本周在实验中终于完美复现了作者的结果，但是道路还是有些许曲折，作者提供的网络在 UCF101 上 map 为 0.6410，在 UCFSports 上为 0.8249。本周的测试中，在 UCF101 上达到 0.6489，UCFSports 上达到 0.8337。另外对于学习率和权重衰减的设置中还发现了一些值得注意的地方。完成 vgg 后，将网络前面提取 feature map 的部分换成了 resnet101，后面的几层仍然采用 SSD 的方案，做到之前的 feature map 大小完全兼容。

base-model	dataset	base_lr	weight_decay	loc loss	conf loss	loss	map	epoch
vgg16	UCFSports	0.0001	5.00E-04	0.4103	0.9734	1.3837	0.8246	108
		0.0001	0.001	0.4994	1.1063	1.6057	0.8244	121
		0.0001	0.001	0.4966	0.934	1.4306	0.8299	137
		0.0001	0.001	0.4428	0.8436	1.2864	0.8337	168
	UCF101	0.0001	5.00E-04	0.9038	1.909	2.8128	0.6007	4
		0.00001	5.00E-04	0.8558	1.7774	2.6332	0.6149	5
		0.00001	5.00E-04	0.8329	1.7256	2.5585	0.6170	6
		0.00001	5.00E-04	0.8679	1.7746	2.6425	0.6132	7
		0.0001	0.001				0.6193	5-1600
		0.0001	0.001	0.8456	1.7747	2.6203	0.6171	5
		0.0001	0.001	0.8102	1.6941	2.5043	0.6225	6
		0.0001	0.001	0.7857	1.6815	2.4672	0.6317	7
		0.0001	0.001	0.7797	1.6174	2.3971	0.6261	8
		0.0001	0.001				0.6334	9-1600
		0.0001	0.001	0.7869	1.5689	2.3558	0.6336	9
		0.0001	0.001				0.6387	10-1200
		0.0001	0.001	0.7259	1.5305	2.2564	0.6341	10
		0.0001	0.001	0.722	1.5324	2.2544	0.6403	11-4000
		0.0001	0.001	0.7192	1.5774	2.2966	0.6460	12-12000
		0.0001	0.001				0.6489	12-16000
		0.0001	0.001	0.6908	1.5071	2.1979	0.6465	13-16000

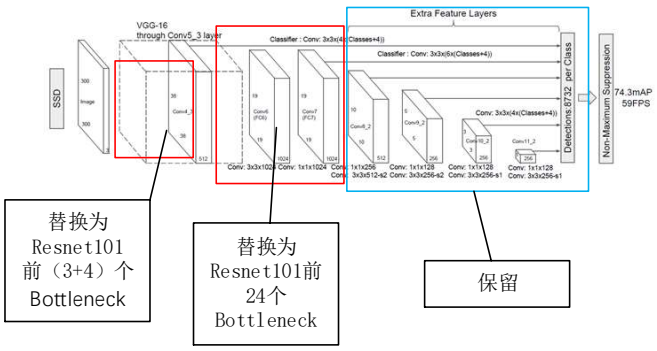
在 vgg 上的测试结果如上图所示，实验过程中为了能够达到比较好的 map 在保持 lr 为 0.0001 的情况下需要经过较长时间的训练，网络训练后期在训练集上的 loss 下降非常缓慢，在测试集上的 map 则呈现跳动，只是随着训练时间的增长略微有上升的趋势，经过足够多的训练可以捕捉到比较满意的结果。

训练过程中，发现如果学习率从 0.0001 的基础上调小，loss 的下降并不明显，反而会导致网络陷入过拟合，网络在训练集上的 map 并没有得到较为明显的提升。另外在知乎上看到一个大佬说权重衰减一般适当调大可以减小过拟合，实验中调整到 0.001 确实比作者给的 5e-4 效果要好一些。

使用 resnet 在 UCFSports 上面完成了两个简单的对比实验，训练速度很快，但是两个的精度并没有明显的优势，经过分析意识到中间有些层 feature map 的膨胀卷积处理还忽略了一些问题。需要继续完善对比。

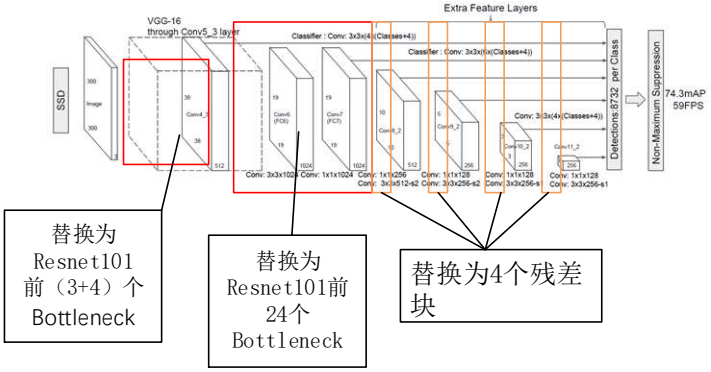
实验 1:

