Two-Stage Learning to Robust Visual Track via CNNs

Dan Hu[1,2(✉)], Xingshe Zhou[1], Xiaohao Yu[3], and Zhiqiang Hou[2]

[1] School of Computer Science,
Northwestern Polytechnical University, Xi'an, China
plahudan@mail.nwpu.edu.cn
[2] Information and Navigation College,
Air Force Engineering University,
Xi'an, China
[3] General Research Institute,
Equipment Academy of Air Force, Beijing, China

Abstract. Convolutional Neural Networks (CNN) are an alternative type of deep neural network that can be used to model local correlations and reduce translation variations, which have demonstrated great performance in some computer vision areas except the visual tracking due to the lack of training data. In this paper, we explore applying a two-stage learning CNN as a generic feature extractor offline pretrained with a large auxiliary dataset and then transfer its rich feature hierarchies to the robust visual tracking task. Instead of traditional neuron models in CNNs, we introduce a strategy to use ReLU for training acceleration. Empirical comparisons prove our CNN based tracker outperforms several state-of-the-art methods on an open tracking benchmark.

Keywords: Visual tracking · Deep learning · Convolutional neural network

1 Introduction

Visual tracking is a fundamental problem in computer vision with a wide range of applications, such as video surveillance for security, intelligent transportation system, and human-computer interface. Although existing applied visual tracking techniques in well-controlled environments, the challenging requirements for practical applications is how to long-term track continuously changed target, which is triggered by the harsh nature, including partial occlusion, illumination change, shape deformation, background interfering, etc. The key point to resolve the above problem is how to improve the robustness of visual tracking algorithm [1].

Unfortunately, in most existing trackers, even those reporting satisfactory results, features are manually defined and combined [2, 3], which may not be good enough for robust tracking, due to the limitations of prior knowledge about the object and the complex environments. Recently, deep learning, which are machine learning algorithms inspired by brains, based on learning multiple levels of representation, have gained significant attention thanks to their success on automatic feature extraction via multi-layer nonlinear transformations, especially in computer vision [4], speech recognition [5] and natural language processing [6].

Y.-J. Zhang (Ed.): ICIG 2015, Part III, LNCS 9219, pp. 491–498, 2015.
DOI: 10.1007/978-3-319-21969-1_44

However, the application of deep learning in visual tracking is less explored. The reason is, in the case of visual tracking, we typically have only very few positive instances extracted from the first video frame for training (in fact mostly we only have one single labeled example), which makes the direct applying of the deep learning approaches infeasible.

In this work, we attempt to train a two-stage Convolutional Neural Network (CNN) [7] as a generic image feature extractor by a purely supervised learning, that is offline pretrained and then transfer its rich feature hierarchies into online tracking to overcome this problem. Our proposed method is similar in spirit to DLT [8], the first work on applying deep neural networks that is a Stacked Denoising Autoencoder (SDAE) [9], to visual tracking, and has reported encouraging results, but there are some key differences that are worth noting:

1. Fully-connected deep learning models, such as SDAE used in DLT, ignore the topology and correlation of 2D images as need to learn weights separately for every location. However, CNNs use convolution and weights sharing technique to capture better local and repetitive similarity in images with much fewer connections and parameters. So our model is easier to train.
2. Shifts or distortions may cause the position of object to vary, which desires the model to incorporate translation invariance, that could be captured by weights sharing and pooling mechanisms much more efficiently in CNNs. So our algorithm demonstrates better performance in some scenes with occlusion and illumination changes.
3. To make training faster, we use ReLU function and a very efficient GPU implementation of the convolution operation.

We evaluate our proposed algorithm through quantitative and qualitative comparisons with DLT and other state-of-the-art trackers on an open tracking benchmark, which manifests the promising substantial improvements over the other trackers.

The rest of this paper is organized as follows: Sect. 2 describes our CNN model. Section 3 presents the details of the CNN tracking algorithm. Section 4 shows promising comparative results and Sect. 5 summarizes the conclusion.

2 Our Model

The architecture of our network is depicted in Fig. 2. Below, we first describe some novel or unusual feature of our network's architecture.

2.1 Rectified Linear Units

Instead of using the standard way to model a neuron's output f as a function of its input x with sigmoid function $f(x) = (1 + e^{-x})^{-1}$ or $f(x) = \tanh(x)$, we refer to neurons with nonlinearity as Rectified Linear Units (ReLUs), that is non-saturating nonlinearity

function $f(x) = \max(0, x)$ introduced by Nair and Hinton [10], in terms of reducing training time with gradient descent. Deep convolutional neural networks with ReLUs train several times faster than their equivalents with tanh units, as demonstrated in Fig. 1, which shows the number of iterations required to reach 25 % training error on the CIFAR-10 dataset for a particular four-layer convolutional network [4]. From this plot, it can be seen that networks with ReLUs consistently learn about 6 times faster than equivalents with traditional neurons, which would have a great influence on the performance of large models trained on large dataset. Thus, in this work, we employ ReLU function to our CNN neurons.

