FasterMDNet: Learning Model Adaptation by RNN in Tracking-by-Detection based Visual Tracking

Hung-Wei Hsu, Jian-Jiun Ding
DISP Lab, National Taiwan University, Taipei, Taiwan
E-mail: r05942039@ntu.edu.tw, jjding@ntu.edu.tw Tel: +886-2-3366-3662

Abstract—Recently in VOT competitions, trackers based on tracking-by-detection and deep neural network discriminators reached impressive accuracy. However, these trackers require time-consuming model adaptation methods like online learning to handle target appearance changes, which tremendously increases the time complexity and becomes an obstruction in real-world applications for these tracking algorithms. In this paper, we propose an efficient RNN-based model adaptation method which extremely decreases the time complexity of trackers. The proposed model learns the relations of relative model change in RNN training and predicts the score and the model adaptation state at the same time in testing, which nearly removes the finetuning time in the cost of additional RNN training. The proposed method is applicable to any tracker based on neural network discriminator. The RNN branch can be further designed with more complicated model under the condition of enough training videos. We apply the proposed algorithm to MDNet and create a new tracker: Faster-MDNet. According to the experiment, using our method can nearly remove the time of finetuning and reduce the bottleneck of time-complexity down to the prediction time.

I. INTRODUCTION

Visual tracking is a challenging task in computer vision due to versatile factors of changes of the aimed target and the quality of the video including illumination variations, scale changes, occlusions, motion blur, target deformations, background clutters, coexistence of similar objects, etc. In the task of visual tracking, the target is provided by a bounding box in the first frame of the video and the tracker predicts the location and the size of the target in the following frames.

Recently, trackers based on tracking-by-detection and machine learning techniques have made impressive results in visual tracking [8]. Tracking by detection splits the task of visual tracking into two parts: bounding-box proposals and evaluating proposals. In the first stage, bounding-boxes are sampled from the current frame. In the second stage, each proposal is classified as the target or not. The prediction accuracy of trackers based on tracking-by-detection lies primarily on a binary classification problem. In VOT 2016 [8], the trackers adopting convolutional neural network classifiers [1, 2] have the highest accuracy among all tracking-by-detection based trackers and TCNN [2] is ranked as the second place among all trackers.

However, different from classic binary classification problem where the data distribution can be predicted by training data, visual tracking has the data distribution provided online. That is, the appearance of the target is unknown until the ground-truth bounding-box enclosing the target in the first frame is provided. In addition, the data distribution, i.e. the target appearance, will change through different video frames. Therefore, trackers based on tracking-by-detection require to update the model to adapt itself to the unexpected change of target appearance. Among tracking-by-detection trackers with convolutional neural network as the binary classifier, online learning [1] is a common method to update the model. These trackers will online collect appropriate positive and negative samples and finetune the network whenever the model adaptation is required, e.g. prediction confidence is too low or a fixed time duration is passed. Yet, since neural network is designed to spend most time on training to learn the data distribution and predict relatively in a short time, the online learning process will create a bottleneck on time-complexity and render the tracking algorithm away from applications.

For example, one long-term update in MDNet[1] takes around 10-15 times the time of prediction in one frame. Even though the long-term update occurs every ten frames in the settings of MDNet, this finetuning process has at least doubled the overall temporal cost without consideration of the time of short-term update and collecting training samples.

In this paper, we develop a model adaptation algorithm that utilizes recurrent neural network to learn how to update the model from online learning. Furthermore, we reuse the
candidate bounding-boxes for predictions to update the model, which removes the time of collecting extra finetuning samples. Our RNN-based method predicts and updates the model at the same time, thus reducing the temporal cost of model adaptation to nearly the same as prediction time. Proposed algorithm is a general idea that can be applied onto any trackers based on tracking-by-detection with neural network classifier. The overall algorithm can be treated as an addition of the RNN branch to the original tracker. The RNN branch can be designed individually without changing the prediction branch of the tracker.

Our contributions can be divided into 3 points:

1) temporal cost: Proposed RNN-based model adaptation method has extremely decreased the overall temporal cost down to the prediction time. The little cost of accuracy can be further decreased with more training videos and more complicated RNN model.

2) general idea of RNN-finetuning: Proposed method is a general idea that can aggregate RNN with any trackers based on tracking by detection with neural network discriminator, which largely increases the practical importance of tracking-by-detection technique.

