Fast Dynamic Convolutional Neural Networks for Visual Tracking

Zhiyan Cui CUIZHIYAN@STU.XJTU.EDU.CN
State Key Laboratory for Manufacturing System Engineering, System Engineering Institute, Xi'an Jiaotong University, Xi'an, Shaanxi, China

Na Lu* LVNA2009@MAIL.XJTU.EDU.CN
State Key Laboratory for Manufacturing System Engineering, System Engineering Institute, Xi'an Jiaotong University, Xi'an, Shaanxi, China
Beijing Advanced Innovation Center for Intelligent Robots and Systems, Beijing, China

Xue Jing JXLSF01240614@STU.XJTU.EDU.CN
State Key Laboratory for Manufacturing System Engineering, System Engineering Institute, Xi'an Jiaotong University, Xi'an, Shaanxi, China

Xiahao Shi SH11AO@STU.XJTU.EDU.CN
State Key Laboratory for Manufacturing System Engineering, System Engineering Institute, Xi'an Jiaotong University, Xi'an, Shaanxi, China

Editors: Jun Zhu and Ichiro Takeuchi

Abstract

Most of the existing tracking methods based on CNN(convolutional neural networks) are too slow for real-time application despite the excellent tracking precision compared with the traditional ones. In this paper, a fast dynamic visual tracking algorithm combining CNN based MDNet(Multi-Domain Network) and RoIAlign was developed. The major problem of MDNet also lies in the time efficiency. Considering the computational complexity of MDNet is mainly caused by the large amount of convolution operations and fine-tuning of the network during tracking, a RoIPool layer which could conduct the convolution over the whole image instead of each RoI is added to accelerate the convolution and a new strategy of fine-tuning the fully-connected layers is used to accelerate the update. With RoIPool employed, the computation speed has been increased but the tracking precision has dropped simultaneously. RoIPool could lose some positioning precision because it can not handle locations represented by floating numbers. So RoIAlign, instead of RoIPool, which can process floating numbers of locations by bilinear interpolation has been added to the network. The results show the target localization precision has been improved and it hardly increases the computational cost. These strategies can accelerate the processing and make it 7× faster than MDNet with very low impact on precision and it can run at around 7 fps. The proposed algorithm has been evaluated on two benchmarks: OTB100 and VOT2016, on which high precision and speed have been obtained. The influence of the network structure and training data are also discussed with experiments. The source code is publicly available on github: https://github.com/ZhiyanCui/fdn-release

Keywords: visual tracking, RoIAlign, convolutional neural networks

1. Introduction

Visual tracking is one of the fundamental problems in computer vision which aims at estimating the position of a predefined target in an image sequence, with only its initial state given. Nowadays, CNN has achieved great success in computer vision, such as object classification Krizhevsky et al.; Simonyan and Zisserman (2014), object detection Girshick et al.; Girshick; Ren et al., semantic segmentation Girshick et al.; He et al. (2017); Girshick et al. (2016) and so on. However, it is difficult for CNN to play a great role in visual tracking. CNN usually consists of a large amount of parameters and needs big dataset to train the network to avoid over-fitting. So far, there are some novel studies Ma et al.; Danelljan et al. (2016); Qi et al.; Valmadre et al. (2017) which combine CNN and traditional trackers to achieve the start of the art. Some of them just use CNNs which are trained on ImageNet or other large datasets to extract features.

On one hand, tracking seems much easier than detection because it only needs to conduct a binary classification between the target and the background. On the other hand, it is difficult to train the network because of the diversity of objects that we might track. An object may be the target in one video but the background in another. And there is no such amount of data for tracking to train a deep network.

MDNet Nam and Han is a novel solution for tracking problem with detection method. It is trained by learning the shared representations of targets from multiple video sequences, where each video is regarded as a separate domain. There are several branches in the last layer of the network for binary classification and each of them is corresponding to a video sequence. The preceding layers are in charge of capturing common representations of targets. While tracking, the last layer is removed and a new single branch is added to the network to compute the scores in the test sequences. Parameters in the fully-connected layers are fine-tuned online during tracking. Both long and short updating strategies are conducted for robustness and adaptability, respectively. Hard negative mining Sung and Poggio (1998) technique is involved in the learning procedure. MDNet won the first place in VOT2015 competition. Although the effect is amazing, MDNet runs at very low speed which is 1 fps on GPU. In this paper, we propose a similar but much faster network for object tracking. It consists of 3 convolution layers and 3 fully-connected layers with a RoIAlign layer He et al. (2017) between them. The convolution is operated on the whole image instead of each RoI, and a RoIAlign layer is used to get fixed-size features which can enhance the calculation speed. When we compare the results of RoIAlign and RoIPool Girshick, it shows that RoIAlign could obtain higher precision. Instead of fine-tuning the network with fixed number of iterations, we use a threshold to adjust the numbers dynamically, which will reduce the iterations and time of fine-tuning. Similar training and multi-domain strategy as in MDNet have been adopted. Given k videos to train the network, the last fully-connected layer should have k branches. Each video is corresponding to one branch. So the last layer is domain-specific layers and others are shared layers. In this way, the shared information of the tracked objects is captured.

The k branches of last layer are deleted and a new one is added to the network before tracking. The parameters in last layer are initialized randomly. The positive and negative samples which are extracted from the first frame of the sequence are used to fine-tune the network. Only parameters in the fully-connected layers will be updated. Bounding boxes in

the current frame around the previous target position are sent to the network for discrimination. The sample with the highest score will be obtained as the target in this frame. The fully-connected layers will be updated with samples from the previous several frames when scores of all the samples are less than a threshold. By this dynamic method, the network can learn the new features of the objects.
The proposed algorithm consists of multi-domain learning and dynamic network tracking. The main contributions are shown below:

- A small neural network with RoIAlign layer is created to accelerate the convolution.
- A new strategy of fine-tuning the fully-connected layers is used to reduce the iterations during tracking. About 7 fps on GPU could be reached which makes it 7× faster than MDNet with only a small decrease in precision.
- The performance of different structures of the network has been compared. The result shows that RoIAlign outperforms RoIPool which is usually used in detection and classification. And neither increasing convolution layers nor decreasing fully-connected layers has positive effects on the accuracy.
- Evaluations of the proposed tracker have been performed on two public benchmarks: object tracking benchmark Wang et al. (2017) and VOT2016 Kristan et al. (a). The results show that a higher precision and faster tracking speed than most of the existing trackers based on CNN have been obtained.

