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 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 posterior probability 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. With the target successfully segmented out of the frame, the algorithm begins anew in the next frame, creating candidate bounding boxes around where it found the target in the previous 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 to find the precise location of the tracked object.

3. DATA DESCRIPTION

Following the form of Hong, et al. [1], we expect to test the tracking algorithm against various benchmark sequences.

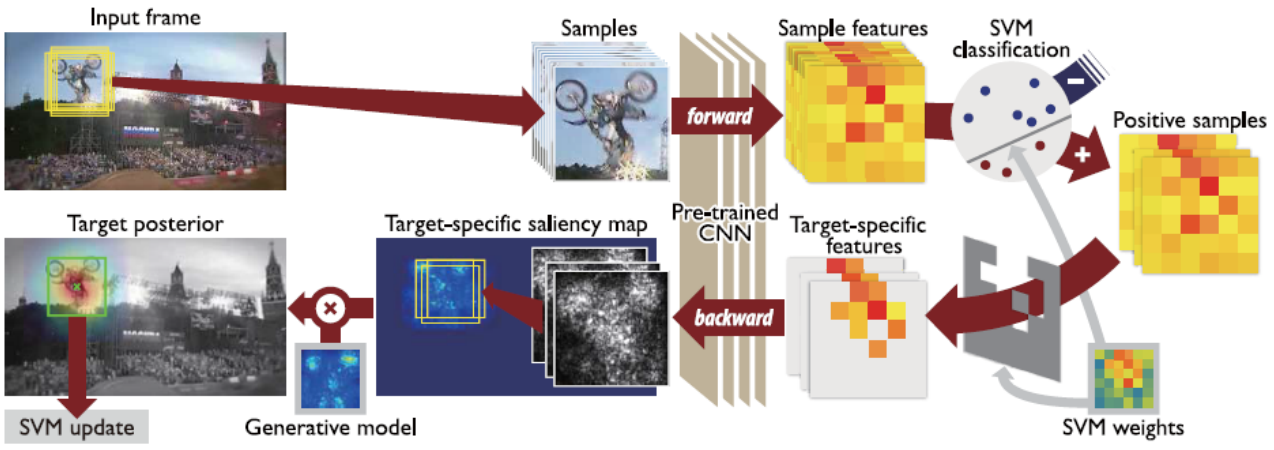

Figure 1: A pictoral description of the algorithm described by Hong, et al. in [1]

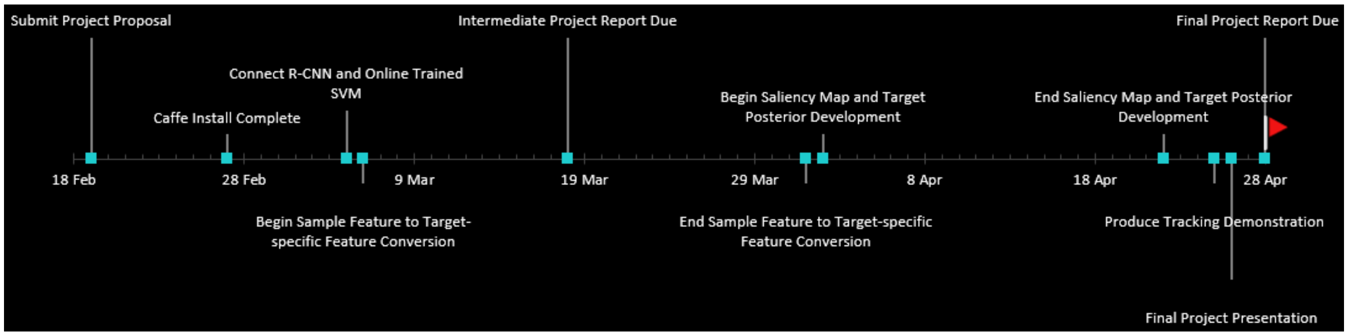

Figure 2: Our project milestones with tentative completion dates. Each tick-mark represents one day on the timeline.

There are a total of 50 tracking sequences which we have downloaded from a tracking benchmark dataset [6]. These sequences present a variety of tracking challenges such as illumination variation, deformation, motion blur, background clutter, etc. In addition to these visual challenges, objects of interest vary from humans, to vehicles, to animals. Each of these sequences contain ground truth text files which contain bounding box information for objects of interest at each frame. Some of the videos contain multiple objects of interest and as a result come with multiple ground truth files. Depending on time availability we may decide to compare our implementation to that of Hong, et al. [1] and other top tracking implementations from the benchmark dataset. Additionally, we may decide to test our implementation and report on our own select video sequences.

