Adaptive GOTURN

Ibrahim Akbar, Yen Chang, Eli Uc, Chao Long
ECE271B Statistical Learning II
University of California, San Diego

Abstract

Deep Learning implementations in generic object tracking have provided effective results in modern state of the art solutions. Generic object trackers take an arbitrary object at a given time and attempt to track it thereafter. Held's GOTURN architecture [10] trained a deep learning implementation completely offline to obtain effective tracking at fast processing times and therefore, high video frame rates. However, offline trackers solely rely on the vast amount of prior knowledge to perform this task, unable to use live feature information in its decision making. We will show that hybrid trackers implementing both offline training and online fine-tuning out perform simple offline trackers. We modify Held's completely offline training by taking the CNN architecture and making its last fully connected layer modifiable to learn weights according to the current environment. Transfer Learning is utilized to share most of GOTURN's same pre-trained weights up to the last fully connected layer to control the computational complexity of the tracker and not severely disrupting the fast processing that GOTURN featured. Our results are comparable with what Held presented however with a significant reduction in speed.

1. Introduction

Generic object trackers capture a desired arbitrary object within a rectangular bounding box in a video frame and attempts to follow this object with a more appropriate adaptively shaped bounding boxes in subsequent frames. Unlike image classifiers, modern state of the art trackers are far behind human level tracking and are easily misdirected under a variety of environments. These challenges seen in the literature today makes generic object tracking currently an open problem.

Vast amounts of solutions in the literature make use of solely online information to drives an algorithm. These tracking solutions take the form of a feature extractor taken from a current frame, and running them through more classical machine learning algorithm such as Support Vector Machine (SVM) or Boosting to properly discriminate. Because these algorithms take a certain processing time to extract the desired feature and calculate through a certain number of iterations, performance was severely capped and frame rates were consequently affected. Though these techniques were capable of producing accurate results (if extracting the proper feature), its live computations made them difficult to use in practical applications.

The strong wave of research done in neural networks has made it clear that we can obtain classification precision that surpasses a human's ability to classify if given a deep enough network trained on a large enough dataset. This is especially true for image classification using convolutional neural networks (CNN). Convolutional neural networks take an input image and convolves it with a series of kernels to obtain more abstract information about the image. This step is performed successively along with downsampling and non-linear activation until it reaches a final linear layer. This final layer takes all the weights connected to each neuron and connects it to a desired number of outputs. During training, these outputs are compared to the ground truth to obtain a loss and ultimately a cost, which utilizes gradient descent to backpropagate to each weight in the network and updates them to reduce the overall cost.

The boom of neural networks in the machine learning realm begs the exploitation of offline weight training using a priori information for later test implementation. This concept was applied by Held's GOTURN algorithm [10] and Bertinetto et al. [3]. GOTURN is a completely offline trained CNN regression tracker and also a state-of-the-art generic object tracker that produced competitive results in precision and robustness in the VOT2014 challenge. This completely offline approach allowed for minimal live computation, as the only processing needed was inputting two images through a network with already pre-trained weights, and no backpropagation. The algorithm is robust to viewpoint changes, lighting variations and deformations, while poorly handling occlusion and intersections with external rigid bodies. This approach exploits the superior performance of neural networks (in contrast with classical machine learning approaches) and manifests the vast amount of a priori videos into a set of pertained weights to reduce computational complexity and track in 100 FPS video.

While having a vast and robust training set is imperative in developing weights that can account for all sorts of object movements and deformations, it is impossible to obtain enough examples to account for all possible object movement, shifts, occlusions, and deformations that a rigid body can undergo. This will always leave completely offline training susceptible to error. Therefore, there is advantage to providing an online tuning mechanism to this algorithm to provide valuable information during movements not sufficiently accounted for during training. From another perspective, an offline tracker can be seen as an open looped system in control theory, and hence independent of environment. This suggests that the performance would increase if adapted to a close looped system given that the environment is stochastic. Thus we propose a hybrid implementation of GOTURN with online training.