如下图所示，考虑到 feature map 的大小兼容，仅前面特征提取部分采用了预训练的残差模块，后面几层保留。



相比较采用 vgg 的方案，其训练速度快了很多就达到了相同的精度，计算速度也有所提升但是在最终的精度最好的情况只达到了 0.8333，在这一点 resnet 这样的使用方法并没有体现出明显的优势。

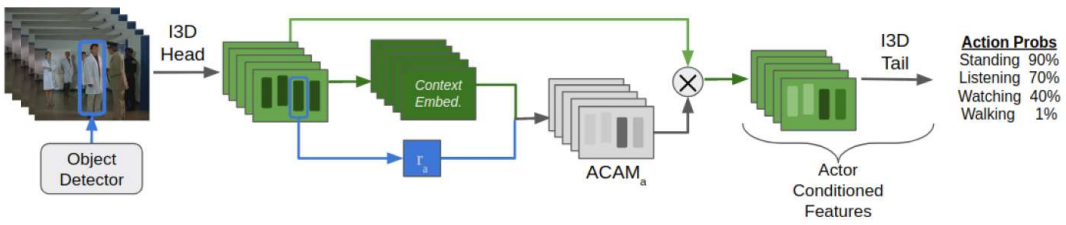
实验二：
如下图所示，将后面四个用于缩小 feature map 的卷积层替换为残差块。



该方案最好的精度只达到了 0.8234，在训练集上的 loss 很快达到了非常低的水平，其对训练集的拟合能力明显比之前的网络都要好，但是随之而来的问题是过拟合，在测试集上的 map 始终得不到提高。

对该结果和之前的网络进行了对比分析，对上面两个实验过程中忽略了采用卷积核的膨胀，现在使用了膨胀卷积的网络正在训练，正好与上述网络形成对比。

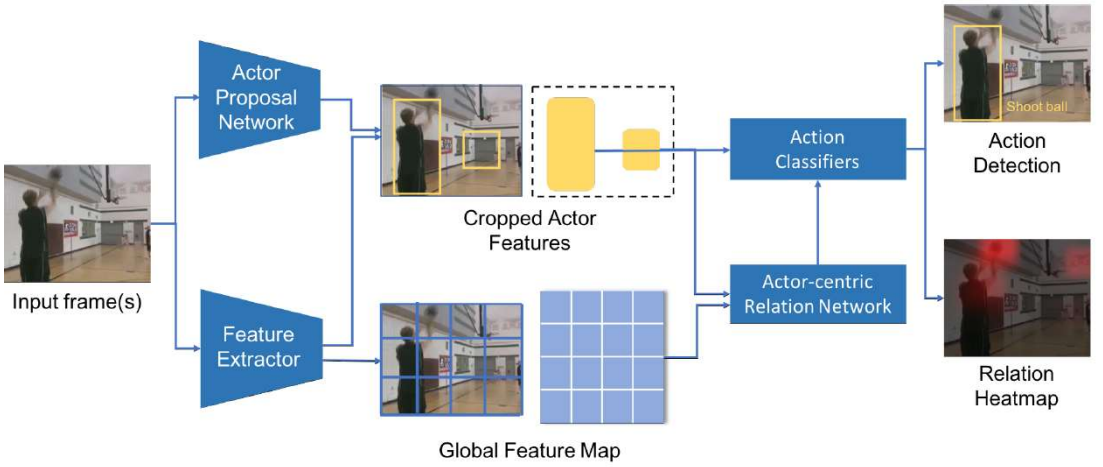
文献方面，发现有人已经在 object relation 方面做了比较多的工作，但是各有关注点。
Actor Conditioned Attention Maps for Video Action Detection：
这篇文章重点在于关注人物之间的关系，他考察的是电影中人和人的互动，所以一个行为可能涉及到多个人。
作者没有只对 ROI 进行考虑，而是将检测出来的人的特征向量在整张 feature map 上面寻找和这个人相关的其他部分，最后生成注意力图，然后根据完整的注意力图融入整张 feature map，将图中无关的信息弱化，相关信息强化，最后用于分类。