2. Related Work

There have been several surveys Wu et al. (2015); Wu et al. of visual tracking during these years. Most of the state of the art methods are based on correlation filter or deep learning. Some trackers have been combined with each other to improve the tracking performance.
Correlation filter (CF) has attracted scientists' attention for several years. It can run at high speed because of the application of Fourier transformation. The calculation is transformed from spatial domain to Fourier domain to speed up the operation. MOSSE Bolme et al. was the first algorithm which uses CF to track objects. CSK Henriques et al. is another algorithm based on CF. It uses circular shift to produce dense samples by making use of more features of the image. And then, there are KCF Henriques et al. (2015) and DCF Henriques et al. (2015). Both of them can use multi-channel features of images. DSST Danelljan et al. (2017) uses translation filter and scale filter to deal with changes in location and scale. Because of Fourier transformation, most of these trackers can reach a very high speed.
Nowadays, deep learning has been widely used in computer vision, such as classification, detection and semantic segmentation. Many trackers based on deep learning have been designed to get a better precision. Some of them combine deep learning with CF. DLT Wang and Yeung is the first tracker to use deep learning. The structure is based on particle filter and it uses SDAE to extract features. HCF Ma et al. uses the features in different layers from a neural network and then applies CF to conduct tracking. MDNet Nam and Han is a novel tracker based on CNN which uses detection method to realize tracking. It samples

bounding box around the target location in the next frame and sends them to the network to find the one with the highest score, which is the target in the next frame. And the network will be fine-tuned when it does not work well any longer. TCNN Nam et al. (2016) is a tracker based on CNN and a tree structure. It builds a tree to evaluate the reliability of the model. GOTURN Held et al. is the fastest tracker based on CNN and it can achieve 165 fps. It uses the deep neural network to regress the bounding box in the next frame. But its target localization precision is poor. In a word, the high precision of a tracker based on CNN is obtained at the sacrifice of speed.

3. Fast Dynamic Convolutional Neural Networks

This section describes the structure of the network and how to train and test the proposed network. The convolution is operated on the whole image and RoIAlign is used to obtain fix-sized features. This strategy can speed up the convolution computation and maintain the localization accuracy in the meanwhile. Different branches have been added to the last layer in correspondence to different video sequences, and the preceding layers are shared layers to get common representations of targets through training. During the testing stage, the last layer is removed and a new one with only one branch is added to the network. Parameters in the fully-connected layers will be updated in a new strategy which will reduce the ineffective updates during tracking to adapt to the currently tracked targets.

3.1. Convolution on the Whole Image

In MDNet, there are 256 RoIs extracted from each frame and all of them are scaled to a fixed size. Then they will be sent to the convolutional layers and get fixed-size features which can then be delivered to the fully-connected layers. But it is a waste of time to do so. It contains a lot of redundancy calculation and the convolution operation has to repeat for many times which will definitely increase the time of process.
To speed up the convolution, a strategy of conducting convolution on the whole image is adopted and it only needs to do the convolutional operation for once. At the end of the convolution layers, a RoIPool or RoIAlign is used to extracted fixed-size features corresponding to the RoIs from the last convolution layer. The experiments show that RoIAlign is better than RoIPool in precision which we will discuss the details in section 4.

3.2. Network Architecture

The architecture of our network is illustrated in Fig. 1. The first three convolution layers are borrowed from the VGG-M Chatfield et al. (2014) network. The size of the filters in the first convolution layer is $7 \times 7 \times 3 \times 96$, which is followed by a Relu, a normalization Ioffe and Szegedy (2015) and a pooling layer. The size of the filters in the second convolution layer is $5 \times 5 \times 96 \times 256$, also followed by a Relu, a normalization and a pooling layer. The size in the third convolution layer is $3 \times 3 \times 256 \times 512$, followed by a Relu layer. Then there is a RoIAlign layer to extract fixed size features (here we set it to $3 \times 3 \times 512$) from the convolution layers. Next, there are two fully-connected layers with dropout (the rate is set to 0.5). The last fully-connected layer is just like the one in MDNet. Suppose we use k videos to train the network, the network will have k branches in the last layer and each of them uses two labels to represent the target and the background. The last layer is domain-specific layer and the others are shared layers. That means, when the network is trained with the $i^{\text{th}}(i = 1, 2, , k)$ video, we just update the parameters in i^{th} branch of the last fully-connected layer and the shared layers.