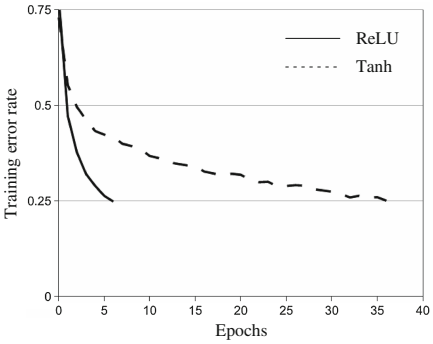

Fig. 1. Comparison of the number of iterations required to reach 25 % training error in a CNN with ReLU and Tanh function [4].

2.2 Our CNN Model

Now we are ready to describe the overall architecture of our own CNN which would be constructed as a generic feature extractor and then be transfered its rich feature hierarchies into our online tracking framework.

As depicted in Fig. 2, our CNN contains two convolutional layers, corresponding ReLUs as activation function and max-pooling operators. The input to the CNN's visible layer is locally normalized 32 × 32 image patches. The first convolutional layer filters the input image with 10 kernels of size 11 × 11, and the second convolutional layer takes as input the output of the first convolutional layer and filters it with 16 kernels of size 5 × 5. All these kernels scan each image in the previous layer with different weight vectors, and the max-pooling operators over the local neighborhoods reduce the resolution from the feature maps derived by the former convolution operators. The fully-connected layer is connected to all neurons in the previous layer, and the output of the full-connected layer is fed to a 256-way softmax which produces a distribution over the 256 class labels. Thus, the number of neurons in the network's remaining layers is given by (32 × 32) − (22 × 22 × 10) − (11 × 11 × 10) − (7 × 7 × 160) − (4 × 4 × 160) − (1024) − (256).

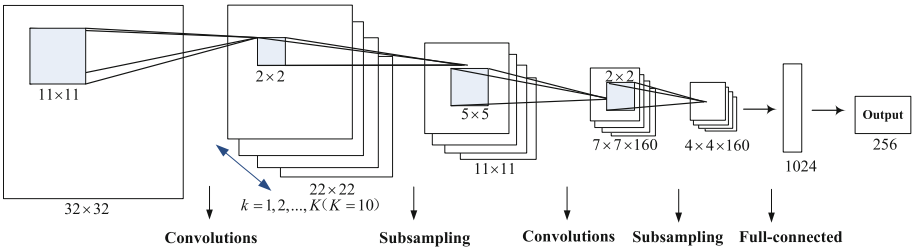

Fig. 2. Architecture of our CNN.

3 Our CNN Tracker

Now we present our tracker based on a two-stage learned CNN. In the first place, we employ a large dataset to offline pretrain a CNN to extract rich feature hierarchies, and then we transfer the learned features to the online tracking tasks to distinguish the tracked object from its surrounding background. The online tracking process will select the region with the highest score, which is the output of a softmax classifier on top of the CNN, as the new location of the object based on a particle filter framework. The whole CNN model is finetuned in a lazy manner only when significant appearance change happens.

3.1 Learning Stage 1: Offline Pretraining

In this work, we use Tiny Images [11], which is a dataset of 79 million unique 32 × 32 color images gathered from the Internet. Each image is loosely labeled with one of 75,062 English nouns, which covers all visual object classes. Despite their low resolution, it is still possible to recognize most of the objects and scenes. Meanwhile, the dataset contains more copious and abundant amount of images which have related objects in similar spatial arrangements. In [12], the authors experimentally probe that pretraining with such training data, would benefit and lead to a large improvement in detection performance. All of these motivate our choice of these low resolution images which can effectively reduce computational complexity.

We randomly sample 1 million images from this dataset with 256 classes as the inputs of our CNN model to pretrain it to be a generic feature extractor. We did not pre-process the images in any other way, except for scaling the raw pixels to the range [0, 1] linearly.

3.2 Learning Stage 2: Online Tracking and Update

Inspired by the work of Wang et al. [8], our visual tracking algorithm is also carried out based on a particle filter framework, which is a sequential Monte Carlo importance sampling method for estimating the latent state variables of a dynamical system based on a sequence of observations.

In the first frame, the object to track has been provided by the bounding box. Then the object region and the surrounding regions are regarded as positive sample and negative samples, correspondingly, which are used to fine-tune the CNN to adapt to the appearance of object in the first frame. When a new video frame arrives, the confidence for each particle is made by the network's softmax by making a simple forward pass through the network.