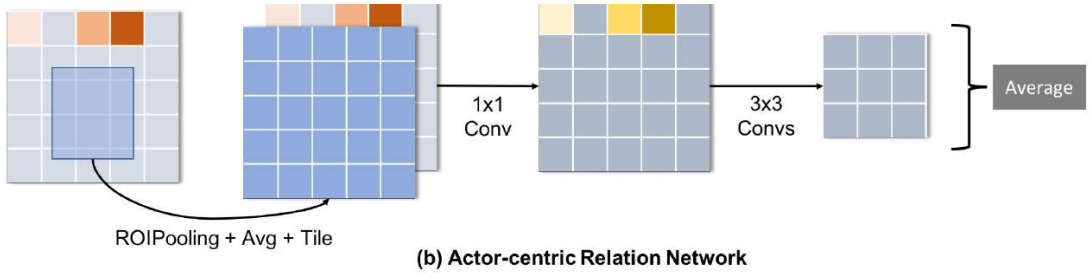


这篇文章的思路也是直接使用预训练的检测器生成高质量，有普遍性的 bbox 后，单独对这些 bbox 考虑它们之间的关系，以对行为进行分类。

Actor-Centric Relation Network：

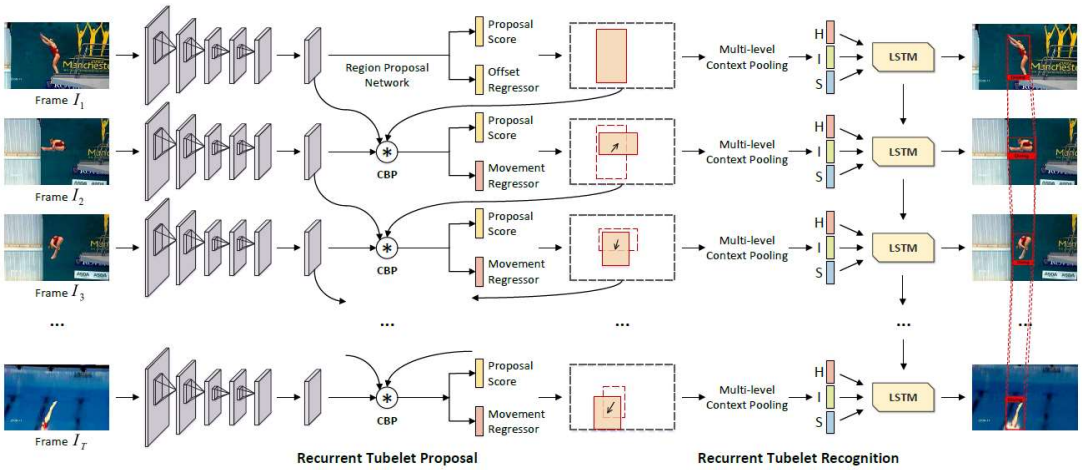
这篇文章则主要关注人和物之间的关系，但是其基本思想和上一篇文章基本上是相似的，也是采用局部特征和全局特征进行匹配的方式寻找人和物的关系。





(b) Actor-centric Relation Network

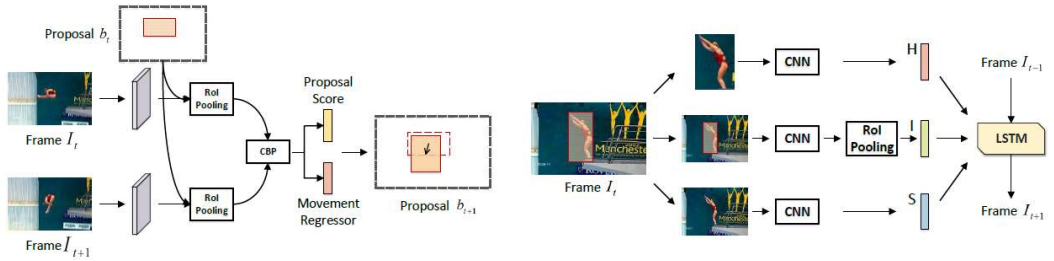
Recurrent Tubelet Proposal and Recognition Networks for Action Detection：



这篇文章使用了递归网络来保留记忆时间上的信息，之前自己也考虑过这个网络，奈何没有什么积累实在是得不出什么结果，如上图所示，相比较现在我使用的 action cuboids，这个递归网络的做法我认为是比较优雅地解决了时序上的困扰，而且这种方法是通过 LSTM 一次性对整个 tubelet 的 class 进行的预测，没有中间的拼合过程。

作者其中一个想法很有意思，用于预测下一帧的位置的 proposal 是来源于上一帧的，网络对当前帧人物位置的回归是基于上一帧的检测结果，因为可以认为两帧之间的区别不太大。