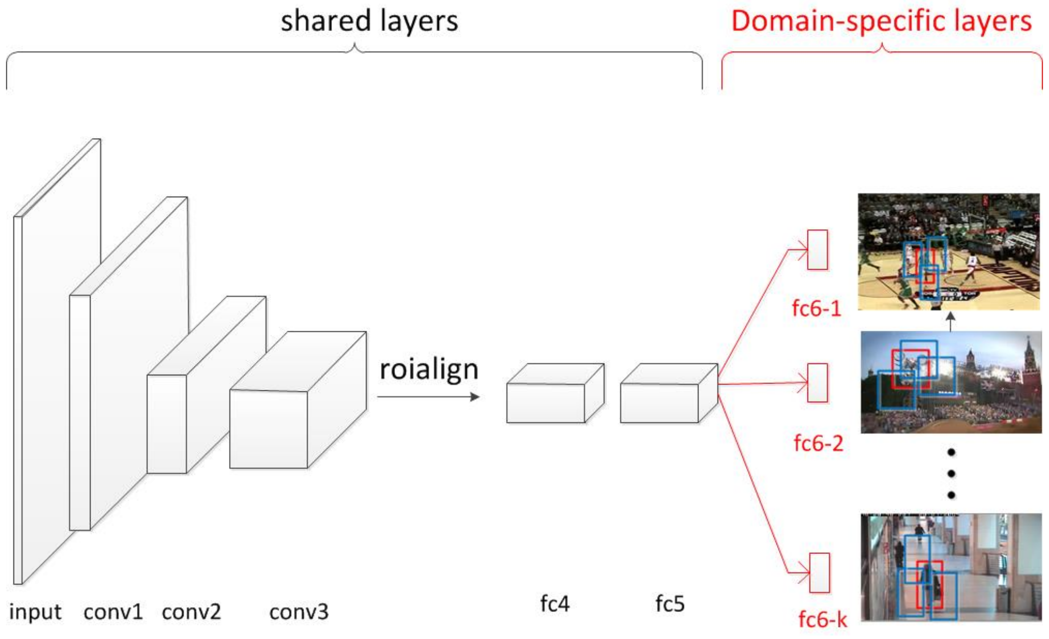


Figure 1: The architecture of fast dynamic convolutional neural network. Red and blue bounding boxes denote the positive and negative samples, respectively.

Most of the neural networks for classification and detection use Pool or RoIPool layer after the convolution layers. RoIPool can get fixed-size (e.g., 3×3) features from each RoI, which can then be sent to the following fully-connected layers. However, there is one major issue with RoIPool. After several convolution operations, the size and position of the RoIs might be float numbers, and we need to divide the RoIs into fixed-size regions (e.g., 3×3). The RoIPool rounds the float numbers to the nearest integers to fulfill the pooling. The localization precision may get lost in this operation.
RoIAlign He et al. (2017) is another way to get fixed-size(e.g., 3×3) features from each RoI. It can work better than RoIPool in theory. Meanwhile, RoIAlign does not spend much more time than RoIPool. The performance comparison of RoIPool and RoIAlign could be found in section 4. Different from RoIPool, RoIAlign keeps the float numbers in the operation. At the last step of RoIAlign, we use bilinear interpolation Jaderberg et al. to calculate n points in each bin and use the largest one to represent this bin(max-RoIAlign).
Fig. 2 shows an example of RoIAlign. The point $A(w, h)$ is what we need and its coordinates are float. The four points:$top\_left(x_{left}, y_{top})$,$top\_right(x_{right}, y_{top})$,$bottom\_left(x_{left}, y_{bottom})$ and $bottom\_right(x_{right}, y_{bottom})$ around it are the nearest integer points. The value of A could be calculated via the following equations:

$$
\begin{aligned}
&V_1 = (1-(w-x_{left})) \times (1-(h-y_{top})) \times Value_{top\_left}\\
&V_2 = (1-(x_{right}-w)) \times (1-(h-y_{top})) \times Value_{top\_right}\\
&V_3 = (1-(w-x_{left})) \times (1-(y_{bottom}-h)) \times Value_{bottom\_left}\\
&V_4 = (1-(x_{right}-w)) \times (1-(y_{bottom-h})) \times Value_{bottom\_right}\\
&Value_A = V_1+V_2+V_3+V_4
\end{aligned}
$$

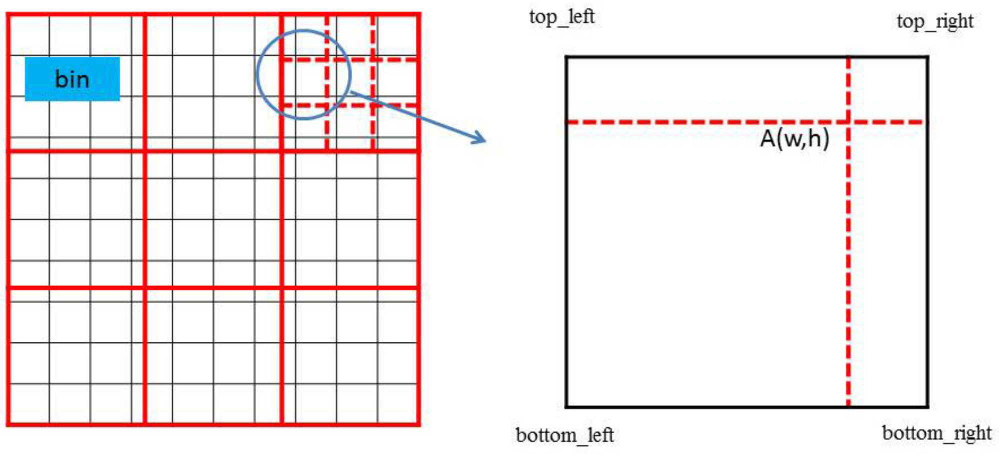


Figure 2: The schematic diagram of RoIAlign. Suppose there is a 10x10 pixels RoI (in black) and we want to get a 3x3 feature (in red). 2x2 points are selected in each bin. Because the locations of the points are float types, we use four integer points nearby and bilinear interpolation to get the value of this point. And the biggest one is used to represent this bin (max RoIAlign).

When it comes to back propagation, just like RoIPool, we just use the points which contribute to the network and propagate their errors. Suppose that point A is contributed to the network and its gradient is represented by $Value_{der}$, and the back propagation is like that:

$$
\begin{aligned}
&Value_{top\_left} = (1-(w-\lfloor w\rfloor)) \times (1-(h-\lfloor h\rfloor)) \times Value_{der}\\
&Value_{top\_right} = (1-(w-\lfloor w\rfloor)) \times (1-(\lceil h\rceil-h)) \times Value_{der}\\
&Value_{bottom\_left} = (1-(\lceil w\rceil-w)) \times (1-(h-\lfloor h\rfloor)) \times Value_{der}\\
&Value_{bottom\_right} = (1-(\lceil w\rceil-w)) \times (1-(\lceil h\rceil-h)) \times Value_{der}
\end{aligned}
$$

For each bin, 2×2 points are chosen and we use the largest one to represent this bin. The precision has dropped slightly if 1×1 point is used and 3×3 points which will increase the calculation indicates that there is almost no difference with 2×2. So 2×2 points in each bin is an acceptable choice.

