Deep Tracking: Biologically Inspired Tracking with Deep Convolutional Networks

Si Chen
The George Washington University
sichen@gwmail.gwu.edu

Meera Hahn
Emory University
mhahn7@emory.edu

Mentor: Afshin Dehghan
University of Central Florida, Center for Research in Computer Vision
adehghan@cs.ucf.edu

Abstract

This paper discusses the problem of tracking from a deep learning approach. This experiment takes cues from how the brain is modeled to create deep convolutional networks that mimic how the human brain tracks objects. By using optical flow and deep networks to implement a dual appearance and motion stream, our tracker outperforms current state of the art methods.

1 Introduction

Tracking is a crucial problem in computer vision that has been addressed by a variety of methods that have been unable to cope with changes in videos such as illumination, occlusion, and scale. Deep tracking has been shown to work effectively for other problems in computer vision such as number recognition and object detection; we decided to investigate the problem of tracking from a deep learning approach to see if it would be a more effective and precise means of tracking that would be able to adapt to changes in the target and background of a video.

The current state of the art trackers have been able to deal with a few specific classical challenges, such as scale or illumination, but none of the trackers are well equipped to deal with the variety of challenges that are possible in a given video. Handcrafted features such as color histogram, HOG, and SIFT are not universally suited to tracking in every type of video. By training a Deep Tracker (DT), which would use previous frames from a video to enable itself to track future frames, rather than a generic classifier, we hope to improve tracking results.

Using a deep architecture allows us to represent an ever-changing target with few parameters[1]. Deep learning techniques enable learning feature hierarchies to understand multiple levels of representation[2]. Using a deep neural network allows for better object detection[3] and in combination with tracking the motion, or temporal stream, we can cover the two aspects of tracking—motion and appearance.

2 Network Structures

We tested different types of neural networks to better understand the strengths and weaknesses behind different approaches. For this section, let the target be the object being tracked (the foreground) and let the background be anything other than the target in the video frame.

2.1 Commonalties

The first step in our tracking algorithm was to collect positive and negative training samples from our video sequence. A positive sample contained the target and had a label of 1. A negative sample contained less than 80% of the target and had a label of 0. We were given the bounding box of the target from the first video frame's annotated ground truth. The ground truth included the center of the annotated

bounding box containing the target as well as the width and height of the bounding box. For the next N frames, a simple tracker would track the location of the target. In these N frames, the simple tracker does not change the width and height of the bounding box.

Using these N locations, we collected labeled positive and negative training samples through data augmentation. By permutating the bounding boxes through rotations and shifting several pixels to each side, we collected 25 positive samples and 50 negative samples per frame. In the majority of our experiments we set N = 4 and collected a total of 400 training samples.

This method worked for cases where the target did not experience a large deformation or illumination change in the first 4 frames. In these cases, we lowered N and received better results. However, lowering N results in a lower total yield of training samples, decreasing accuracy.

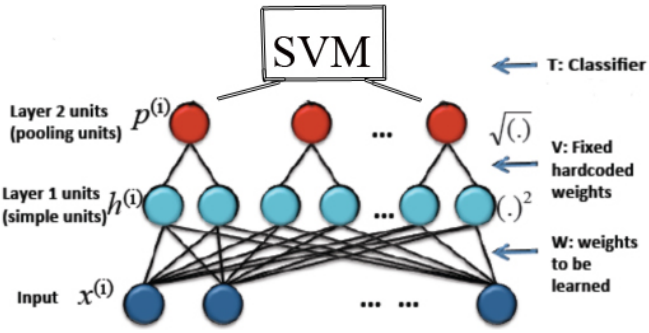


Blue: Annotated frame
Yellow: Positive & negative training samples

To collect testing samples for each frame, we took the location of the target in the previous frame and randomly choose test samples around that location using Gaussian weights. This made it so that test samples near the last location were more likely to be chosen than test samples further away from the last location.

2.2 Auto-encoder and SVM

Our first tracking algorithm used auto-encoders to extract features from the target and the background. An autoencoder is a feed-forward neural network that uses unsupervised learning only on the basis of an input vector $X^{(t)4}$. The autoencoder used in our algorithm had three hidden layers. These hidden layers would extract the important information from the output of the previous layer. The algorithm took the output of the third hidden layer and passed it to an SVM classifier.