The hybrid implementation that we designed needed to 1) maintain the use of the pre-tained weights to take advantage of the large number of movements that it manifests and 2) be able to take in live information to produce new weights to specifically describe the object being tracked. This was accomplished by applying transfer learning to the CNN architecture designed by Held. The last two fully connected layers were sliced from the CaffeNet architecture [11] in GOTURN and were to have its weights updated by the input image. The rest of the weights remained the same. Intersection over Union (IoU) and robustness were the key metrics used to compare implementations.

2. Related Works

2.1. Online Training Tracker

Generic object trackers are typically trained online, and a foreground-background classifier is created to estimate the new location of the target objects in next frame during the training process [10]. They can produce state-of-art results as well as adapt to the changes of motion for specific objects at test time. However, since the training process is trained completely online, it does not take advantages of the large numbers of videos easily available offline that can improve the performance [10]. The highly effective neural networks can be very time-consuming to train online, and the resulting tracker will therefore perform slowly during test time. Most online trackers range from 0.8 fps to 15 fps with optimal trackers running at 1 fps on a GPU [10, 14].

2.2. Offline Training Tracker

In contrast to generic object tracker traditionally trained entirely from scratch online, Offline training trackers take the advantages of the large amounts of data and can teach the tracker to handle rotations, changes in view points, light changes, and other complex modifications [10]. These trackers often try to learn a similarity function between frames as Held et al. uses the previous frame [10] and Bertinetto attempts to track from the original input bounding box [3].

2.3. Model Based Tracker

In contrast to generic object based trackers, model based trackers are designed to track a specific kind of object that is being catered by a particular type of model. [7]. During the training process, they learn classifiers for various kinds of objects and rely on them to find the specific object of interest. Although these trackers are trained offline, they cannot be generalized due to practical constraints of each learner and systems.

2.4. Hybrid Tracker

Hybrid trackers use a combination of offline and online training to track objects [6, 16]. They attempt to maximize the benefits of online and offline training. There is a trade off between performance and speed. In our experiments, we propose a hybrid tracker by implementing GOTURN combined with online training and transfer learning.

3. Technical Approach

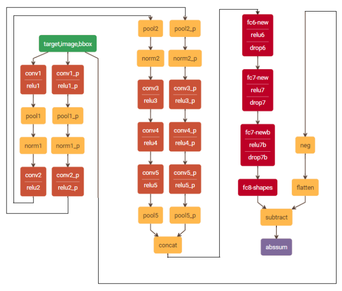


Figure 1: GOTURN Neural Net Architecture

3.1. GOTURN

The architecture of GOTURN is a convolutional neuron network based of first five convolutional layers of CaffeNet and the output of the neuron network contains 4 nodes representing the output bounding box. In the testing process, GOTURN uses a one-pass feed forward network to get the consecutive frame with target bounding box. During training process, GOTURN utilizes the $t - 1$ frame knowledge of target and regresses to the believed coordinates of the target region. In addition, a Laplace distribution based motion model was applied for example generation to get more generic result to address overfitting problems. For the training process both for videos and images, L1 loss and Stochastic Gradient Descent have been applied to train the neuron network. And the datasets GOTURN uses are ImageNet and ALOV300++.

3.2. Transfer Learning

It is common practice to avoid "reinventing the wheel" when using deep CNNs. The most successful architectures are several layers deep and have an intricate set of hyper-parameters that a variety of research groups have tuned. Common architectures such as VGG, AlexNet, CaffeNet, etc. are readily available to use. Transfer Learning is the practice of using these architectures and freezing the kernels up to a particular layer. The succeeding layers are then trained with backpropagation. We propose using Transfer Learning for online training. By using a section of a pre-trained networks, only a smaller set of fully connected layers need to be trained and the amount of training time can be reduced significantly by transfer learning. Also, since we only tune the last fully connected layer in our training, it will prevent the overfitting problems by fine-tuning the last few fully connected layers.