A smaller network than those which usually used in the detection or classification is constructed in our study. The most serious issue of the deep network solution for image tracking is the low time efficiency. Considering that in the tracking scenario only binary classification is involved, so a small and fast network can satisfy the need of tracking task. Otherwise, with more convolution layers added, the localization information will get diminished over layers which would influence the tracking accuracy. Some other network structures are compared with our network and the results show the superiority of the proposed network. The details could be found in section 4.

3.3. Training Method

Softmax cross-entropy loss and SGD are used to update the network. When i^{th} video is used to train the network, i^{th} branch of the last layer and shared layers are working together and their parameters are updated at the same time. Other branches in the last layer are disabled and do not work. It was shown in Nam and Han that k branches could get better performance than only one branch in the last layer during training.
Samples are extracted around the target from each frame in a video. The samples whose IoU are larger than a threshold $t1$ are used as positive examples, and the ones with IoU less than a threshold $t2$ are used as negative examples. In our experiment, we set $t1$ to 0.7 and $t2$ to 0.5.
The three convolution layers have the same structure as in VGG-M network which have been pre-trained on ImageNet. The parameters in the fully-connected layers are initialized randomly which subject to Gaussian distribution. Because the convolution layers have been trained on ImageNet, a smaller learning rate is adopted from the beginning. We set the learning rate to 0.0001 and weight decay to 0.0005. It takes 100 iterations to get the local optimum.

3.4. Tracking Method

After the proposed network has been trained, we delete the last fully-connected layer and add a fully-connected layer with two outputs, representing the target and the background, respectively. The parameters in the last layer are initialized randomly which subject to Gaussian distribution. The three fully-connected layers will be updated while tracking, but the convolution layers will not be changed.
In the beginning, the positive and negative examples from the first frame are used to fine-tune the network. Then, we select samples from the current frame near the target and send the samples to the network. The sample with the highest score will be obtained as the target in this frame. If the score is higher than a threshold m, both positive and negative samples near the target in the current frame are collected for updating. When the highest score is less than the threshold m, we use samples collected from the previous frames to update the fully-connected layers. m is set to 0 in our experiments.
In MDNet, the network will be iterated 10 times if the highest score is less than the threshold m. But in our experiments, we find that a lot of iterations are useless and spend much time. So we define a loss threshold l. The iteration will stop if the loss of the network is less than l. It is a very simple strategy, but it is an effective one which will save a large amount of time. We use an example to explain this strategy and its consequence. The loss threshold l is set to 0.01.
We use the sequence of Ironman in OTB100. When the score is less than m, which is 0, the network should be fine-tuned. When the score is larger than m, positive and negative samples will extracted from this frame for the next fine-tuning. When the network is being

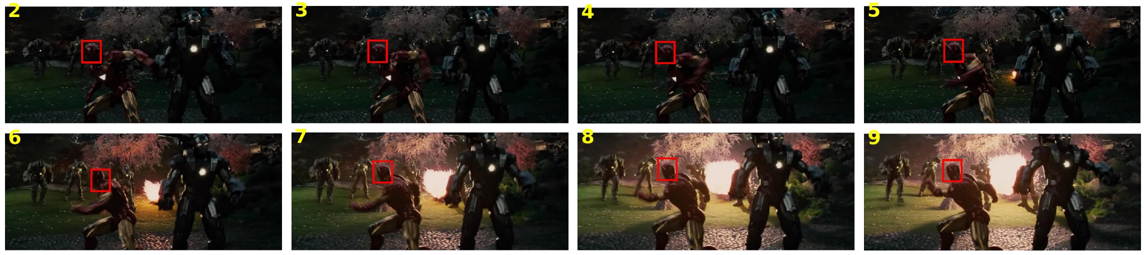


Figure 3: frame 2-9 in the sequence of Ironman

Table 1: the states of fine-tuning in the frame 2-9 of Ironman

frame	2	3	4	5	6	7	8	9
score	0.392	-1.443	3.199	1.444	-5.294	-3.340	-2.419	-1.221
loss	-	0.03	-	-	0.006	0.004	0.003	0.001
iteration	-	7	-	-	6	1	1	1

updated, it will iterated 10 times and will stop iteration if the loss is less than l, which is 0.01. We could see the results of frame 2-9 in Fig. 3 and know some parameters from Table. 1. In Table. 1, scores in frame 6-9 are less than the threshold m, so the network has to be updated. But the loss is very small in frame 6. The small loss means that the parameters in the network is been updated very well. In frame 7,8 and 9, we have to use the same positive and negative samples from frame 1,2,4 and 5(whose scores are larger than 0) to update the network, and because the loss is very small, these updates are ineffective and time consuming. So we use the loss threshold l to control the update and it will stop the fine-tuning when the loss is less than l.
We could see from the Fig. 3 that it can track the object very well even thought the scores in frame 6,7,8 and 9 are less than the threshold m. It means that m is not suitable for this sequence. But there are 100 sequences in OTB100 and we can not change the threshold in each sequence. So we could just use the loss threshold l to control the fine-tuning of the network.
The last row of Table. 1 is the numbers of iteration which the network needs. We can see that it only needs $7 + 6 + 1 + 1 + 1 = 16$ iterations in these frames. But in MDNet, the network always has to be updated 10 times and the total number of iteration is $10 \times 5 = 50$. We know that updating the network when tracking will have huge influence on the speed of the tracker. By this method, we can reduce a large amount of time and it will not affect the tracking result. Because the iterations we delete are useless and ineffective.