4. PROJECT MILESTONES

As a reference the milestones presented in our project proposal are featured in Figure 2.

4.1 Completed Milestones

4.1.1 Generating Candidate Sample Patches

We have developed code that allows us to specify an initial bounding box around the starting object of interest. From each frame we are now able to generate target candidates by drawing 120 samples 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. This is the same procedure as used by [1]. These samples are then resized to 227 x 227 and used as input to our CNN. An example of these samples is shown in Figure 3.

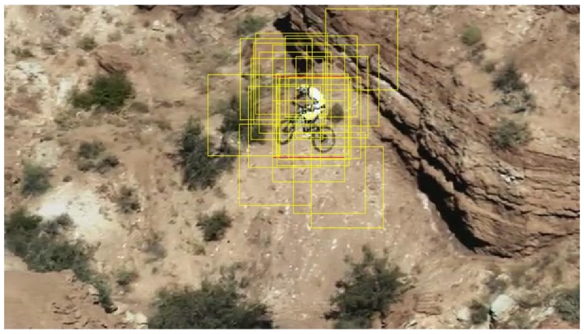

Figure 3: The bounding box (red) and the normally sampled candidate patches (yellow) for a biker from the video tracking dataset.

4.1.2 Computing Sample Features with a CNN

The original algorithm by Hong, et al. [1] used the Caffe Model Zoo implementation of the pretrained R-CNN created by Girshick, et al. [7]. The algorithm takes an image sample as an input and takes the output of the first fully connected layer of the R-CNN as the feature vector of the image sample. 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 [8], a CNN toolkit for Matlab with a simple installation process. MatConvNet provides several pretrained CNNs, and we decided to use VGG-F, a CNN that has achieved state-of-the-art performance on the ImageNet object recognition database [9]. Upon acquiring this net, we wrote code that takes the network and the sample image as an input and returns the sample features of that image (i.e. the output of the first fully connected layer).

4.1.3 Implementing an Online SVM

Online SVM code has been acquired from Cauwenberghs, et al. [10]. We may use different online SVM code in the future to match exactly what Hong, et al. [1] use in their approach, but for the moment, this code is sufficient. This code allows us to tweak various parameters to suite our needs, for example, whether or not to train the SVM incrementally and which kernel functions to use when training the SVM. While we have yet to connect the CNN and SVM, we have run tests on the SVM to help us better understand the manner in which to integrate our SVM, as well as to understand the results after training online and testing.

4.1.4 Computing Class Saliency Maps

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 [11]. 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.

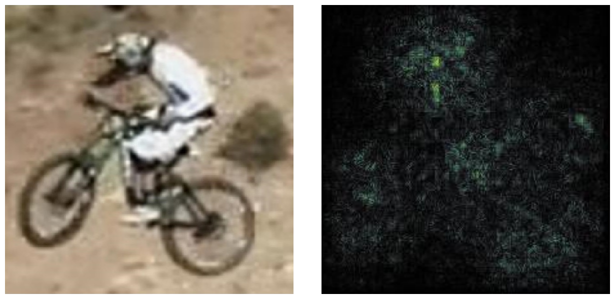
(a) Original Image (b) Saliency Map

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

4.2 Remaining Milestones

The connection of our CNN and SVM are not complete just yet. However, as we are now able to feed our CNN the appropriate input it is simply a matter of using the output of the first fully connected layer as input to our SVM. This process should be fairly straight forward, and thus we expect little issue connecting these two large components of our implementation. After connection we need to compute the SVM scores of candidate samples represented by the neural network features and classify them into target or background. Only samples with a positive SVM score will be used for developing the target-specific saliency map. We also need to implement the online training of the SVM after each frame, which may take a bit longer.

Although the class saliency maps have already been generated, a target-specific saliency map still needs to be generated by aggregating all the class saliency maps from a specific video frame. We do not project that this will take very long to accomplish. Then, in order to perform foreground-background segmentation, a generative model must be created and updated at each frame and convoluted with the target-specific saliency map. We are projecting that this task may take a substantial amount of our remaining time, as generating the model seems to be an intensive task.

After completing all the individual components of the algorithm, we will need to put them all together into one component, which we can iterate through frame by frame. As this involves all the components previously worked on, it will likely take a few weeks to complete and debug. Once having completed this, the remaining milestones will be to create a demo, record some results, create the final report, and create the final project presentation. To create quality documents, this will also take several weeks to complete.

5. 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] 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.

[8] 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.

[9] 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.

[10] 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.

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