3.3. Online Training

In the online training process, we use Stochastic Gradient Descent instead of Adam or other optimizers because we do not want loss function converging too fast and leads to an overfitting to the previous frame.

$$w_i = w_i - \frac{\partial y_i}{\partial w_i} + \lambda w_i \tag{1}$$

Since this is a multiple task outputs for the location of bounding box, we use Cross-Entropy loss for our empirical risk loss function.

$$R_{emp} = \frac{1}{N}\sum_{i=1}^{N} L[y_i, g(x_i)] \tag{2}$$

$$L[y_i, g(x_i)] = (1 - y_i)(1 - \mathbb{P}\{g(x_i)\}) + (y_i)\mathbb{P}\{g(x_i)\}$$

3.3.1 Example Generation

In order to further prefer smaller motions over larger motions and increases the training amount training samples are generated with shifted and scaled transformations. These transformation have a Laplacian Distribution placed over the GOTURN as Held et al. [10] observed this distribution for motion in the ImageNet and ALOV training set.

$$L(\mu, z) = \frac{1}{2z} e^{-\frac{(x-\mu)}{z}} \tag{3}$$

This means after an output bound box coordinates (c_x, c_y, w, h) are determine it is used as a ground truth and it is possible to generate new coordinates as,

$$c_x^{'} = c_x + w\Delta x$$

$$c_y^{'} = c_y + h\Delta y$$

$$w^{'} = \gamma_w w \qquad h^{'} = \gamma_h h$$

where $(\Delta x, \Delta y, \gamma_w, \gamma_h)$ are random variables. This is how Held et al. introduce motion smoothness into their network and we implement this method during online training as well to re-enforce this concept and provide extra samples.

4. Experiments

This network was tested on the Video Object Tracking (VOT) Challenge Set 2014 which is a popular set due to a standardized method for assessment emerging in conjunction with the set. This also allowed for direct comparison against the baseline network. Metrics of interests include Intersection of Union, Tracking length, and Robustness; whereas the parameters that are of interest are the learning rate, method of annealing, and amount of layers being trained.

4.1. Evaluation Metrics

4.1.1 Intersection of Union

Intersection of union is one of the most important metrics for object detection and object tracking. It is an indicator of how accurate is the predicted bounding box. IoU can be calculated by intersection of ground truth and predicted bounding box over union of them.

$$IoU = \frac{A_{GT} \cap A_{PD}}{A_{GT} \cup A_{PD}} \tag{4}$$

A_{GT} stands for area of ground truth bounding boxes and A_{PD} stands for predicted bounding boxes.

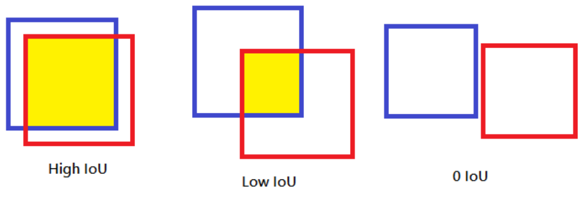


Figure 2: Illustration of IoU

4.1.2 Tracking Length

The tracking length represents the number of successfully tracked frames, which is from it is initialized to its first failure. The failure can be defined as the error is higher than a threshold τ or, namely, the IoU is lower than threshold τ. In order to get the tracking length, we will reinitialize the tracker once it is failed. Reinitialization is simply input the ground truth bounding box instead of previous predicted bounding box.