3) a big step toward end-to-end tracking-by-detection based tracker: With proposed RNN-based mode adaptation, it’s a big step toward end-to-end tracking-by-detection tracker. For example, an end-to-end tracking-by-detection based tracker can be composed of region proposal network used in the area of object detection, a tracking-by-detection based tracker and proposed RNN model adaptation method.

II. RELATED WORK

A. Visual Tracking Algorithms

Visual tracking is a fundamental problem in computer vision. Most trackers can be divided into two genres: the generative model and the discriminative model. The generative model attempts to describe the appearance of the targets and search for the best-fitting regions in frames [13, 14]. The discriminative model aims to transfer visual tracking into a problem of separation of foreground and background where the foreground is the target model. Recently, correlation filters [4] and CNNs [1, 2, 3] have gained a large attention in visual tracking. Among the top-10 trackers of VOT2016, 4 trackers were derived from CNNs, 4 trackers were variations of correlation filters. Rank-1 tracker C-COT[4] is based on correlation filter and utilizes CNN features for strong representations.

B. Tracking by Detection

In the field of object detection, many works [5] predict the result by classification and regression of candidate bounding boxes. Tracking by detection adopts the idea and predict the tracking result by proposing candidate bounding-boxes and evaluating each candidate. The major difference between detection and tracking-by-detection [1, 2, 3] is that tracking by detection requires to update the model to adapt to the change of target appearance.

C. Recurrent Neural Network on Visual Tracking

RNN is a neural network modeling designed to process temporal-spatia information and has gained huge attentions in language processing field like language understanding, language generation, captioning, etc. Recently, [10, 11, 12] has addressed temporal-spatial information in object tracking using RNN. However, they focus on artificially generated sequences and synthesized data. [9] is a tracker which uses deep convolutional neural network and object detection framework [5] to extract image features and feed them into LSTM to predict the target. However, although their experiments have shown a successful result on selected videos of natural images, [9] does not finetune the network for the target object in the first frame of the video, which restricts the tracker to track only the saliency or pre-trained objects in one video rather than the object enclosed by the bounding-box of the first frame. In cases like tracking a ball when a group of people are playing basketball, [9] may not be appropriate.

Proposed RNN-based model adaptation separates the prediction and the model adaptation as two branches. The prediction branch will be finetuned on the first frame, so it can handle unexpected objects and track different objects in the same video. The model adaptation branch learns the conditional distribution of adaptation states given proposal and scores in pre-training, which is independent of the appearance of the tracked target. Therefore, our model has the capability of tracking unexpected objects.
III. RNN-BASED MODEL ADAPTATION

A. MDNet

MDNet is a tracker based on tracking-by-detection and VGG-M network [15]. In MDNet, VGG-M is first pre-trained on ImageNet dataset. Then, the network is transferred to video domain by multi-domain learning. Before tracking, the last layer will be randomly initialized and finetuned on the first frame. In tracking, positive and negative samples are collected from frames with confident predictions. A short-term update is executed when the prediction score is too low, which takes around half the time of prediction in one frame. A long-term update is executed every ten frame, which is around 10-15 times the time of prediction in one frame. The details can be referred to the paper of MDNet [1].

The overall temporal cost of MDNet in prediction including the time of (1) prediction (2) collecting finetuning samples (3) short-term finetuning (4) long-term finetuning. (1)(2) occur nearly in every frame. (3) occurs only when the confidence is two low. (4) occurs in every 10 frames. (1), (2), and (3) can be executed in a very short time, while (4) can be 10-15 times longer than others. The proposed algorithm completely replaces (4) and (2) and the additional temporal cost is smaller than that of (1).

B. Faster-MDNet

In this section, we explain by example of MDNet how we can aggregate a tracker with proposed RNN-based model adaptation.

As in figure 1, the original MDNet has only one branch to predict score. We aggregate the original MDNet with a new branch which is a copy of the fc layers. We name the new branch as model adaptation branch and the original branch as prediction branch.

For each proposal, the prediction branch produces a score and the model adaptation branch produces a model adaptation state. The model adaptation state is the outputs of fc layers in model adaptation branch.

As in figure 2, since there are a bunch of proposals, a number of model adaptation states and scores will be produced. To emphasize the higher-scored proposals, we weight the model adaptation states with their scores and take the mean to produce the final model adaptation state.

\[ S_i(t) = \frac{1}{N} \sum_{n=1}^{N} w_n S_{n,i}(t) \]  

(1)

At timestep \( t \), \( w_n \) is the score of \( n^{th} \) proposal, \( S_{n,i}(t) \) is the output of \( n^{th} \) proposal at fc\(_i\) layer, \( S_i(t) \) is the final model adaptation state at fc\(_i\) layer.

