状态

- 文章链接
- 已开源(tensorflow实现)

简介

思想就是利用多个template的结合来提高追踪的鲁棒性，本文主要是利用了一个attention选择roi作为LSTM模块的输入，LSTM管理template，输出最合适的template。 为了省一些存储空间，所以采用了存储base template残差。

在siamrpn++之前，一般tracking by detection的performance是最高的，但是速度很慢，只有1fps左右，这种方式是利用gt template来finetune。

Dynamic Memory Networks for Tracking

主要是提出了一个动态记忆网路，拥有读写两种操作。首先是一个backbone CNN网络提取出feature，然后再将之歌feature送入一个attentional LSTM网络。然后通过这个LSTM网络从memory当中选择出一个residual templates跟initial template结合得到final template。将这个final template和search image做conv得到respond map。然后预测bbox，再将bbox 对应的feature crop下来，写入memory当中。

Feature Extraction

给定t帧的输入 I_t，先提提取出search patch S_t，然后计算出SiamFC的feature $f(S_t)$

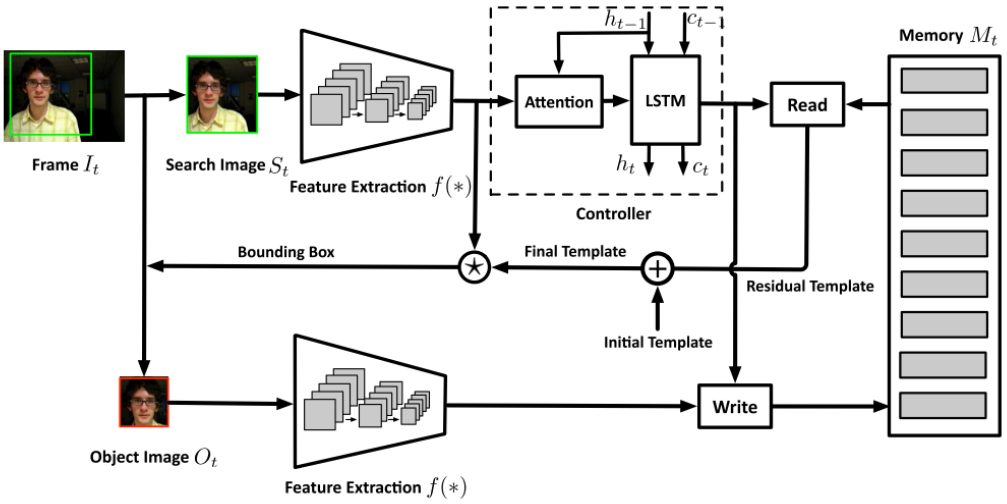

Fig. 1. The pipeline of our tracking algorithm. The green rectangle are the candidate region for target searching. The *Feature Extractions* for object image and search image share the same architecture and parameters. An attentional LSTM extracts the target's information on the search feature map, which guides the memory reading process to retrieve a matching template. The residual template is combined with the initial template, to obtain a final template for generating the response score. The newly predicted bounding box is then used to crop the object's image patch for memory writing.

Attention Scheme

首先要利用attention机制选择search image里target object的位置。
其输入是 $f_{t,i} \in R^{nxnxc}$，也就是第i个nxnxc的feature patch，最后输入LSTM的是每个patch feature的attention based weighted sum。但是因为feature size过大为了减少运算量。所以用AvgPool减少spatial size:

$$f^*(S_t) = \text{AvgPooling}_{n\times n}(f(S_t)) \tag{1}$$

attention的结果如下：

$$\mathbf{a}_t = \sum_{i=1}^{L} \alpha_{t,i}\mathbf{f}^*_{t,i} \tag{2}$$

其中的权重通过softmax计算：

$$\alpha_{t,i} = \frac{\exp(r_{t,i})}{\sum_{k=1}^{L}\exp(r_{t,k})} \tag{3}$$

where

$$r_{t,i} = W^a \tanh(W^h \mathbf{h}_{t-1} + W^f \mathbf{f}^*_{t,i} + b) \tag{4}$$