Instead of updating the CNN model at each frame, which would be computationally expensive, we propose to update the CNN in a lazy manner, only when the maximum confidence of all particles in a frame is below a predefined threshold, which indicates significant appearance change of the object being tracked occurs. This method accelerates our tracking algorithm exceedingly on the reason that the appearance of the object is not always changing in adjacent frames, our CNN model can remain discriminant until significant appearance change happens.

4 Experiments

4.1 Experimental Setup

(1) Evaluation Dataset: We evaluate the performance of our proposed method on a recently released benchmark [13], which is the largest open dataset consisting of 50 fully annotated sequences and attributes, to facilitate tracking evaluation. These attributes are defined by the factors that affect tracking performance, such as occlusion, fast motion, and illumination variation. We compare our CNN tracker with some state-of-art trackers, including DLT [8], MIL [14], IVT [15], CT [16], and VTD [17].

(2) Evaluation Metrics: Performances are measured by tracking success rate (TSR) and tracking precision (TP). Tracking success rate is calculated by the percentage of frames in which the overlapping ratio between the estimated location and the ground truth against the entire union box is larger than 50 %. Tracking precision is defined as the Euclidean distance between the center of bounding box and the ground truth in pixels.

(3) Implementation Details: We run our algorithm in Matlab on a desktop PC with a 3.2 GHz i5 quad core CPU and a NVIDIA GTX750 GPU, by invoking the Matlab parallel computing toolbox to accelerate the computation. We use the contrastive divergence algorithm with momentum for optimization. We start with learning rate of 0.1 with momentum 0.5 and increase it to 0.9 after 5 epochs. We train about 20 epochs in total with the batch size to 100. The threshold for online fine-tuning the whole network is set to 0.8. The particle filter uses 1000 particles in a search window twice to the area of bounding box around the estimated location in last frame. We also run the DLT code[1] on our platform. The results of other trackers are obtained from [13].

[1] http://winsty.net/dlt.html.

4.2 Experiment Comparison

Performances of the 6 tracker over 8 video sequences are summarized in Table 1, the best results are highlighted in bold font. It's clearly observed that our proposed method achieves the best results compared with other trackers on 5 video sequences. For the other 3 video sequences, ours is also among the best three methods, and all results outperform DLT. The key to this success is the translation invariance gained by weights sharing and pooling mechanisms, which make our algorithm demonstrates better performance in some scenes with occlusion and illumination changes.

Table 1. The performance comparison of our proposed method and the other visual trackers. The results are shown in the order of TSR/TP.

	Woman	David	Shaking	Trellis	Girl	Singer1	Bolt	David3
Ours	**83.1/7.5**	73.8/6.0	70.9/7.2	**96.4/3.0**	**83.7/2.9**	**100/2.7**	40.3/85.8	**69.3/50.9**
DLT	67.1/9.4	66.1/7.1	35.4/11.5	93.6/3.3	73.5/4.0	100/3.3	2.3/388.1	33.3/104.8
MIL	12.2/123.7	17.7/13.1	26.0/28.6	25.9/71.7	29.4/13.7	10.3/26.0	1.1/393.5	68.3/29.7
IVT	21.5/111.2	**92.0/3.9**	1.1/138.4	44.3/44.7	18.6/22.5	96.3/7.9	1.4/397.0	63.5/52.0
CT	16.0/109.6	25.3/15.3	92.3/10.9	23.0/80.4	17.8/18.9	10.3/16.8	0.6/363.8	34.9/88.7
VTD	17.1/133.6	49.4/27.1	**99.2/5.2**	30.1/81.3	**-/-**	99.4/3.4	**55.7/14**	**-/-**

Thanks to the GPU and ReLU which accelerate our CNN training, our tracker achieve an average frame rate of 19.6 fps on our platform, as shown in Table 2, which is sufficient for many real-time applications.

Table 2. Running time on 8 video sequences (fps)

Woman	David	Shaking	Trellis	Girl	Singer1	Bolt	David3	Average
25.73	20.08	19.26	23.50	16.85	18.73	14.67	18.17	19.62

Figure 3 shows some key frames with bounding boxes reported by all 6 trackers for each of the 8 video sequences, which present our tolerance to occlusions, pose and illumination changes.

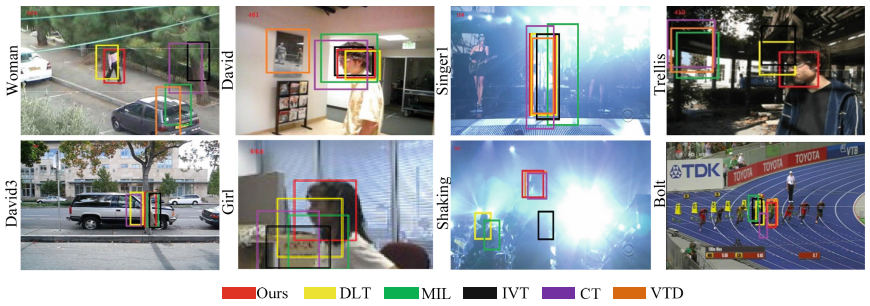

Fig. 3. Comparison of 7 trackers on several key frames of 8 video sequences.