In prediction at timestep \( t + 1 \), the model adaptation state is treated as the hidden state of RNN and added directly onto fc4, fc5, and fc6 in prediction branch. That is,

\[ X_{n,4}(t) = \sigma(W_{n,pred}X_{n,3}(t) + S_i(t - 1)) \]  

(2)

\[ X_{n,5}(t) = \sigma(W_{n,pred}X_{n,4}(t) + S_5(t - 1)) \]  

(3)

\[ X_{n,6}(t) = \sigma(W_{n,pred}X_{n,5}(t) + S_6(t - 1)) \]  

(4)

where \( X_{n,i}(t) \) is the features of proposal \( n \) at fc\(_i\) layer in prediction branch at timestep \( t \), \( W_{n,pred} \) is the weights at fc\(_i\) layer in prediction branch, \( \sigma \) is the non-linearity.

In this paper, we treat the final model adaptation state as the mean of score-weighted model adaptation state of all proposals. Yet, the relation among the final state, the score and the state of each proposal can be more complicated, e.g. a model of neural network. That is,

\[ S_i(t) = f(S_{i,1}(t), S_{i,2}(t), ..., S_{i,N}(t), w_1, w_2, ..., w_N) \]  

(5)

Plus, the RNN state can have more complicated relations with the prediction branch in the next step. For example, we can use LSTM cells to handle those states rather than direct summation. That is,

\[ X_{n,i}(t) = g(S_i(t - 1), X_{n,i-1}(t)) \]  

(6)

However, the number of training videos would be an issue toward more complicated model.

C. Three-Phase Training

With this aggregation, we can train the prediction branch and the model adaptation branch together in the following way. We divide the overall training into three phases:

1) Pre-Training on ImageNet: In this phase, we pre-train the VGG-M network on multi-label classification of ImageNet dataset.

2) Multi-Domain Learning on Frames: In this phase, we train the VGG-M network by multi-domain learning as in MDNet. This phase is to transfer the model from multi-label classification to the domain of visual tracking.

3) Model Adaptation Learning on Videos: In this phase, we first create the model adaptation branch by copying both the structure and the weights from fc layers of the network pre-trained in phase 1 and phase 2 as mentioned in section III.B. Second, for each video, we randomly initialize the fc6 layer in prediction branch and prepare a zero state as our initial model adaptation state. Third, we finetune the prediction branch by extracting positive and negative samples from the first frame. Finally, we unroll our RNN by max-numstep \( M(M = 40 \text{ in our experiment}) \), and in each training batch, we extracts positive and negative proposals from consecutive M frames as our training data to train the unrolled RNN. The final model adaptation state will be fed into next training batch as the initial state adaptation state.

D. Prediction

In prediction, we randomly initialize the last layer in prediction branch and prepare a zero-state as the initial model adaptation state. Then we finetune the prediction branch on the first frame. In the following frames, the model adaptation is completed primarily by the model adaptation branch.
TABLE I
THE AVERAGE ELAPSED TIME ON FASTER-MDNET AND MDNET.

<table>
<thead>
<tr>
<th>Task at one frame</th>
<th>elapsed time (second)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Faster-MDNet: predict scores of all proposals and</td>
<td>~0.8</td>
</tr>
<tr>
<td>generate the final model adaptation state</td>
<td></td>
</tr>
<tr>
<td>Faster-MDNet: predict scores of all proposals and</td>
<td>~1.3</td>
</tr>
<tr>
<td>generate the final model adaptation state + short-term</td>
<td></td>
</tr>
<tr>
<td>finetuning</td>
<td></td>
</tr>
<tr>
<td>MDNet (prediction and long-term finetuning)</td>
<td>~11.9</td>
</tr>
</tbody>
</table>

TABLE II
SUMMARY OF AVERAGE OVERLAP SCORES (AOS) RESULTS.

