A Reimplementation of Online Tracking by Learning Discriminative Saliency Map with Convolutional Neural Network

Aaron Gonzales
gonza647@msu.edu

Steven MW Hoffman
hoffm470@msu.edu

1. INTRODUCTION

We chose to implement our machine learning project with a *deep learning thrust*. We are basing our project on the work of Hong, et al. [1], and our goal is to reproduce their work through the creation of a viable demo. Hong, et al. propose a novel online tracking scheme for use in applications where tracking an object through frames of a video is desired. Although there are a large range of potential applications for object tracking software, it is still a difficult problem due to challenges of occlusion, pose variations, illumination changes, fast motion, and background clutter [1], and any potential tracking solution must have robust methods to overcome these challenges. This paper proposes to solve these problems using the combined utility of both a convolutional neural network (CNN) and support vector machine (SVM), wherein a discriminative saliency map is produced and used to calculate the posterior probability of the location of the target in the image. The object tracking algortihgm by Hong, et al. is described below and is portrayed in Figure 1.

The tracking algorithm proposed by [1] begins by first generating a set of sample images, each of which is drawn from candidate bounding boxes near where the target was located in the previous frame. Each of these sample images is passed through a pre-trained CNN [2]. A CNN is used because CNNs have been shown to be very successful at creating image representations useful for object discrimination. Additionally, CNNs have shown promise in overcoming challenges from many of the current difficulties presented in object tracking, including pose variations, illumination changes, and background clutter. For each image, the output from *the first fully-connected layer* of the network is extracted and is used as the feature vector describing that image sample. The image sample feature vector is then given to an SVM which will classify it as either a positive sample, including the object we are tracking, or a negative sample, which does not include the object we are tracking. In contrast to the CNN, which is learned offline on generic image data not specific to the target, the SVM is learned online using the samples it has seen up to the previous video frame. This allows the SVM to adapt to different types of objects which the user would like to track. For each positive sample, the target-specific features are extracted by using those features which corresponded to positive weights in the SVM, setting all other feature values to zero. The positive weights of the SVM are chosen because they are the weights which correspond to positively identifying a target. These target-specific features are then backpropagated through the CNN, producing an image containing a saliency map. A saliency map is created for every positive sample, and these are combined to make a final target-specific saliency map where larger values in the map indicate a larger likelihood that the target is located at that pixel. Through these means, the target can be segmented out of the image at a near pixel level. A generative model is then computed to refine the likelihood estimate based on what has been seen in previous frames. Finally, the posterior probability of each original sample containing the target is calculated, and the bounding box containing the highest posterior is selected as the target location. With the target successfully found, the algorithm begins anew in the next frame, creating candidate bounding boxes around where it just found the target in the preceding frame.

2. RELATED WORK

The problem of object tracking in video is a large domain, so we will restrict our discussion here to a few works which also attempted to use CNNs to perform tracking, as these are most relevant to the paper we have chosen by Hong, et al. [1]. We also highlight how the approach proposed by Hong, et al. differs from these approaches, making it a novel work.

[3] utilizes a CNN for tracking; however they use an offline trained CNN. They also require a separate class-specific network to track various other objects. Hong, et al [1], in contrast, proposes using a pre-trained CNN used for large scale image classification which is trained on generic image data. An online trained SVM is then used in conjunction with the CNN by Hong, et al. to learn the target specific information.

[4] also uses a pre-trained network where a stacked denoising autoencoder is trained using a large number of images to learn generic image features. However, as this network is trained on small grey images, its representation power is limited.

[5] proposed a target-specific CNN for tracking, where the CNN is trained online. However, this network is shallow in comparison with the deep CNN proposed by [1], and as such does not take advantage of the rich information a deep CNN provides.