Moreover, hard negative mining Sung and Poggio (1998) and bounding box regression Nam et al. (2016) have been employed. Experiments Nam and Han show that both of them can improve the tracking performance. Hard negative mining is a trick to select effective samples. The negative samples will be sent to the network to get scores. And the samples with higher scores are taken as the hard negative samples. We use these hard negative samples instead of all the negative samples to update the fully-connected layers.

The bounding box regression is just like what they used in R-CNN Girshick et al. (2016). A simple linear regression model is trained to predict the target region using the features of the 3rd convolution layer. The training of the bounding box regression is only conducted with the first frame. The learning rate is set to 0.0005 in the first frame and it is set to 0.0015 for online updating.

4. Experiment

We evaluated our network on two datasets: object tracking benchmark (OTB) and VOT2016. Our code is implemented in MATLAB with matconvnet Vedaldi and Lenc. It runs at about 7 fps on Tesla p40 GPU.

4.1. Experiment on OTB

OTB Wu et al. (2015) is a popular tracking benchmark. Each of those vedios is labeled with attributes. The evaluation is based on two metrics: center location and bounding box overlap ratio. Three evaluation methods have been employed: OPE(one pass evaluation), TRE(temporal robustness evaluation) and SRE(spatial robustness evaluation). The proposed method has been compared with state of the art methods including CSK Nam et al. (2016), Struct Hare et al. (2016), CF2 Ma et al. and Staple Bertinetto et al.. To train the network, 58 videos from VOT2013 Kristan et al. (b), VOT2014 Liris (b) and VOT2015 Liris (a) have been employed, excluding the videos included in OTB100. The results of OTB100 are shown in Fig. 4. And some examples of the visual tracking results are shown in Fig. 5. From these results it could be concluded that superior or comparable performance has been obtained via the proposed method.
The comparison of MDNet and our network is in Table. 2. Our method is 7$\times$ faster than MDNet with only a small drop in precision.

4.2. Experiment on VOT2016

For further comparison, the algorithm has also been evaluated on the VOT2016 Kristan et al. (2016) which consists of 60 videos. VOT is a challenging competition organized for several years. The benchmark will re-initialize the targets when the tracker fails and evaluate module reports with both accuracy and robustness in terms of the bounding box overlap ratio and the number of failures, respectively. We compare our tracker with SRDCF Danelljan et al. (a),DPT Lukei et al. (2017),CCCT Chen et al.,DSST2014 Danelljan et al. (b),SO-DLT Wang et al. (2015b), MIL Babenko et al. (2011) and IVT Ross et al. (2008). 84 videos from OTB100 are used to train the network, which do not include the sequences in VOT2016. The results are shown in Fig. 6. As illustrated in Fig. 6 , our tracker shows great results both in accuracy and robustness. It also shows that our method rank 1st in these trackers according to EAO(Expected Average Overlap) criterion.

4.3. Comparisons of RoIAlign and RoIPool

Most of the networks using RoIPool from fast-RCNN to accelerate the convolution have obtained good performance on detection and classification task. However, for tracking tasks,

