CBR 论文的总结：

本文提出的思路也是先通过 proposal 部分得到正样本视频序列段，再通过 detection 部分判断每个 proposal 的具体类别。两部分的 boundary 均做了回归处理。文章重点是提出了 CBR（Cascaded Boundary Regression）的回归方法；non-parameterized regression offsets，将这种回归方式与类似 faster rcnn 的 bounding box 的回归方式做了比较；提出了 frame-level 和 unit-level 的概念。提出的几个思路通过实验对比得到了较好的结果。

Temporal Coordinate Regression: 在做空间域边界回归时，采用了参数化坐标偏移，把中心点和长度两个量的偏移作为计算量。文中提出了非参数化坐标偏移，用 boundary 的首尾两个坐标的偏移作为计算量，这两个坐标可以是 unit-level 下的也可以是 frame-level 下的。连续的一段 frame 构成 unit，连续的 unit 构成 clip。

Cascaded Boundary Regression: 在 proposal 和 detection 两个阶段都是将得到的范围内的特征向量通过多层感知器（MLP）进行回归处理，并将再次输出的 boundary 通过相同参数 MLP 进行重复的回归。实验结果表明在不同的基础网络上通过 2-3 次 MLP 可以取得最好的结果。

实验:文中用 C3D 和 two-stream 两个网络结构来提取特征并进行比较，C3D 取 FC6 层的 4096 维特征当做 proposal 模块的输入，two-stream 中取每个 unit 的中间帧提取 spatial 特征，取中间连续六帧提取 motion 特征，最后通过 global pool 层将两个特征 concatenate 为 4096 维特征作为 proposal 模块输入。在 feature 上通过相互有特定交集的一系列滑框取特定段向量进行回归，得到了在 unit-level 下进行回归比 frame-level 更有效的结论，非参数化坐标偏移在 unit-level 下得到的结果比参数化更好，由此推测出在时间域做尺寸上的比较可能不合理。同时 two-stream 的结果比 C3D 结果在不同条件下得到的 mAP 高 5-7 个点。

THUMOS14：从 UCF101 中选取了二十个类作为分类，如下图所示。原视频长度从 10s 到将近 30min，所给的 ground truth 以 0.1 秒为单位进行标注，FPS 统一在 30，视频尺寸统一为 320x240。20 个类均为运动项目，且强调人的行为而不强调与人交互的物，例如 GolfSwing 不一定打击高尔夫球，可能是击打一个箱子，投掷类的视频标记的 ground truth 只有人做动作的部分不包括物体空中飞行的部分。有些 ground truth 是回放内容，会存在慢动作的情况。不同类之间可能有一定的关系，例如 CliffDiving 的所有 ground truth 都包含在 Diving 中，Diving 中会有额外的动作（例如从泳池跳板跳水）；CricketBowling 和 CricketShot 两个动作往往相连，之间可能还存在重叠时间。不同类别所包含的 ground truth 数量不同，从二三十个到上百个，动作持续时间不同，击球之类的动作可能持续一秒不到，跳远跳高等动作持续时间可以达到十多秒。有些动作会因为之后接了回放等原因相对来说不算完整（例如跳高并没有跨到栏杆的镜头）。

```
7 BaseballPitch 投掷棒球
9 BasketballDunk 扣篮
12 Billiards 击打台球
21 CleanAndJerk 挺举
22 CliffDiving 高处跳水
23 CricketBowling 板球投球
24 CricketShot 板球击球
26 Diving 跳水
31 FrisbeeCatch 接飞盘
33 GolfSwing 挥高尔夫球杆
36 HammerThrow 挥链球
40 HighJump 跳高
45 JavelinThrow 投掷标枪
51 LongJump 跳远（有助跑）
68 PoleVault 撑杆跳
79 Shotput 投掷铅球
85 SoccerPenalty 足球点球
92 TennisSwing 挥网球拍
93 ThrowDiscus 丢铁饼
97 VolleyballSpiking 击打排球
```

VIRAT: 我只把数据集先下载下来了，视频有 1920x1080 和 1280x720 两个分辨率，拍摄的范围很大，摄像头固定在一个指定位置。

RC3D Caffe 代码: 通过 http://ethereon.github.io/netscope/#/editor 可将 caffe 中的 prototxt 文件进行可视化处理。 看了 nms 和 rpn 两部分的代码。NMS 部分和 https://www.cnblogs.com/king-lps/p/9031568.html 差不多，博客中说明了二维情况下的算法，在时域下处理会少两个坐标变量。RPN 部分还没有全部看完，这一部分找了一份 pytorch 的 faster rcnn 的代码 https://github.com/jwyang/faster-rcnn.pytorch/blob/master/lib/model/rpn，因为想知道这部分结构在 pytorch 里面怎么和整个网络连接的，生成的 anchor 的具体输入输出的逻辑还没有完全搞明白。