In addition to the novelties described above, the tracking method proposed by [1] differs from all three of the above papers in a few important ways. First, it uses an online trained SVM with the offline trained CNN in order to adapt the tracking to whatever type of object the algorithm happens to be presented with. Secondly, it uses saliency maps

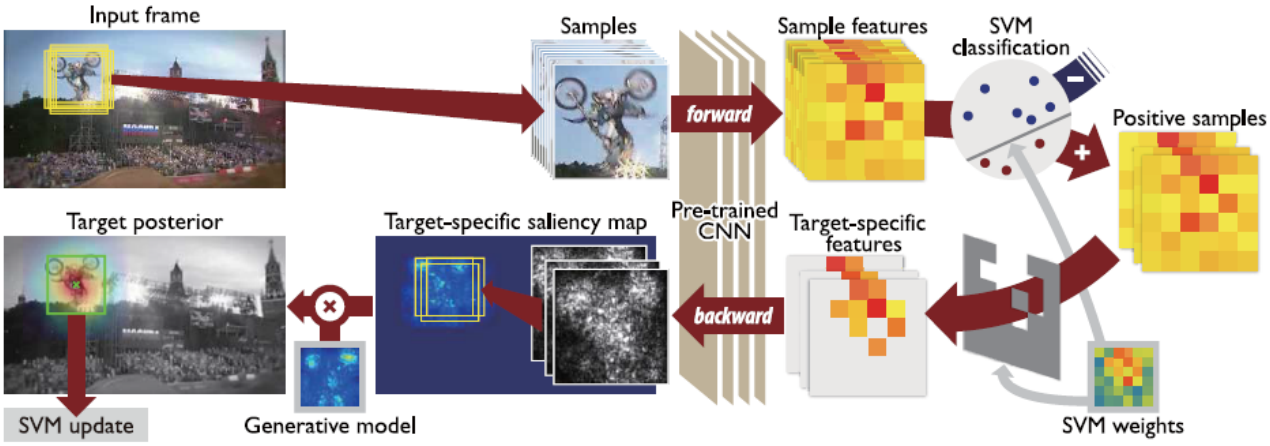


Figure 1: A pictoral description of the algorithm described by Hong, et al. in [1]. Candidate frames are passed through a pre-trained convolutional neural network (CNN) and a feature vector is extracted for each sample from the first fully-connected layer of the CNN. These feature vectors are passed through an SVM, retaining only samples which the SVM believes to contain the target object. The model weights of the SVM are used to determine the target-specific features of each positive sample, which are then back-propagated through the CNN to retrieve a target-specific saliency map. A generative model, produced from previous frames, is convolved with the target-specific saliency map, and the posterior probability that each of the original samples contained the target object is computed.

to find the precise location of the tracked object.

3. DATASET DESCRIPTION

Following the form of Hong, et al. [1], we tested the tracking algorithm against video sequences from the Tracker Benchmark v1.0 dataset [6]. This dataset contains 50 testing sequences, i.e. videos containing a specific object to track, all of which vary in video length. These sequences present a variety of tracking challenges such as illumination variation, deformation, motion blur, background clutter, etc, which can be seen in Table 1. In addition to these visual challenges, objects of interest vary from humans, to vehicles, to animals, and to various inanimate objects. Each of these sequences contain ground truth text files which contain bounding box information for objects of interest at each frame of the video. Some of the videos contain multiple objects of interest and as a result come with multiple ground truth files. We will analyze our tracking results on these videos qualitatively. Video frames from a selection of these testing sequences is shown in Figure 2.

4. TRACKING ALGORITHM

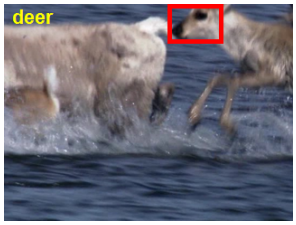
In the introduction, we provided a brief overview of the tracking algorithm proposed by Hong, et al. [1], which we have implemented for our project. In this section, we will go into more detail describing this algorithm and how we have implemented it. All of the code for this project was written in Matlab and was executed on both Matlab 2014b and Matlab 2015b. Note that Figure 1 provides a nice pictorial overview of the algorithm.