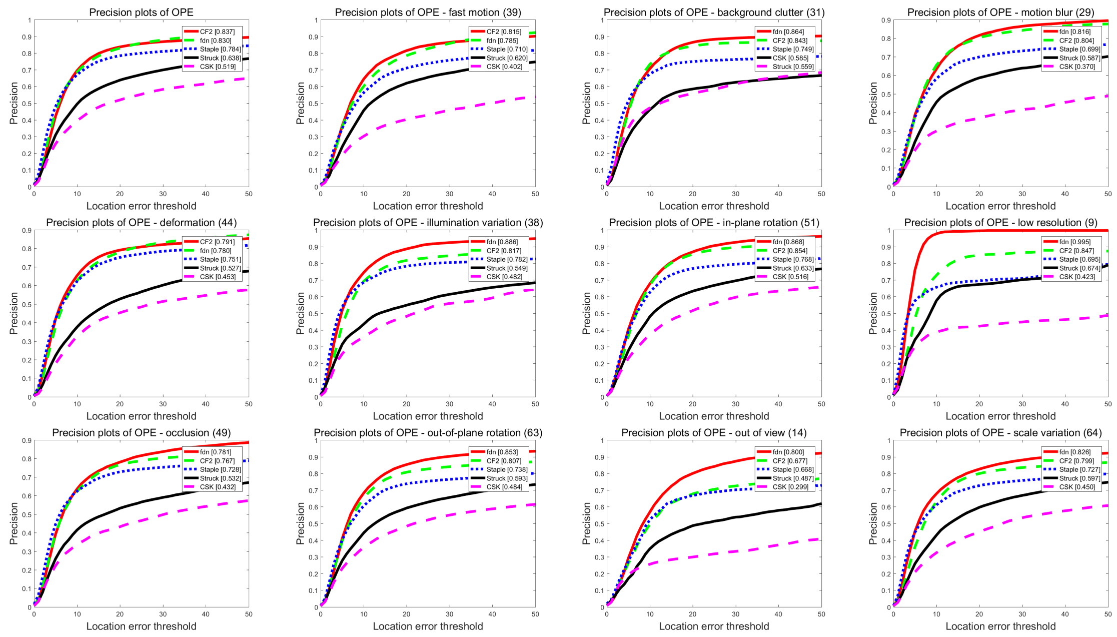


Figure 4: Results on OTB100

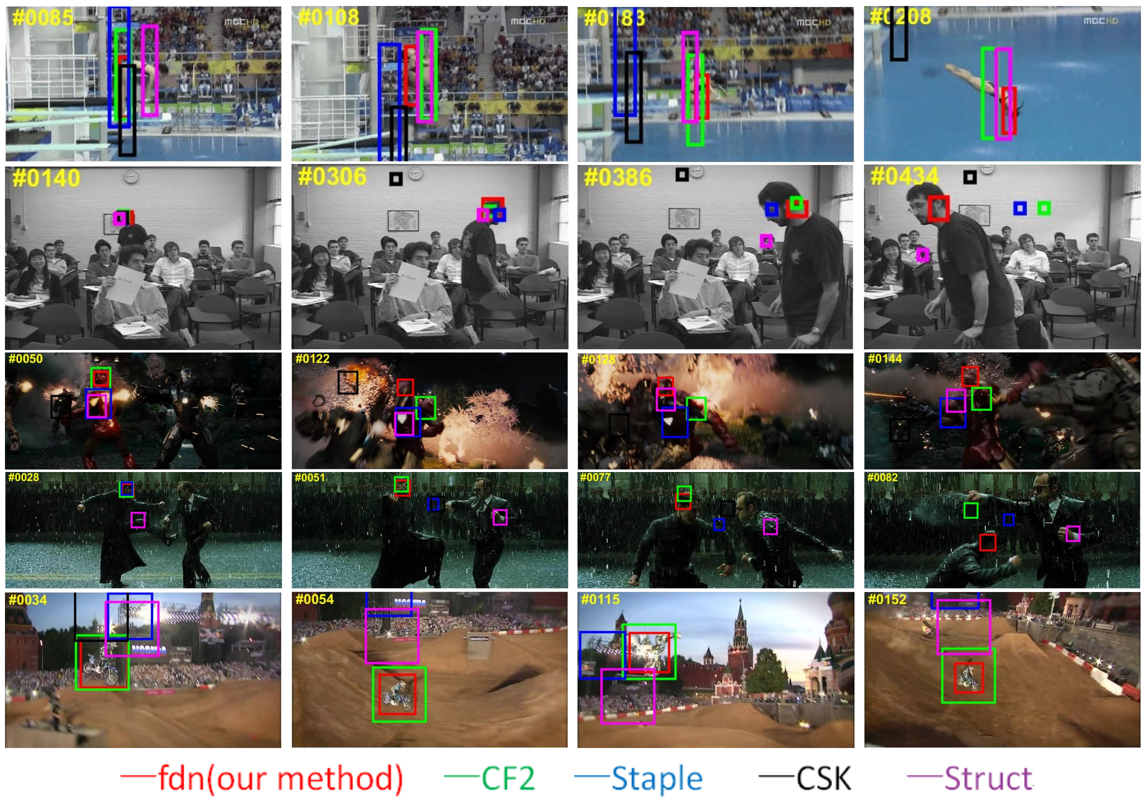
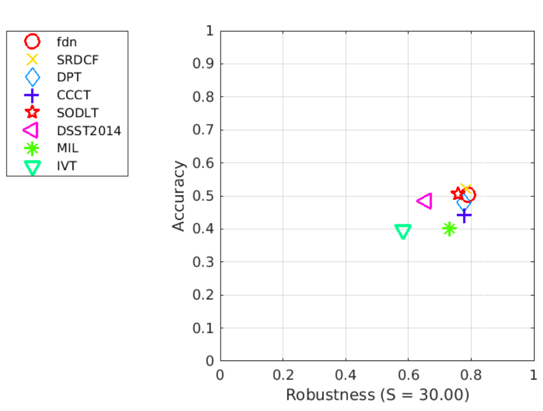
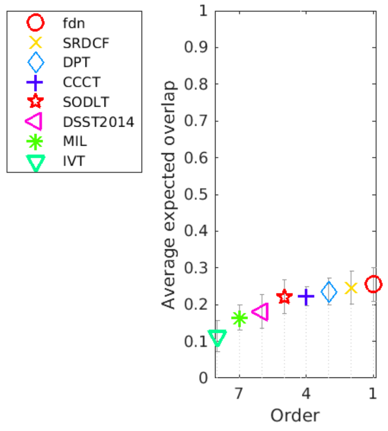


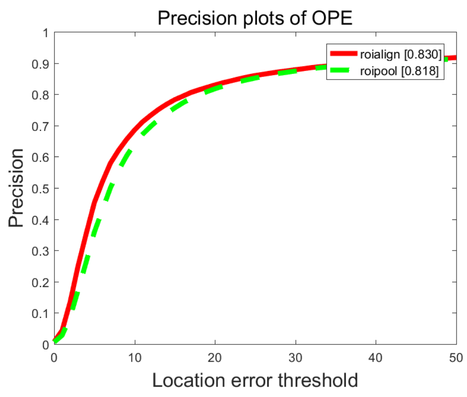
Figure 5: Comparison of some visual tracking examples





(a) VOT2016 accuracy-robustness plot. The better trackers are closer to the top-right corner. (b) VOT2016 ranking in terms of expected average overlap

Figure 6: Results on VOT2016



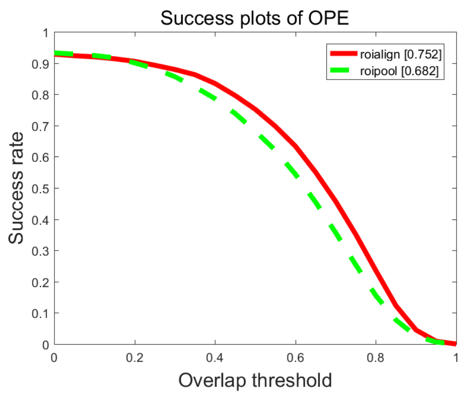


Figure 7: The results of RoIPool and RoIAlign on OTB100

the localization error will continue to be accumulated and enlarged along the working of the tracker. Even a small error of the current frame will affect the ones after it. So we use RoIAlign instead of RoIPool to improve the target localization precision. Fig. 7. shows the results of RoIAlign and RoIPool on OTB100.