这里的attention权重是来自于LSTM网络，其输入是每一个square patch和之前的hidden state。这个LSTM的实际含义是，比较LSTM在各个square patch上的hidden state的历史信息和当前信息，这样目标区域会产生比较大的attention weights。

LSTM Memory Controller

LSTM每一帧的输入为经过attention后的feature vector $\mathbf{a}_t$，还有前一帧的hidden state h_{t-1}。其输出是新的hidden state，用来得到read key, read strength, bias gates, decay rate、LSTM的内部框架用的是标准模型但是输出层经过修改来产生需要的控制信号。

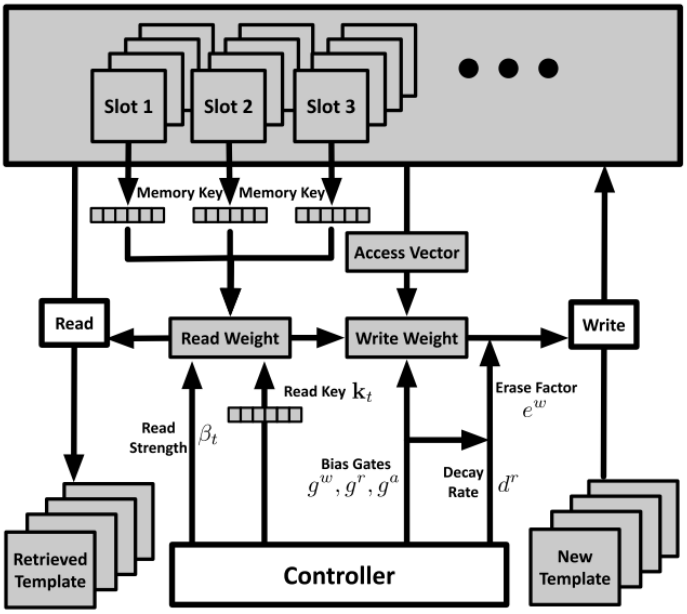

Fig. 3. Diagram of memory access mechanism.

Memory Reading

假设 $M_t \in R^{N\times n\times n\times c}$ 代表记忆模块，那么 $M_t(j) \in R^{n\times n\times c}$ 代表第j个位置上的记忆template，总共有N个位置。输出read key $k_t \in R^c$ 和read strength $\beta_t \in [1,\infty]$：

$$\mathbf{k}_t = W^k \mathbf{h}_t + b^k \tag{5}$$

$$\beta_t = 1 + \log(1 + \exp(W^\beta \mathbf{h}_t + b^\beta)) \tag{6}$$

其中weight key的左右就是匹配内存当中的template，最后会计算出一个read weight:

$$\mathbf{w}_t^r(j) = \frac{\exp\{C(\mathbf{k}_t, \mathbf{k}_{\mathbf{M}_t(j)})\beta_t\}}{\sum_{j'} \exp\{C(\mathbf{k}_t, \mathbf{k}_{\mathbf{M}_t(j')})\beta_t\}} \tag{7}$$

$k_{Mt(j)} \in R^c$ 是由nxn $M_t(j)$ 通过avgpool得到的memory key, $C(x,y)$ 是余弦相似度，最后的template是通过加权求和的方式得到：

$$\mathbf{T}_t^{\text{retr}} = \sum_{j=1}^{N} \mathbf{w}_t^r(j)\mathbf{M}_t(j). \tag{8}$$

Residual Template Learning

直接使用最近一次的template容易导致过拟合，所以作者采用了一种残差 + channel-wise gate vector的方法来得到最后用于corr的template。

$$\mathbf{T}_t^{\text{final}} = \mathbf{T}_0 + \mathbf{r}_t \odot \mathbf{T}_t^{\text{retr}}, \tag{9}$$