4.1 Generating Candidate Sample Patches

In order to search for the target object in the current frame, candidate sample patches need to be generated, i.e. small portions of the image frame cropped to the size of the target object's initial ground-truth bounding box. The

Table 1: Various video attributes contained within the Tracker Benchmark v1.0 dataset which make objects harder to track in the video. Each of the 50 test videos is annotated with the attributes which that video contains [6].

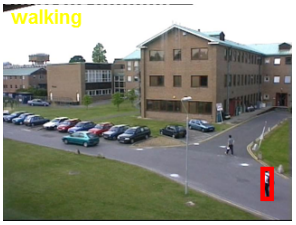
Video Attribute	Description
IV	*Illumination Variation*: the illumination in the target region is significantly changed.
SV	*Scale Variation*: the ratio of the bounding boxes of the first frame and,the current frame is out of the range [1/ts, ts], ts >1 (ts=2).
OCC	*Occlusion*: the target is partially or fully occluded.
DEF	*Deformation*: non-rigid object deformation.
MB	*Motion Blur*: the target region is blurred due to the motion of target or camera.
FM	*Fast Motion*: the motion of the ground truth is larger than tm pixels (tm=20).
IPR	*In-Plane Rotation*: the target rotates in the image plane.
OPR	*Out-of-Plane Rotation*: the target rotates out of the image plane.
OV	*Out-of-View*: some portion of the target leaves the view.
BC	*Background Clutters*: the background near the target has the similar color or texture as the target.
LR	*Low Resolution*: the number of pixels inside the ground-truth bounding box is less than tr (tr =400).



(a) Deer: MB, FM, IPR, BC, LR

(b) MountainBike: IPR, OPR, BC

(c) Tiger1: IV, OCC, DEF, MB, FM, IPR, OPR



(d) Walking: SV, OCC, DEF

Figure 2: Video frames from sample sequences in the Tracker Benchmark v1.0 dataset along with bounding boxes drawn around the target object [6]. The video attributes are also listed for each shown video.

rest of the tracking algorithm will then be focused on choosing the candidate sample patch, which we will also call a sample or a bounding box, that has the maximum posterior probability of containing the target. To generate candidate samples, we first let the bounding box containing the target in the previous frame be a candidate sample for the current frame. Note that our algorithm will always use the ground truth bounding box provided with the dataset for at least the first frame, beginning tracking with the next frame; this ensures there will always be a bounding box from a previous frame for our algorithm to draw on and it allows us to easily specify which object the algorithm should track in a video by placing the initial bounding box over that object. We then generate 120 additional samples by drawing from a normal distribution defined by $x_i \sim \mathcal{N}(x^*_{t-1}, \sqrt{wh}/2)$, where w and h denote the width and height of the target bounding box in the initial frame, x^*_{t-1} is the location of the target in the previous frame, and x_i is the ith sample generated at frame t, the current frame. This is the same procedure as used by [1], and it is based on the assumption that the most likely locations for the object are distributed around the location of the object in the previous frame. An example of some of these samples is shown in Figure 3.

4.2 Computing Sample Features with a CNN

For the next step of the algorithm, a CNN is used to transform each sample patch into a sample feature vector. The CNN used should be pre-trained on a dataset for generic object detection in images; this is because CNNs trained for such a task have been known to create transformed representations of images useful for object discrimination [7]. Each candidate sample is passed through the CNN, and the output of the CNN's first fully-connected layer is used as the feature vector representing that sample.

The original algorithm by Hong, et al. [1] used the Caffe

Figure 3: The location of the target from the previous frame (red box) and the normally sampled candidate patches (yellow boxes) for a biker from the video tracking dataset [6].

