15th IAPR International Conference on Machine Vision Applications (MVA)
Nagoya University, Nagoya, Japan, May 8-12, 2017.

A neural network approach to visual tracking

Zhe Zhang, Kin Hong Wong*, Zhiliang Zeng, Lei Zhu
Department of Computer Science and Engineering, The Chinese University of Hong Kong,
HSH Engineering Building, CUHK, Shatin, Hong Kong
* khwong@cse.cuhk.edu.hk

Abstract

Fully Convolution Networks (FCNs) have been shown to be effective in semantic segmentation through fine-tuning classification networks on segmentation data. In this paper, we present that FCNs can be further fine-tuned on target-background images in order to solve visual tracking problems. Pixel level models (FCNs) trained on segmentation data are superior to class level models (e.g. VGG net and GoogLeNet) in visual tracking tasks due to their powerful ability in discriminating between objects and background. Our work is based on a FCN network structure. The result is achieved by first fine-tuning the first image of a sequence and then the tracking and updating processes are conducted through classical forward and backward processes of neural networks. The proposed model achieves high precision and tracking success rates in online object tracking benchmark (OTB) data. It indicates our approach is competitive to state-of-the-art approaches as well.

1 Introduction

Visual tracking is important in many applications such as robotics, automatic driving and surveillance etc. Traditional approaches include mean-shift based trackers [1, 2] that transfer the tracking problem to be probability density estimation problems. Recently, deep convolutional neural networks (CNNs) enjoy great popularity due to their success [3] in many applications, such as object detection [4]. They also outperformed many approaches in ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) [5]. The approach of Fully Convolutional Networks (FCNs) is first proposed [6] for semantic segmentation. The key feature of this network is that the fully connected layers in CNNs models, such as VGG net [7] and GoogLeNet [8], are replaced with convolutional layers in order to keep image spatial information throughout the whole network. A more recent work, FCNT [9], is similar to ours. The method first selects two feature maps one from the top layer and one from a lower layer, then export them to two networks for performing detection. This pipeline is complex and slow. In contrary, our approach uses one network, straight from image pixels to the output-bounding box, so it can achieve higher efficiency. We believe the problems of semantic segmentation and visual tracking are similar. The goal is to distinguish between the object and the background. Both tasks produce the pixel-level output. Moreover, in segmentation each pixel of the input image needs to be annotated to some category during semantic segmentation, while in object tracking each pixel is used to find the target center. Thus, we propose to utilize FCNs [6] to pre-train on the segmentation data as our tracking prior. In our approach, a slight modification on the network structure is needed, that is the fine-tuning on the first image of a sequence is made, then FCNs are adapted to the task of visual tracking. The performance of our method is comparable to other state-of-the-art approaches when applied to public benchmarks testing data (OTB [10]). Although our method is implemented with a deep, neural network structure, it can still run at the speed of 15 frames per second using a common computer, faster than many current trackers. This paper is organized as follows: In Section 2, we discuss the neural network structure used. The tracking method is described in Section 3. In Section 4, implementation details and experimental results are shown. Our work is concluded in Section 5.

2 Network Structure

Our network structure is similar to FCN-8s proposed in [6] except that one more convolutional layer is appended after the last feature map layer. The purpose of the newly added convolutional training layer is to transform the segmentation response map (or called heat map) into a target center highlighted response map. The overview structure is illustrated in Figure 1. In order to achieve better efficiency, we performed modifications on the original structure of FCN. The details are described as follows. In the layer *conv1_1*, *pad* is set to be 1 (originally 100); in the layer *fc6*, *pad* with value of 3 is added; the places of *score2* and *score-pool4* in the first *crop* layers are exchanged and in the following *fuse* layer, *score2* is replaced by *score-pool4*; the places of *score4* and *score-pool3* are exchanged; in the following *fuse* layer, *score4* is replaced with *score-pool3*. In contrast to the soft-max loss function defined in [6], our loss function is the Euclidean loss between the predicted response map (output of layer *conv-final*) and the ground-truth response map.

$$\mathcal{L} = \sum_{i,j} \|\widehat{f_{ij}} - f_{ij}\|^2 \tag{1}$$

where $\mathcal{L}$ denotes the objective function, $\widehat{f_{ij}}$ and f_{ij} denote the predicted response map and ground-truth response map respectively, i and j are image coordinates.