The features were extracted from the testing samples via the autoencoder and then, the SVM output a confidence score which was a percentage of how likely the test image should be given a label of 1. In addition to the SVM classifier, a motion vector was calculated and applied via a Gaussian model to the bounding boxes being tested. Therefore, bounding boxes that were closer to the motion vector's prediction were given more weight.

2.3 CNN with pre-training and SVM

Our second tracking algorithm used a convolutional neural network (CNN) inspired by Krizhevsky's network[5]. The network had five convolutional layers and two fully connected layers. Additionally there was a max-pooling layer in between each convolution and a softmax regression loss.

We obtained a model of the network that was pre-trained on the large ImageNET[6] dataset. Our algorithm passed each of the labeled training images through the network and extracted the feature vector from the 2nd fully connected layer. We passed these feature vectors to an SVM classifier for training. In testing we used the same method of running the images through the network, extracting the feature vector and passing it to the SVM for classification.

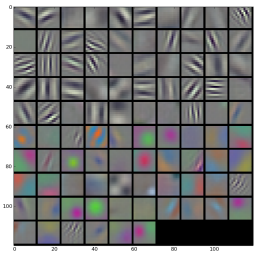
2.4 CNN with pre-training and fine-tuning

The second CNN network we tested was almost identical to the one in 2.3 except that it had an additional fully connected layer. This third fully connected layer had two outputs to match our two classes: background and foreground. In this model, the classifier and the features were learned simultaneously to increase efficiency. Using the same pre-trained model as in 2.3, we fine-tuned the three fully connected layers with our labeled training samples.

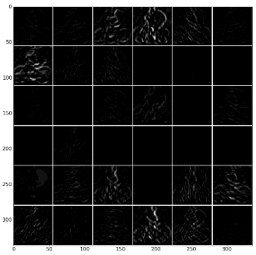
3 Additional Pipeline Components

3.1 Feature Visualization

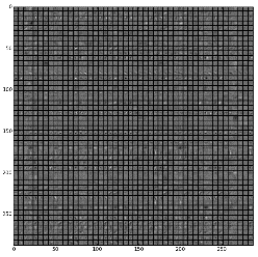
To confirm that the features extracted from test images were meaningful, we visualized the features at different levels in the network. According to work by Erhan[7] done on number recognition, higher-level filters are interpretable and complex. Visualizing these filters allowed us to confirm that our extracted features were meaningful and resembled the test images.



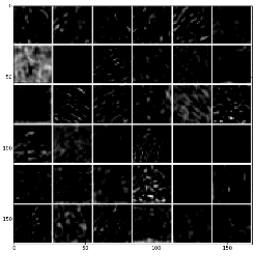
Conv1



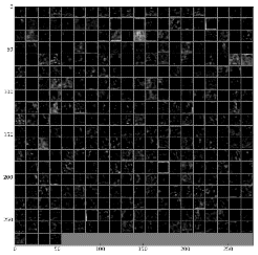
Conv2



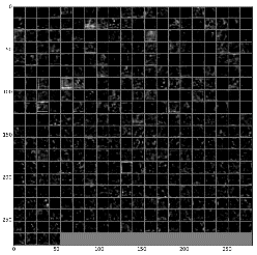
Conv3



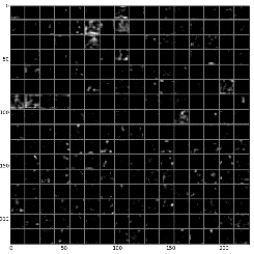
Conv4



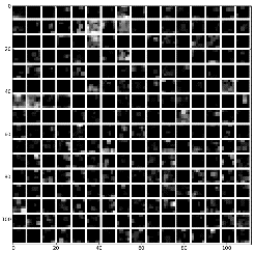
Conv5



FC7



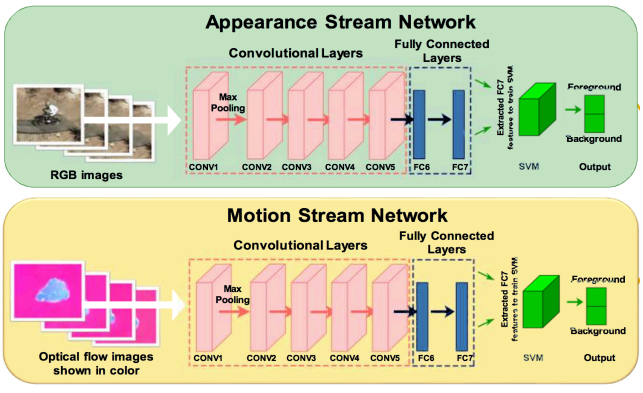
FC8



FC9

3.2 Optical flow

The network was extracting meaningful features, yet sometimes, due to occlusion, deformation or other common tracking problems, the tracker would suddenly jump to another spot much further away in the frame. We used a motion vector in the network in 2.2 in addition to the auto-encoder and SVM but it wasn't very effective. We decided to add a motion stream network to our appearance stream network to capture the temporal data, an approach inspired by Simonyan and Zisserman[8].