4.4. Influence of the Number of Convolution or Fully-connected Layers

Considering we just use 3 convolution layers of the VGG-M network, and it has 5 convolution layers in the VGG-M network. More convolution layers may lead to better performance, so we use 5 convolution layers to build a network, and the rest part are kept the same with our previous method. The comparison resuts on OTB100 are given in Fig. 8. In addition, considering there are only two categories, there is no need to use more parameters in the fully-connected layers. Therefore, to evaluate the influence of the number of fully-connected layers, a network with just two fully-connected layers is constructed and the rest are the

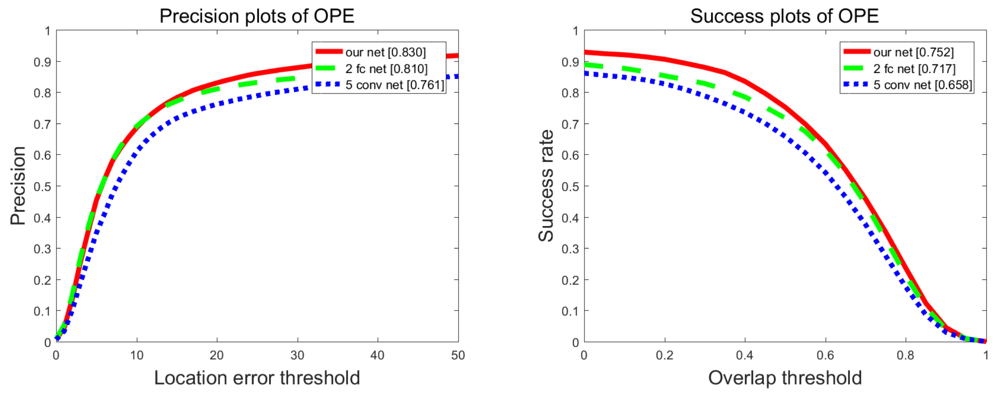


Figure 8: The results of our network,5 convolution network and 2 fully-connected network on OTB100

Table 2: comparison of MDNet and our method on OTB2013

	precision	success rate	speed
MDNet	0.948	0.708	1 fps
Our Method	0.916	0.667	7 fps

same. The results on OTB100 are also shown in Fig. 8. We can see that the network structure used in the proposed method obtains the best performance.
More convolution layers may lead to a better results in detection. But a deep CNN is less effective for precise target localization since the spatial information tends to be diluted as a network goes deeper Wang et al. (2015a). Localization information is very important in tracking. So it explain why more convolution may not result in a better precision.

4.5. Do We Need to Train the Network with Videos

We all know that convolution layers are in charge of extracting features and fully-connected layers are the classifiers to distinguish objects. The convolution layers in our network are trained on ImageNet and it is a good feature extractor. Maybe it does not need to be trained again and fine-tuning the fully-connected layers can accomplish tracking. So we just use first 3 convolution layers of VGG-M and put three randomly initialized fully-connected layers after them to build a network. The results on OTB100 are shown in Fig. 9. It demonstrates that the network still can work but not as well as the trained one.

5. Conclusions

A fast dynamic convolution neural network for visual tracking was proposed. To speed up the computation and maintain the accuracy in the meantime, a whole frame based convolution and RoIAlign has been employed. There are 3 convolution layers, 3 fully-

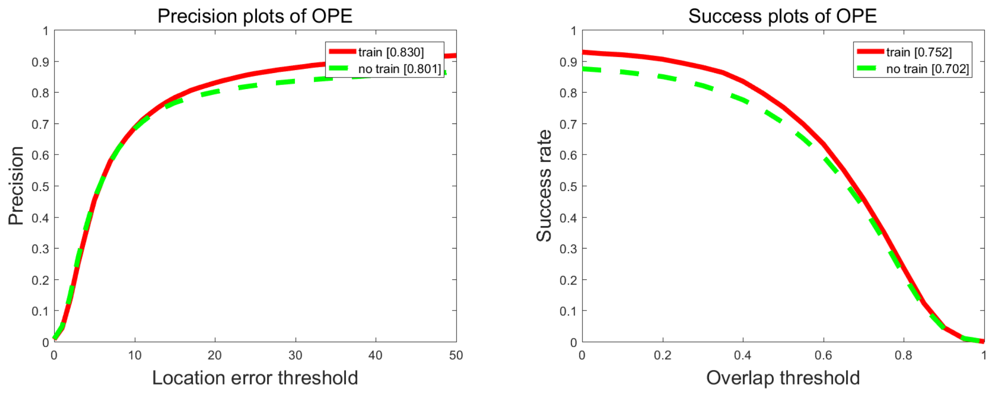


Figure 9: The results of train and no train network on OTB100

connected layers and a RoIAlign layer between them. The RoIAlign layer will accelerate the convolution and performs better than RoIPool with higher target localization precision. The last fully-connected layer has k branches which correspond to k videos during training. During tracking, the last fully-connected layer is deleted and a new one which is randomly initialized is added. The first frame is used to train the fully-connected layers and update the parameters. A new strategy of fine-tuning the fully-connected layers is used to accelerate the update when tracking. Finally about 7 fps on GPU is reached and outstanding performance on two tracking benchmarks, OTB and VOT2016, has been obtained.

References

Boris Babenko, Ming-Hsuan Yang, and Serge Belongie. Robust object tracking with online multiple instance learning. *IEEE transactions on pattern analysis and machine intelligence*, 33(8):1619–1632 IEEE.

Luca Bertinetto, Jack Valmadre, Stuart Golodetz, Ondrej Miksik, and Philip H. S. Torr. Staple: Complementary learners for real-time tracking. pages 1401–1409. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

David S. Bolme, J. Ross Beveridge, Bruce A. Draper, and Yui Man Lui. Visual object tracking using adaptive correlation filters. pages 2544–2550 Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on.

Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*, 2014.

Dapeng Chen, Zejian Yuan, Yang Wu, Geng Zhang, and Nanning Zheng. Constructing adaptive complex cells for robust visual tracking. pages 1113–1120. Proceedings of the IEEE International Conference on Computer Vision.