3 Tracking: Model training, Target Center prediction, Response map design and Model update

After fine-tuning on the first image of a sequence, subsequent images are input to the network. The corresponding output of the layer *conv-final* is used as the response (heat) map for further process to localize the

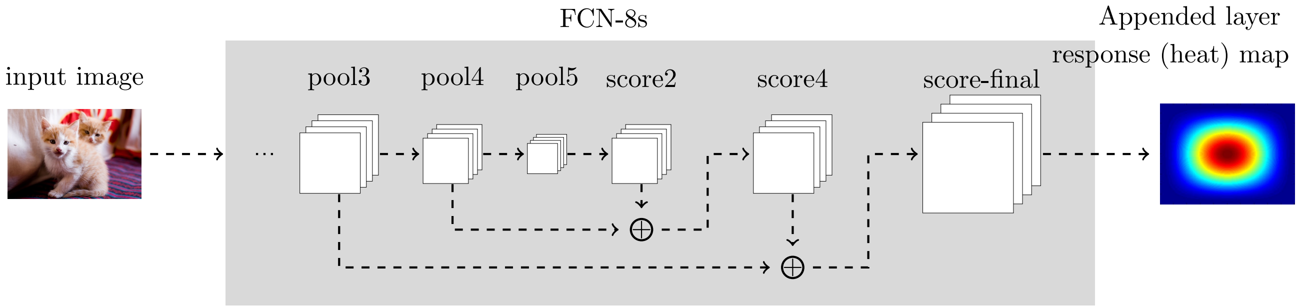

Figure 1. An overview of our network structure. The feature maps in the layer of *pool5* are convolved and up-sampled to become feature maps in the layer of *score2* which are further cropped and summed by the feature maps convolved from the layer of *pool4*. The same trick is applied to the layers of *score4* and *pool3*. We append an additional convolutional layer (response or heat map) after the layer *score-final*, this appended layer is of the same size as the input image.

```
Algorithm 1: FCNs in Visual Tracking
Input: Target location in the starting frame $l_{start}$
Output: Target locations in the following frames $l_{start+1...end}$
Crop the input image patch according to the crop function in Equation 2;
Construct the ground-truth confidence map as in Equation 4;
for $iter \leftarrow 1$ to $iterN$ do
    Forward the network ;
    Backward the network ;
    Update the weights in the network ;
end
for $frame \leftarrow start + 1$ to $end$ do
    Crop the input image patch as in Equation 2;
    Forward the network ;
    Extract the output of layer conv-final and process the confidence map as in Equation 3 ;
    Update the target location according to the confidence map ;
    Backward the network ;
    Update the weights in the network ;
end
```

target center. For each image, the FCN model is updated through the feedback from the loss layer. The whole procedure is listed in Algorithm 1. To begin with, we need to fine-tune our FCNs on the tracking sequences. For each sequence for tracking, we crop a square region centered on the object center given in the first image. The size of the square region is chosen as $\alpha\sqrt{wh}$, where w and h denote the width and the height of the object in the first image respectively. The reason to do so is that we want our input image patch to cover some context information but not too much. Considering that some objects may be very small, we constrain the input image patch to be no smaller than the expected input size of our network. Thus, the final input image patch is cropped as:

$$E = max(s, \alpha\sqrt{wh}) \tag{2}$$

where E denotes the edge length of the square region and s denotes the expected input size of our network. Then no matter how large the input image patch is, we resize it to be s before inputting to the network. The model training is conducted through the typical feed-forward and feed-backward training process the network. Except for the first iteration, all the following iterations are processed between layer *fc6* and layer *loss* in order to save training time. The next step is Target center prediction. After fine-tuning on the first image patch, the network is already able to predict the target center. The input image patch is also cropped as in Equation 2, since we believe that in most sequences the movement of the object is smooth and the object will not be lost by properly selecting the parameters. We forward the input image patch until the layer *conv-final* and the output is a response map inside which each value indicates the possibility of the target center. Due to the assumption of smoothness of the object movements, it is reasonable to introduce a prior which impose higher weights near the center and lower weights in the surrounding regions. In this work, we utilized the Hann window as our prior distribution. Then the confidence map now becomes:

$$\mathcal{C} = \mathcal{P} f_c \tag{3}$$

where $\mathcal{C}$ denotes the confidence map, $\mathcal{P}$ denotes the prior distribution and f_c is the output of the layer *conv-final*. The generated confidence map is filtered with a Gaussian kernel $\mathcal{G}$ to remove noise. Our Response map design is described as follows. Similar to [11], we design the ground-truth response map as a Gaussian shaped distribution