- T_0 最开始的gt template
- $\odot$ channel wise mul
- $r_t \in R^C$ 是由LSTM control产生的residual gate

$$\mathbf{r}_t = \sigma(W^r \mathbf{h}_t + b^r), \tag{10}$$

residual gate决定了最后的residual由每个channel占多少比例，其实相当于feature selection（个人觉得用一个卷积kernel去学基本能起到同样的效果），这个模块相当于特征选择。
作者展示了把每个channel通过反卷积可视化出来，看以看出不同的channel对物体不同部分由不同的响应，所以作者认为这样的channel-wise feature residual learning有利于更新不同的物体部分对应特征。作者在ablation study里也展示了这个部分对performance的影响最大。

memory writing

这个模块决定了新的template如何被写道memory当中，作为下一帧的目标继续追踪。
这里牵扯到了有多少新的部分被保留，多少历史信息被提出和留下来。通过学习不同的权重来决定每个部分的比例。
这个模块的设计和LSTM当中的memory保存机制很像。我个人觉得设计的有些过于复杂。

实验及结果分析

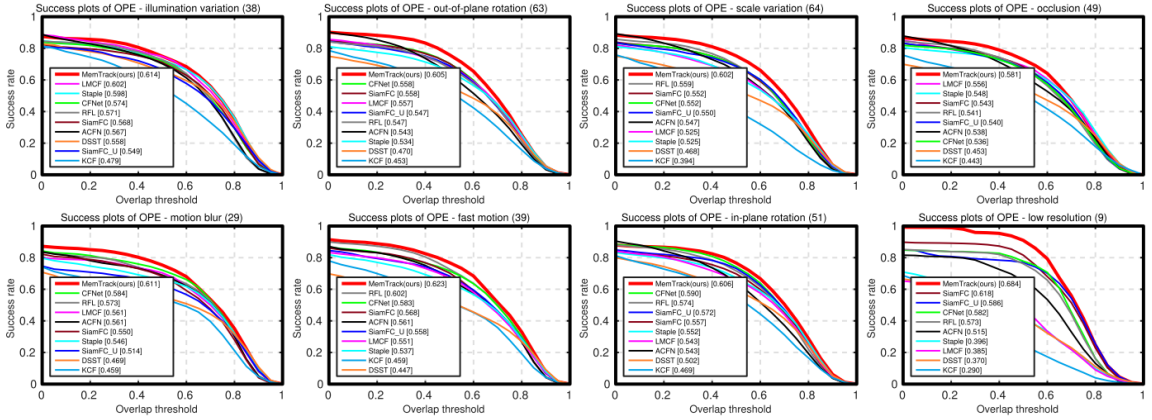

Fig. 9. The success plot of OTB-2015 on eight challenging attributes: illumination variation, out-of-plane rotation, scale variation, occlusion, motion blur, fast motion, in-plane rotation and low resolution

Trackers	MemTrack	SiamFC	RFL	HCF	KCF	CCOT	TCNN	DeepSRDCF	MDNet
EAO (↑)	0.2729	0.2352	0.2230	0.2203	0.1924	0.3310	0.3249	0.2763	0.2572
A (↑)	0.53	0.53	0.52	0.44	0.48	0.54	0.55	0.52	0.54
R (↓)	1.44	1.91	2.51	1.45	1.95	0.89	0.83	1.23	0.91
fps (↑)	50	86	15	11	172	0.3	1	1	1

Table 1. Comparison results on VOT-2016 with top performers. The evaluation metrics include expected average overlap (EAO), accuracy and robustness value (A and R), accuracy and robustness rank (Ar and Rr). Best results are bolded, and second best is underlined. The up arrows indicate higher values are better for that metric, while down arrows mean lower values are better.

可以发现对于一些遮挡，形变，亮暗变化等因素的鲁棒性得到了进一步加强。