Model Zoo implementation of the pre-trained *R-CNN* created by Girshick, et al. [8] to create their sample feature vectors. However, [1] specified in their paper that CNN models other than Girshick's R-CNN may also be used for similar results. Thus, to circumvent the large amount of time needed to install Caffe, we decided to instead use MatConvNet [9], a CNN toolkit for Matlab with a simple installation process. MatConvNet provides several pre-trained CNNs, and we decided to use *VGG-F*, a CNN that has achieved state-of-the-art performance on the ImageNet object recognition database [10].

4.3 Using an SVM to Find Target-Specific Features from Sample Features

Online SVM code has been acquired from Cauwenberghs, et al. [11].

Still need to do this!!!

4.4 Computing Class Saliency Maps

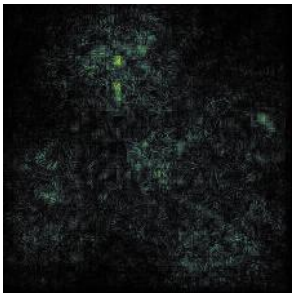
Update this:

Still need to do this!!!As one step of the algorithm, the target-specific features are back-propagated through the CNN in order to generate a target-specific saliency map, as detailed in [12]. This process involves taking the target-specific feature vector, inserting it into the first fully connected layer of the network, back-propagating this data to the input layer, and performing minor post-processing on this data. Due to the process of back-propagation, the saliency maps are meant to highlight those pixels which have the most impact in identifying the object in the image, effectively segmenting the foreground from the background. A Matlab function was written to perform this operation. Because the code to generate the target-specific features has not yet been written, this code currently takes as a parameter the sample features directly, which may include information about the background. An example image with its corresponding saliency map is shown in Figure 4. Note that the saliency map mostly highlights those pixels that correspond to the target foreground.

5. WORK DISTRIBUTION AMONG AUTHORS

This section outlines how the work was distributed between the two authors of this report.

(a) Original Image



(b) Saliency Map

Figure 4: The class saliency map for an image taken from the video tracking dataset used in this paper.

5.1 Aaron Gonzales

Aaron Gonzales was primarily responsible for generating the candidate sample patches, for using an SVM to find the target-specific features from the sample features, and training the online SVM. He also researched the related work and was the primary contributor towards creating the Final Project presentation.

5.2 Steven Hoffman

Steven Hoffman was primarily responsible for writing the skeleton code for the overall tracking algorithm, passing samples through the CNN to get the sample features, back-propagating the target-specific features through the CNN to create a target-specific saliency map, computing the generative model, and finding the target posterior.

5.3 Both Authors

Both authors worked together to write up the various project reports, to give the final presentation, and to run the code on various testing sequences (i.e. videos).

6. REFERENCES

[1] Seunghoon Hong, Tackgeun You, Suha Kwak, and Bohyung Han. Online tracking by learning discriminative saliency map with convolutional neural network. *arXiv preprint arXiv:1502.06796*, 2015.

[2] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia*, pages 675–678. ACM, 2014.

[3] Jialue Fan, Wei Xu, Ying Wu, and Yihong Gong. Human tracking using convolutional neural networks. *Neural Networks, IEEE Transactions on*, 21(10):1610–1623, 2010.

[4] Naiyan Wang and Dit-Yan Yeung. Learning a deep compact image representation for visual tracking. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 809–817. Curran Associates, Inc., 2013.

[5] H. Li, Y. Li, and F. Porikli. Deeptrack: Learning discriminative feature representations online for robust visual tracking. *Image Processing, IEEE Transactions on*, PP(99):1–1, 2015.

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

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[8] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Region-based convolutional networks for accurate object detection and segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 38(1):142–158, 2016.

[9] Andrea Vedaldi and Karel Lenc. Matconvnet: Convolutional neural networks for matlab. In *Proceedings of the 23rd Annual ACM Conference on Multimedia Conference*, pages 689–692. ACM, 2015.

[10] 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.

[11] Gert Cauwenberghs and Tomaso Poggio. Incremental and decremental support vector machine learning. *Advances in Neural Information Processing Systems 13: Proceedings of the 2000 Conference*, 13:409, 2001.

[12] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.