$$f_{ij} = e^{-\frac{\|i-c_x\|^2+\|j-c_y\|^2}{2\sigma^2}} \tag{4}$$

where f_{ij} denotes the value in the response map indexed by i and j, c_x and c_y denotes the horizontal and vertical coordinates of the target center in the input image patch, σ is the predefined parameter to adjust the shape of the distribution. Finally, it is the model update part of the system. The model update are processed through backwarding the network as in model training 3. The backwarding method used in our work is the general stochastic gradient descent (SGD) function in the Caffe toolbox [12]. Since the whole computation is processed inside Caffe without any external functions, we claim that our system is simple and

trackers	FCN1	FCN2
s	224*224	128*128
k	31*31	15*15
p	15	7
α	4	4
$\mathcal{G}(size)$	5*5	5*5
$\mathcal{G}(\sigma)$	1	1
σ	8	4
$lr1$	e^{-12}	e^{-10}
$lr2$	e^{-13}	e^{-11}
$iterN$	100	100

Table 1. Parameters used in our FCN1 and FCN2

straightforward. We have also implemented the MATLAB wrappers for the functions *forward_from_to* and *backward_from_to* to save running time.

4 Implementation and Experiment

We have implemented two trackers named as FCN1 and FCN2. FCN1 and FCN2 are the trackers configured to test the Online Tracking Benchmark (OTB) [10]. In Table 1, s denotes the input image size of the network. k and p denote the kernel size and the padding in the appended convolution layer *conv-final*. The α in Equation 2 is set as 4. The $size$ and σ of the smooth Gaussian kernel $\mathcal{G}$ are 5*5 and 1 respectively. σ denotes the Gaussian shaped response map in Equation 4. The learning rates for the fine-tuning on the first image and model update in the following images are e^{-12} and e^{-13} respectively in FCN1, e^{-10} and e^{-11} respectively in FCN2 and e^{-11} and e^{-12} respectively in FCN3. The fine-tuning lasts for 100 iterations while the model update in the subsequent images is only one pass forward and backward. We evaluate our proposed approach on a commonly used benchmark in visual tracking, i.e. Online Tracking Benchmark (OTB) [10]. The detailed system settings and experimental results are described in the following. Our system is implemented in MATLAB R2013a on top of the deep learning toolbox Caffe [12]. We rewrite some parts of the *matcaffe.cpp* file and add some new functions. The pre-trained model and basic deploy files are the *fcn-8s-pascal* files in [6]. All the parameters in our model are kept the same throughout all the sequences of both benchmarks. Online Tracking Benchmark [10] is used here. Two metrics are used. Precision plot measures the average Euclidean distance between the center locations of the tracked objects and the ground-truth. The success plot measures the overlapping ratio between the tracked object windows and the ground-truth windows. Apart from the conventional one-pass evaluation (OPE), two types of evaluation methodologies, i.e. temporal robustness evaluation (TRE) and spatial robustness evaluation (SRE) are added to test the robustness the trackers. In addition, all the sequences are marked by 11 different attributes in order for further performance analysis of the trackers in distinct situations. The tracker names: ASLA,CSK,CXT,..etc. are defined in [10]. We rank the trackers based on threshold of 20 for Precision Plot and threshold of 0.5 for Success Plot. The experimental results for OPE are shown in Figure 2. From the overall experimental results, we can observe that our proposed methods outperform the state-of-the-art trackers in both metrics. Probably because the input image size is nearly doubled, the results of FCN1 in both Precision Plots and Success Plot are approximately 3% higher than that of FCN2 (0.877 v.s. 0.845 and 0.738 v.s. 0.709). However, even the results of FCN2 are marginally better than the best state-of-the-art approach MEEM [13]. Observed from Table 2, FCN1 demonstrates the best performance in nearly all attributes, except in deformation and out-of-view, where FCN2 has the best performance. Although FCN2 is inferior to FCN1, it is still competitive to the best state-of-the-art trackers, i.e. MEEM [13] and RPT [14]. It is well-worth to note that the improvements of our methods (both FCN1 and FCN2) on low resolution sequences are considerable, about 30 percent improvement on error evaluation and 17 percent improvement on overlap evaluation. In terms of running speed, KCF is the fastest approach with around 30 frames per second. The proposed FCN1 runs at approximately 3 frames per second, close to RPT and MEEM. At the cost of losing 3 percent accuracy performance, FCN2 runs at 10 frames per second, nearly 3 times as fast as FCN1. More information of this work can be found at http://www.cse.cuhk.edu.hk/ khwong/papers.html