The woman and david3 sequence are challenging for severe occlusions and pose changes. Our tracker doesn't drift for woman whilst most other trackers fail or drift at about frame 550. For david3, our tracker rarely misses the target completely expect full occlusion. The girl, david, shaking, singer1 and trellis are all arduous since drastical pose changes in addition to illumination vary for the last three. For singer1 and trellis, our method can track the object accurately along the entire sequence. For girl, most trackers drift at about frame 86, while our method can track the girl even after the girl turns. For david, all trackers drift or even fail to different degrees except for IVT, our tracker yield the second best results. For shaking, VTD and CT give satisfactory results, followed by ours which is much better than DLT. The Bolt is challenging for the severe deformation, most trackers fail or drift in early frames, and our results yield the second best result followed by VTD.

Our tracker shows the promising performance for most video sequences. The empirical comparisons suggest the outstanding future for the application of deep learning in visual tracking. Furthermore, our results can be improved simply by waiting for faster GPUs and bigger datasets to become available.

5 Conclusion

We have proposed a novel two-stage learning method for visual tracking based on Convolutional Neural Networks. To realize this approach, we first train a CNN model using an auxiliary Tiny Images dataset to learning generic image feature representation. Then we transfer it to a particle filter online tracking framework, which predicts the new location with highest confidence using the output of our CNN. Our CNN model was finetuned only when significant appearance change occurs. Empirical comparisons demonstrate that CNN based tracker achieves encouraging results and CNN has better capability than SDAE in visual tracking application.

Acknowledgements. The authors would like to thank the editors for their time and effort. This research was supported by the National Natural Science Foundation of China (61472391, 61403414)

References

1. Yang, H., Shao, L., Zheng, F., Wang, L., Song, Z.: Recent advances and trends in visual tracking: a review. Neurocomputing **74**(18), 3823–3831 (2011)
2. Adam, A., Rivlin, E., Shimshoni, I.: Robust fragments-based tracking using the integral histogram. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR (2006)
3. Hare, S., Saffari, A., Torr, P.H.: Struck: structured output tracking with kernels. In: IEEE International Conference on Computer Vision, ICCV (2011)
4. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Annual Conference on Neural Information Processing Systems, NIPS (2012)

5. Sainath, T.N., Kingsbury, B., Saon, G., Soltau, H., Mohamed, A.R., Dahl, G., Ramabhadran, B.: Deep convolutional neural networks for large-scale speech tasks. Neural Networks **64**, 39–48 (2015)
6. Socher, R., Liu, C., Ng, A.: Parsing natural scenes and natural language with recursive neural networks. In: International Conference on Machine Learning, ICML (2011)
7. LeCun, Y., Bengio, Y.: Convolutional networks for images, speech, and time-series. In: Arbib, M.A. (ed.) Handbook of Brain Theory and Neural Networks. MIT Press, Cambridge (1995)
8. Wang, N., Yeung, D.Y.: Learning a deep compact image representation for visual tracking. In: Annual Conference on Neural Information Processing Systems, NIPS (2013)
9. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.A.: Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion. J. Mach. Learn. Res. **11**, 3371–3408 (2010)
10. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: International Conference on Machine Learning, ICML (2010)
11. Torralba, A., Fergus, R., Freeman, W.T.: 80 million tiny images: a large data set for nonparametric object and scene recognition. IEEE Trans. Pattern Anal. Mach. Intell. **30**(11), 1958–1970 (2008)
12. Agrawl, P., Girshick, R., Malik, J.: Analyzing the performance of multilayer neural networks for object recognition. In: European Conference on Computer Vision, ECCV (2014)
13. Wu, Y., Lim, J., Yang, M.: Online object tracking: a benchmark. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR (2013)
14. Babenko, B., Yang, M., Belongie, S.: Robust object tracking with online multiple instance learning. IEEE Trans. Pattern Anal. Mach. Intell. **33**(8), 1619–1632 (2011)
15. Ross, D., Lim, J., Lin, R., Yang, M.: Incremental learning for robust visual tracking. Int. J. Comput. Vis. **77**(1), 125–141 (2008)
16. Zhang, K., Zhang, L., Yang, M.-H.: Real-time compressive tracking. In: European Conference on Computer Vision, ECCV (2012)
17. Kwon, J., Lee, K.: Visual tracking decomposition. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR (2010)