作者在最后的 LSTM 中，融合了 3 个部分的信息：从图片上裁剪下来的人提取出的特征（H），使用全局图像提取特征后，只对人这一块进行 ROI pooling，利用略大的感受野保留人周围的信息（I），以及全图的特征作为背景（S）。该方法确实也考虑到了一定人和物的之间的关系，但是没有上面两个紧密。



(a) RTP (b) RTR

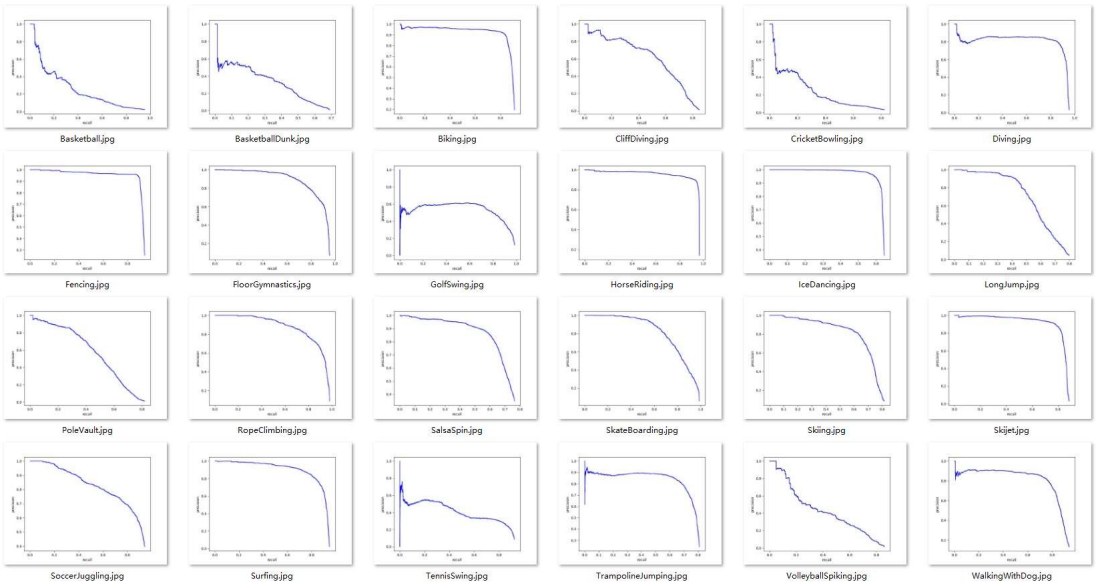
作者的这个想法想要摆脱 RPN 网络，但是如果视频中有新的人物的进出，还是需要采用 RPN 网络来对人物进行检测，因为利用上一帧的结果作为 proposals 只能够对已检测出来的人物进行回归。

但是从效果上来看，作者的这种方法还是极度依赖光流，在 UCF101 这个数据集上，单 RGB 的 map 还不如 action cuboids(我复现的文章)，这是不是也说明 LSTM 其实也不一定好使。

Model	H	I	S	HI	IS	HS	HIS	H	I	S	HI	IS	HS	HIS
	UCF-Sports							**J-HMDB (split 1)**						
RGB	81.5	85.7	82.5	86.4	87.2	83.7	**87.6**	45.2	59.1	46.5	59.9	60.7	53.2	**61.7**
Flow	90.8	95.4	91.2	95.8	96.1	93.4	**96.5**	61.6	76.4	63.6	77.4	77.3	68.5	**78.1**
Fusion	92.3	96.8	93.4	97.4	97.1	95.1	**97.8**	66.3	78.5	70.8	79.8	80.0	74.3	**80.7**
	UCF-101							**AVA**						
RGB	54.8	59.2	56.3	59.4	59.8	58.0	**60.4**	14.4	18.2	13.7	18.9	18.7	17.2	**19.4**
Flow	65.6	71.5	68.1	72.0	72.7	70.5	**73.4**	11.2	13.7	9.4	15.2	14.9	12.5	**15.6**
Fusion	69.3	74.9	71.2	75.2	75.8	72.1	**76.3**	16.3	18.9	15.1	19.7	19.5	18.1	**20.1**

最后用于LSTM的3个部分

对 UCF101 上 AP 图像（下图所示）分析，发现对于瞬时性的动作，持续时间不长的行为的检测效果很差，所以新的问题也提出来，如何让网络去发现这些细微的动作？



就目前了解的情况来看，基本思路可以理解为先检测人，再采用 relation module 来考虑人和场景（全图）之间的关系，我觉得我最初的想法考虑人和物体之间的关系，如果单独一一考虑似乎有太多不必要的工作，因为这样太过于依赖前面的目标检测网络。
同时网络还要有能力聚焦于细微的动作，这有点类似于小目标检测。也许可以通过查阅当前的目标检测如何解决小目标从中得到启发。