Martin Danelljan, Gustav Hager, Fahad Shahbaz Khan, and Michael Felsberg. Learning spatially regularized correlation filters for visual tracking. pages 4310–4318, a. Proceedings of the IEEE International Conference on Computer Vision.

Martin Danelljan, Gustav Hger, Fahad Khan, and Michael Felsberg. Accurate scale estimation for robust visual tracking. BMVA Press, b. British Machine Vision Conference, Nottingham, September 1-5, 2014.

Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. Eco: Efficient convolution operators for tracking. *arXiv preprint arXiv:1611.09224*, 2016.

Martin Danelljan, Gustav Hger, Fahad Shahbaz Khan, and Michael Felsberg. Discriminative scale space tracking. *IEEE transactions on pattern analysis and machine intelligence*, 39 (8):1561–1575 IEEE.

Ross Girshick. Fast r-cnn. pages 1440–1448. Proceedings of the IEEE international conference on computer vision.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. pages 580–587. Proceedings of the IEEE conference on computer vision and pattern recognition.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Region-based convolutional networks for accurate object detection and segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 38(1):142–158 IEEE.

Sam Hare, Stuart Golodetz, Amir Saffari, Vibhav Vineet, Ming-Ming Cheng, Stephen L. Hicks, and Philip H. S. Torr. Struck: Structured output tracking with kernels. *IEEE transactions on pattern analysis and machine intelligence*, 38(10):2096–2109 IEEE.

Kaiming He, Georgia Gkioxari, Piotr Dollr, and Ross Girshick. Mask r-cnn. *arXiv preprint arXiv:1703.06870*, 2017.

David Held, Sebastian Thrun, and Silvio Savarese. Learning to track at 100 fps with deep regression networks. pages 749–765. Springer. European Conference on Computer Vision.

Joo F. Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. Exploiting the circulant structure of tracking-by-detection with kernels. pages 702–715. Springer. European conference on computer vision.

Joo F. Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):583–596 IEEE.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Max Jaderberg, Karen Simonyan, and Andrew Zisserman. Spatial transformer networks. pages 2017–2025. Advances in neural information processing systems.

Matej Kristan, Roman Pflugfelder, Ale Leonardis, Jiri Matas, Fatih Porikli, Luka Cehovin, Georg Nebehay, Gustavo Fernandez, Toma Vojir, and Adam Gatt. The visual object tracking vot2013 challenge results. pages 98–111 Computer Vision Workshops (ICCVW), 2013 IEEE International Conference on.

Matej Kristan, Roman Pflugfelder, Ale Leonardis, Jiri Matas, Fatih Porikli, Luka Cehovin, Georg Nebehay, Gustavo Fernandez, Toma Vojir, and Adam Gatt. The visual object tracking vot2013 challenge results. pages 98–111 Computer Vision Workshops (ICCVW), 2013 IEEE International Conference on.

Matej Kristan, Ale Leonardis, Jii Matas, Michael Felsberg, Roman Pflugfelder, Luka ehovin, Tom Vojr, Gustav Hger, Alan Lukei, and Gustavo Fernndez. *The Visual Object Tracking VOT2016 Challenge Results.* Springer International Publishing, 2016.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. pages 1097–1105. Advances in neural information processing systems.

F. Liris. The visual object tracking vot2015 challenge results. a.

France Liris. The visual object tracking vot2014 challenge results. b.

Alan Lukei, Luka ehovin Zajc, and Matej Kristan. Deformable parts correlation filters for robust visual tracking. 2017. IEEE.

Chao Ma, Jia-Bin Huang, Xiaokang Yang, and Ming-Hsuan Yang. Hierarchical convolutional features for visual tracking. pages 3074–3082. Proceedings of the IEEE International Conference on Computer Vision.

Hyeonseob Nam and Bohyung Han. Learning multi-domain convolutional neural networks for visual tracking. pages 4293–4302. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Hyeonseob Nam, Mooyeol Baek, and Bohyung Han. Modeling and propagating cnns in a tree structure for visual tracking. *arXiv preprint arXiv:1608.07242*, 2016.

Yuankai Qi, Shengping Zhang, Lei Qin, Hongxun Yao, Qingming Huang, Jongwoo Lim, and Ming-Hsuan Yang. Hedged deep tracking. pages 4303–4311. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. pages 91–99. Advances in neural information processing systems.

David A. Ross, Jongwoo Lim, Ruei-Sung Lin, and Ming-Hsuan Yang. Incremental learning for robust visual tracking. *International journal of computer vision*, 77(1-3):125–141 Springer.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

K. K. Sung and Tomaso Poggio. Example-based learning for view-based human face detection. *IEEE Transactions on pattern analysis and machine intelligence*, 20(1):39–51 IEEE.

Jack Valmadre, Luca Bertinetto, Joo F. Henriques, Andrea Vedaldi, and Philip H. S. Torr. End-to-end representation learning for correlation filter based tracking. *arXiv preprint arXiv:1704.06036*, 2017.

Andrea Vedaldi and Karel Lenc. Matconvnet: Convolutional neural networks for matlab. pages 689–692 Proceedings of the 23rd ACM international conference on Multimedia.

Lijun Wang, Wanli Ouyang, Xiaogang Wang, and Huchuan Lu. Visual tracking with fully convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 3119–3127, 2015a.

Mengmeng Wang, Yong Liu, and Zeyi Huang. Large margin object tracking with circulant feature maps. *arXiv preprint arXiv:1703.05020*, 2017.

Naiyan Wang and Dit-Yan Yeung. Learning a deep compact image representation for visual tracking. pages 809–817. Advances in neural information processing systems.

Naiyan Wang, Siyi Li, Abhinav Gupta, and Dit-Yan Yeung. Transferring rich feature hierarchies for robust visual tracking. *arXiv preprint arXiv:1501.04587*, 2015b.

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

Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1834–1848 IEEE.