨干网络的SIFCN网络结构, 主要包含三部分: (a) 特征提取部分, (b) 特征融合部分, 和 (c) 输出部分。

4. 多尺度损失函数

$$L = \sum_{s \in S} w_s (\alpha L_s + \beta L_r + L_d), \quad S = \{1, 2, 3, 4\}$$

$$\begin{cases} L_s = \frac{2 \sum_i^N p_i \theta_i}{\sum_i^N p_i^2 + \sum_i^N \theta_i^2} \\ L_r = 1 - \cos(\theta - \theta_0) \\ L_d = -\log \frac{\bar{X} \cap X}{\bar{X} \cup X} \end{cases}$$

多尺度损失函数增加了对网络中间层的监督, 加快了模型收敛的速度

实验

> VIVA 人手数据集 (Das, OhnBar, and Trivedi 2015)

- Train: 5,500 images Test: 5,500 images
- 真实的驾驶场景
- 标注为轴对齐的边界框

Table 1: Results on VIVA Dataset.

Methods	Level-1(AP/AR)/%	Level-2(AP/AR)/%	Speed/fps	Environment
MS-RFCN (Le et al. 2017)	95.19/4.5	86.0/8.4	4.65	6 cores@3.5GHz, 32GB RAM, Titan X GPU
MS-RFCN (Le et al. 2016)	94.29/1.1	86.9/7.3	4.65	6 cores@3.5GHz, 32GB RAM, Titan X GPU
Multi-scale fast RCNN (Yan et al. 2017)	92.8/8.2	84.7/6.5	3.33	6 cores@3.5GHz, 64GB RAM, Titan X GPU
FRCNN (Zhou, Pillai, and Yalla 2016)	90.7/5.9	86.5/5.3	-	-
YOLO (Redmon et al. 2016)	76.4/46.0	69.5/39.1	35.00	6 cores@3.5GHz, 16GB RAM, Titan X GPU
ACF-Depth4 (Das, Ohn-Bar, and Trivedi 2015)	70.1/53.8	60.1/40.4	-	-
Ours (VGG16+UF)	88.9/82.8	72.6/56.7	13.88	-
Ours (VGG16+UF+Multi-Scale Loss)	92.9/88.3	80.9/62.7	13.16	-
Ours (VGG16+CWF+Multi-Scale Loss)	92.3/89.1	83.6/68.8	13.10	-
Ours (ResNet50+UF)	93.7/89.9	83.6/73.6	20.40	4 cores@4.0GHz, 32GB RAM, GeForce GTX 1080
Ours (ResNet50+UF+Multi-Scale Loss)	94.0/90.1	85.7/74.0	20.00	-
Ours (ResNet50+CWF+Multi-Scale Loss)	94.6/92.1	86.3/75.8	19.68	-

> Oxford 人手数据集 (Mittal, Zisserman, and Torr 2011)

- Train: 2,250 images Test: 436 images
- 多种场景
- 标注为带有旋转角度的边界框

Table 2: Results on Oxford Dataset.

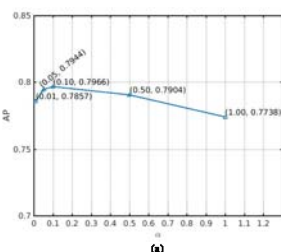
Methods	AP/%
MS-RFCN (Le et al. 2017)	75.1
Multiple proposals (Mittal, Zisserman, and Torr 2011)	48.2
Multi-scale CNN (Yan et al. 2017)	58.4
Ours (VGG16+UF)	68.7
Ours (VGG16+UF+Multi-Scale Loss)	77.8
Ours (VGG16+CWF+Multi-Scale Loss)	78.0
Ours (ResNet50+UF)	78.2
Ours (ResNet50+UF+Multi-Scale Loss)	78.6
Ours (ResNet50+CWF+Multi-Scale Loss)	80.4



结果对比: SIFCN (青色, 我们的) 和 Multi-scale fast RCNN (红色, Yan et al. 2017). (a) VIVA 数据集 (b) Oxford 数据集

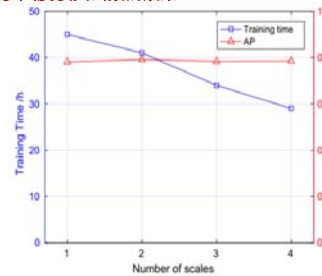
消融研究

得分图和角度图的作用



在Oxford 人手数据集上平均准确率 (average precision) 随 α 和 β 的变化趋势. (a) 平均准确率 vs. α ($\beta = 20$). (b) 平均准确率 vs. β ($\alpha = 0.01$).

多尺度损失函数的效果



Training time and AP score vs. different numbers of scales for the Oxford dataset.

补充加权特征融合 (CWF) 模块的作用

- 无论使用VGG16还是ResNet50作为骨干网络, 使用补充加权特征融合模块 (CWF) 的性能都优于非加权模块 (UF)
- 使用补充加权特征融合模块 (CWF) 相较于非加权模块 (UF) 产生的假负例更少, 更好的利用了不同尺度的特征。

Code here