<table>
<thead>
<tr>
<th>Sequence</th>
<th>ROLO +SORT</th>
<th>MDNet</th>
<th>Faster-MDNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human2</td>
<td>0.545</td>
<td>0.636</td>
<td>0.728</td>
</tr>
<tr>
<td>Gymn</td>
<td>0.599</td>
<td>0.460</td>
<td>0.563</td>
</tr>
<tr>
<td>Human8</td>
<td>0.364</td>
<td>0.416</td>
<td>0.628</td>
</tr>
<tr>
<td>Skater</td>
<td>0.618</td>
<td>0.283</td>
<td>0.657</td>
</tr>
<tr>
<td>SUV</td>
<td>0.627</td>
<td>0.455</td>
<td>0.684</td>
</tr>
<tr>
<td>Dancer2</td>
<td>0.627</td>
<td>0.201</td>
<td>0.746</td>
</tr>
<tr>
<td>Singer2</td>
<td>0.588</td>
<td>0.400</td>
<td>0.676</td>
</tr>
<tr>
<td>Woman</td>
<td>0.649</td>
<td>0.358</td>
<td>0.781</td>
</tr>
<tr>
<td>David3</td>
<td>0.622</td>
<td>0.224</td>
<td>0.708</td>
</tr>
<tr>
<td>Dancer</td>
<td>0.755</td>
<td>0.551</td>
<td>0.725</td>
</tr>
<tr>
<td>Human7</td>
<td>0.456</td>
<td>0.291</td>
<td>0.596</td>
</tr>
<tr>
<td>Bird1</td>
<td>0.362</td>
<td>0.048</td>
<td>0.403</td>
</tr>
<tr>
<td>CarDark</td>
<td>0.674</td>
<td>0.211</td>
<td>0.825</td>
</tr>
<tr>
<td>Couple</td>
<td>0.464</td>
<td>0.204</td>
<td>0.477</td>
</tr>
<tr>
<td>Skating1</td>
<td>0.572</td>
<td>0.443</td>
<td>0.551</td>
</tr>
<tr>
<td>Singer1</td>
<td>0.653</td>
<td>0.332</td>
<td>0.404</td>
</tr>
<tr>
<td>BlurCar3</td>
<td>0.539</td>
<td>0.191</td>
<td>0.801</td>
</tr>
<tr>
<td>Girtl2</td>
<td>0.517</td>
<td>0.337</td>
<td>0.708</td>
</tr>
<tr>
<td>Average</td>
<td>0.506</td>
<td>0.336</td>
<td>0.648</td>
</tr>
</tbody>
</table>

IV. EXPERIMENT RESULTS

A. Implementation Details

The experiment is implemented in Python2.7 and TensorFlow r1.0.0 on Ubuntu 14.04 LTS, Intel Core i7-6700K @ 4.00GHz x8, NVIDIA GeForce GTX TITAN X.

To reduce the influence of implementation language , we re-implement MDNet on python and tensorflow1. This python-version MDNet is faster than the original MDNet in MATLAB.

The ImageNet-pretrained VGG-M network is downloaded from caffe-model zoo. We trained the network on VOT 2013, 2014 and 2015 and tested on OTB dataset[6]. All the settings of phase2 training are the same as in MDNet. In phase3 training, we trained for 10 epochs. To prevent gradient explosion, we clipped the gradient by global norm with clip norm 1. The overall training, excluding ImageNet pretraining, takes around one week to complete.

B. Short-Term Finetuning

To increase the robustness of Faster-MDNet, we preserve the short-term finetuning of MDNet which takes only half the time of prediction. The temporal cost of short-term finetuning is negligible compared with the time of the replaced long-term finetuning. In most of time, RNN model update can handle the appearance change, short-term finetuning occurs only when the target has abrupt appearance change.

C. Results

1) Elapsed Time: Table I is the result of elapsed time. We compared Faster-MDNet with MDNet on testing time. It can be shown that the elapsed time of proposed Faster-MDNet is nearly the same as the prediction time in MDNet. The time saved from long-term finetuning is around 10 seconds.

2) Accuracy: Table II shows the average overlap scores compared with ROLO [9], YOLO[5]+SORT, MDNet[1] and proposed Faster-MDNet.

CONCLUSIONS

We propose an RNN-based model adaptation method to extremely decrease the time of finetuning. The method can be generalized to any trackers based on tracking-by-detection and deep classifier. An experiment on MDNet: Faster-MDNet is conducted, which shows a large decrease of time complexity. The proposed method is a huge step toward an end-to-end tracking-by-detection deep-tracker.

ACKNOWLEDGEMENTS

We are strongly grateful for Qualcomm Inc. for their assistance, discussion, and recommendations on this research.

REFERENCES


1https://github.com/HungWei-Andy/PyMDNet