Fusion: Occurs after the SVM gives a confidence value and combines the outputs from both networks.

We calculated optical flow between two frames for the sequence and produced RGB optical flow images using Liu's optical flow

code[9]. We passed the RGB optical flow images into an identical network as we had in 2.3 and trained a separate SVM classifier for the motion stream network. We ran both sets of test images through both networks and combined the SVM output scores using late fusion to take both the motion and the appearance of the target in the video into account.

3.3 Updating the model

Because the DT uses a two-stream network, the two streams were not updated simultaneously. The motion stream network was updated more frequently than the appearance stream network because the direction and speed of the target changed more rapidly than its appearance. The motion stream network was updated every four frames whereas the appearance stream network was updated either every 50 frames or if the confidence values of the test images dropped below a defined threshold. This threshold was set to .85 during our experiments and indicated that a sudden occlusion or illumination change had happened.

Since our algorithm to produce training samples uses a small number of frames to collect a large number of training samples, over-fitting could have occurred. However, since our tracker only focuses on a single target, training our networks and classifiers for each video sequence is not an issue. By updating the model, any over-fitting that could interfere with the way the tracker handles target deformation or illumination is handled.

Our method of updating the model allows the DT to handle over-fitting, change in motion, target shape, occlusion, and illumination.

3.4 Handling Scale Change

Handing scale change was an important component of our pipeline because both datasets contained videos where scale change was present. Additionally, if the target increases or decreases in size while the bounding boxes stay the same size, the resulting produced test images contain a large amount of background information or only a portion of the target, leading to incorrectly positive test samples.

From prior experimentation, we found that the features our network was learning were largely scale invariant and that our network was accurately handling the location of the target during the beginning of the scale change. Thus, update the model based on a slight scale change was not necessary, and we could instead pass test samples that contain different scales to address scale change.

Using the original methodology of collected training and testing samples and the two-stream network described in 3.1, the tracker would choose the location of the target. At that chosen location, 20 different scales of that image were passed as test images to the appearance network. The scale with the highest confidence score would be chosen and the size of the bounding box would be updated.

4 Results

4.1 Data Sets

The Deep Tracker was tested on the 315 video sequences from the Amsterdam Library of Ordinary Videos for tracking (ALOV++)[10] and the 29 video sequences from the Visual Tracker Benchmark[11] datasets. These datasets were chosen based on their videos' diversity in circumstance—various combinations of classical computer vision problems such as occlusion, illumination, shape change, and low contrast, were present in these videos.

Both datasets included the videos in single frames, the ground-truth annotations, and the results of state of the art trackers that had been run on the dataset. The ALOV++ dataset had ground-truth annotations for every 5 frames and the Benchmark dataset had ground-truth annotations for every frame. Both ground-truth annotations acknowledged scale change.