5 Conclusion

In this paper, we propose to apply the method of fully convolutional networks to the visual tracking task. After per-trained on segmentation data and fine-tuned on the first image of a sequence, our FCNs can track the targets and online update the model efficiently. The whole pipeline is straightforward and simple but quite effective. Experimental results on both OTB and VOT benchmark show that our tracker is competitive to state-of-the-art trackers.

Acknowledgement

This work is supported by a direct grant (Project Code: 4055045) from the Faculty of Engineering of the Chinese University of Hong Kong.

References

[1] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *PAMI*, 25(5):564–577, 2003.

[2] Zhe Zhang and Kin Hong Wong. Pyramid-based visual tracking using sparsity represented mean transform. In *Proceed. of IEEE Conf. on Comp. Vision and Patt. Rec.*, pages 1226–1233, 2014.

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105. 2012.

[4] Christian Szegedy, Alexander Toshev, and Dumitru Erhan. Deep neural networks for object detection. In *NIPS*, pages 2553–2561. 2013.

[5] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, pages 1–42, 2015.

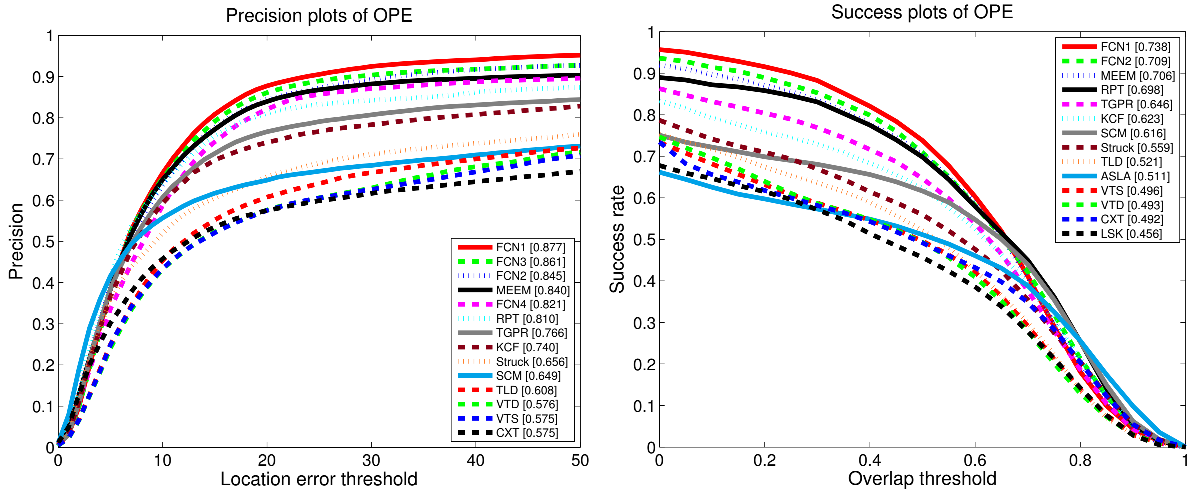

Figure 2. Overall experimental result of one-pass evaluation on VTB. We choose to compute the performance scores in the Precision Plots of OPE (left) at the location error threshold of 20 and Success Plot of OPE (right) at the overlap threshold of 0.5 respectively.

		ASLA	CSK	CXT	VTS	VTD	TLD	SCM	Struck	KCF	TGPR	RPT	MEEM	our FCN1	our FCN2
IV	error	0.517	0.481	0.501	0.573	0.557	0.537	0.594	0.558	0.728	0.687	0.825	0.778	0.867	0.816
	overlap	0.503	0.388	0.416	0.503	0.480	0.460	0.568	0.491	0.581	0.591	0.684	0.667	0.711	0.681
SV	error	0.552	0.503	0.550	0.582	0.597	0.606	0.672	0.639	0.679	0.703	0.802	0.808	0.859	0.822
	overlap	0.544	0.352	0.435	0.453	0.460	0.494	0.635	0.471	0.479	0.508	0.627	0.599	0.616	0.582
OCC	error	0.460	0.500	0.491	0.534	0.545	0.563	0.640	0.564	0.749	0.708	0.764	0.814	0.841	0.806
	overlap	0.451	0.404	0.434	0.465	0.468	0.468	0.599	0.493	0.618	0.596	0.628	0.694	0.702	0.672
DEF	error	0.445	0.476	0.422	0.487	0.501	0.512	0.586	0.521	0.740	0.768	0.746	0.859	0.883	0.915
	overlap	0.456	0.370	0.372	0.441	0.443	0.456	0.565	0.473	0.671	0.700	0.664	0.719	0.801	0.803
MB	error	0.278	0.342	0.509	0.375	0.375	0.518	0.339	0.551	0.650	0.578	0.781	0.740	0.829	0.795
	overlap	0.281	0.336	0.361	0.328	0.320	0.482	0.339	0.518	0.595	0.553	0.712	0.721	0.777	0.751
FM	error	0.253	0.381	0.515	0.353	0.352	0.551	0.333	0.604	0.602	0.575	0.743	0.757	0.809	0.761
	overlap	0.260	0.380	0.413	0.325	0.319	0.473	0.335	0.567	0.557	0.548	0.688	0.726	0.741	0.708
IPR	error	0.511	0.547	0.610	0.579	0.599	0.584	0.597	0.617	0.725	0.706	0.794	0.809	0.843	0.808
	overlap	0.488	0.457	0.533	0.477	0.500	0.476	0.560	0.528	0.615	0.601	0.692	0.643	0.699	0.649
OPR	error	0.518	0.540	0.574	0.604	0.620	0.596	0.618	0.597	0.729	0.741	0.806	0.853	0.865	0.852
	overlap	0.494	0.439	0.487	0.496	0.510	0.497	0.575	0.506	0.608	0.623	0.681	0.695	0.720	0.685
OV	error	0.333	0.379	0.510	0.455	0.462	0.576	0.429	0.539	0.650	0.495	0.641	0.730	0.786	0.810
	overlap	0.359	0.410	0.496	0.508	0.491	0.516	0.449	0.550	0.650	0.519	0.634	0.741	0.794	0.807
BC	error	0.496	0.585	0.443	0.578	0.571	0.428	0.578	0.585	0.753	0.761	0.840	0.808	0.884	0.809
	overlap	0.468	0.491	0.392	0.516	0.515	0.388	0.550	0.545	0.672	0.692	0.752	0.741	0.803	0.729
LR	error	0.156	0.411	0.371	0.187	0.168	0.349	0.305	0.545	0.381	0.539	0.478	0.494	0.847	0.808
	overlap	0.163	0.397	0.340	0.183	0.170	0.327	0.308	0.410	0.357	0.399	0.391	0.472	0.640	0.609
overall	error	0.532	0.545	0.575	0.575	0.576	0.608	0.649	0.656	0.740	0.766	0.810	0.840	0.877	0.845
	overlap	0.511	0.443	0.492	0.496	0.493	0.521	0.616	0.559	0.623	0.646	0.698	0.706	0.738	0.709

Table 2. Detailed results on all attributes, i.e. illumination variation (IV), scale variation (SV), occlusion (OCC), deformation (DEF), motion blur (MB), fast motion (FM), in-plane rotation (IPR), out-of-plane rotation (OPR), out-of-view (OV), background clutters (BC), low resolution (LR). The thresholds for error and overlap evaluations are 20 pixels and 50 percent respectively. For each row, the rank-1st, rank-2nd and rank-3rd results are marked in red, green and blue respectively.

[6] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1411.4038, 2014.

[7] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[8] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[9] L. Wang, W. Ouyang, X. Wang, and H. Lu. Visual tracking with fully convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3119–3127, 2015.

[10] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Online object tracking: A benchmark. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2411–2418, 2013.

[11] J.F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. *PAMI*, pages 583–596, 2015.

[12] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[13] Jianming Zhang, Shugao Ma, and Stan Sclaroff. Meem: Robust tracking via multiple experts using entropy minimization. In *ECCV*, pages 188–203. 2014.

[14] Yang Li, Jianke Zhu, and Steven C.H. Hoi. Reliable patch trackers: Robust visual tracking by exploiting reliable patches. In *CVPR*, pages 353–361, 2015.