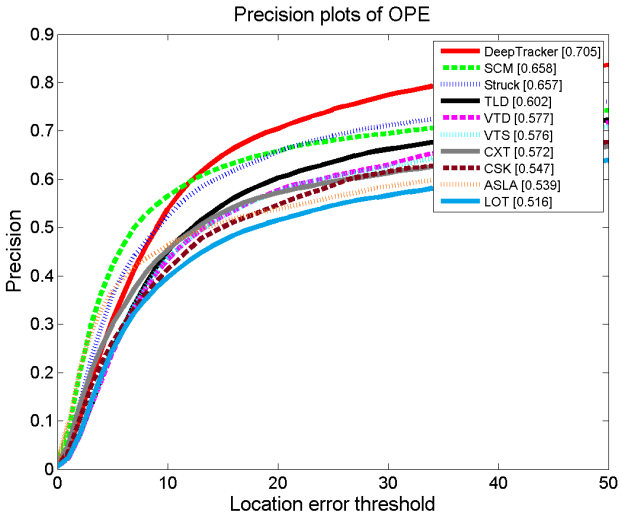
4.2 F-Scores

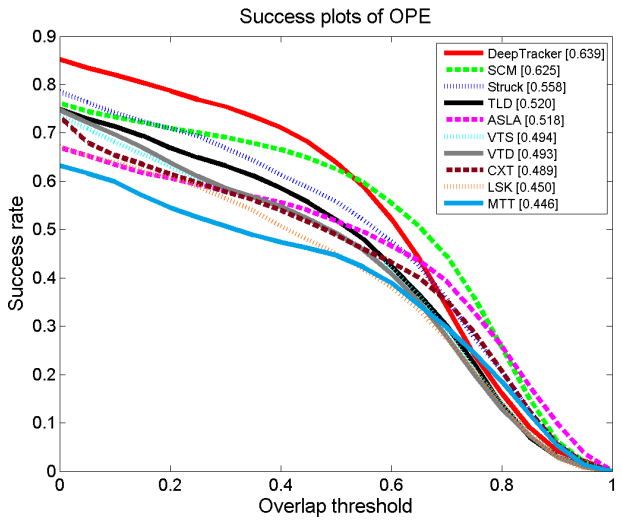
We used two methods of comparison to evaluate our tracking method against two of the current state of the art methods. The first method was by calculating the F-score of the trackers using precision and recall to measure the accuracy in the tracking. The scores are shown in the below tables. In the Visual Tracker Benchmark and the ALOV++ dataset, our DT outperforms the state of the art tracker by at least 4.28% and 2.77% respectively.

F-Score Comparisons			
Visual Tracker Benchmark			
	DT	**Struck**	**SCM**
Average	65.92	55.19	61.64
ALOV++			
	DT	**Struck**	**Difference**
Average	68.96	66.00	2.77

4.3 Success and Precision Plots

Furthermore, we compared the DT to the top ten state of the art methods using success and precision plots. These plots use receiver operating characteristic curves (ROC) to illustrate the tradeoffs between sensitivity and specificity, which have an inverse relationship. These curves were generated by plotting the comparison of true positives out of the number of total true positives and the fraction of false positives out of the total number of false positives. The success plot shows the overlap in the DT's bounding boxes and the ground-truth and the ratio of successful tracking whereas the precision plot shows the error in center location of the bounding box. The overall success and precision plots are shown below.





Success and precision plots were also calculated for specific classical computer vision problems. The DT outperformed state of the art tracking methods in several categories including occlusion, fast motion, background clutter, out of plane rotation, out of view, illumination variation, deformation, motion blur, and low resolution.

Overall, our results show that the DT outperforms current state of the art tracking methods.

References

[1] Bengio, Yoshua, P. Lamblin, D. Popovici, H. Larochelle. “Greedy Layer-Wise Training of Deep Networks.”

[2] P. Lamblin,and Y. Bengio. “Important Gains from Supervised Fine-Tuning of Deep Architectures on Large Labeled Sets.”

[3] Y. Yang, G. Shu, and M. Shah. ‘Semi-supervised Learning of Feature Hierarchies for Object Detection in a Video.”

[4] A. Krizhevsky, I. Sutskever, and G. Hinton. “ImageNet Classification with Deep Convolutional Neural Networks.” *NIPS 2012: Neural Information Processing Systems.*

[5] Ibid.

[6] Y. Jia “Caffe: An Open Source Convolutional Architecture for Fast Feature Embedding.” 2013: <http://caffe.berkeleyvision.org>.

[7] D. Erhan, Y. Bengio, A. Courville, and P. Vincent. “Visualizing Higher-Layer Features of a Deep Network.” 2009.

[8] K. Simonyan and A. Zisserman “Two-Stream Convolutional Networks for Action Recognition in Videos.” 2014.

[9] C. Liu. Beyond Pixels: Exploring New Representations and Applications for Motion Analysis. Doctoral Thesis. Massachusetts Institute of Technology. May 2009.

[10] A. Smeulders, W. M., Dung M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah. “Visual Tracking: an Experimental Survey.” *IEEE Transaction on Pattern Analysis and Machine Intelligence (PAMI)*, 2013.

[11] Y. Wu, J. Lim, and M. Yang. “Online Object Tracking: A Benchmark.” *IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 2013.