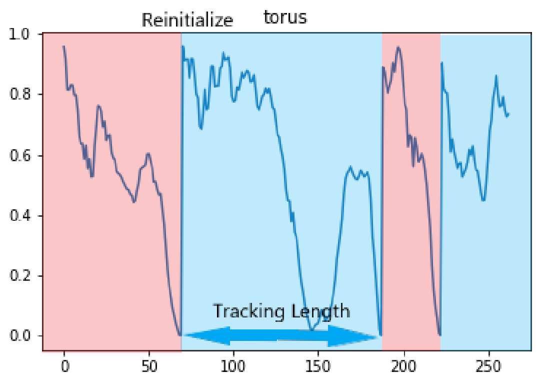


Figure 3: Illustration of Tracking Length

4.1.3 Robustness

In general, the robustness is calculated as the total number of times the tracker has to be re-initialized. Re-initialization occurs when the tracker loses its target and hence has an accuracy of zero. Therefore, the higher the robustness value is, the more unsteady the tracker is.

$$R = \frac{1}{N}\sum_{t=1}^{N} \mathbb{1}(IoU_t) \tag{5}$$

$$\mathbb{1}(IoU) = \begin{cases} 1 & IoU = 0 \\ 0 & IoU > 0 \end{cases}$$

4.2. Experiments Setup

In the experiments, we want to find out the best strategies and parameters for online training. We anneal the learning rate rather than having it fixed under the intuition of decreasing confidence of predicted results. This comes from the fact that we do not have an observation means that provides us information at each current frame. There are different strategies for learning rate decaying, such as step decaying, exponential decaying and sigmoid decaying. We tried the following strategies and believe these are most reasonable as they are continuous and we assume that the uncertainty has a continuous probability distribution given that the environment is a real space. Let lr, γ, t, st denote the learning rate, decay rate, current time frame, and step size.

Exponential Decay

$$lr_{t+1} = lr_0 \cdot \gamma^t \tag{6}$$

Inverse Decay

$$lr_{t+1} = \frac{lr_0}{(1+\gamma \cdot t)^p} \tag{7}$$

Sigmoid Decay

$$lr_{t+1} = \frac{lr_0}{1+\exp\left(\gamma \cdot (t - st)\right)} \tag{8}$$

Using these various rates we assessed the tracker's performance, based on the number of layers, trained, the number of examples per iteration, and the parameters necessary for the simulated annealing.

5. Results

From all the graphs we can see that the performance metrics increase marginally while have the capability of decreasing significantly if the parameters are tuned incorrectly. This goes to show the large impact online training can have on a network. There is a correlation between robustness and accuracy which is to be expected; however, because robustness requires re-initialization it also introduces a bias into the accuracy presented. Kristian et al. [12] show that this bias introduced does not cause the accuracy to deviate far from the actual accuracy of the tracker and consider the various accuracies when assessing VOT submitted trackers. Thus we concluded that it would be a valid metric for assessing our performance. It is further noticeable that the parameters for the annealing play an important role in determining how well the network will adjust itself.

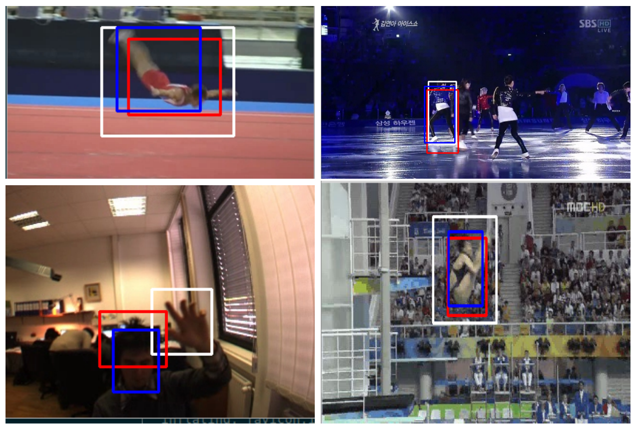
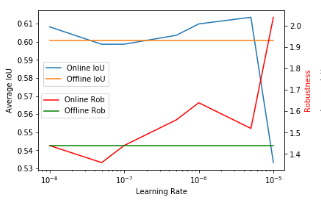
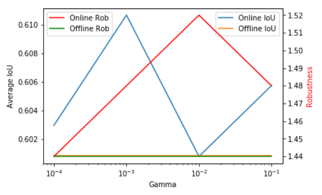


Figure 4: Result Illustration, white one is ground truth, blue one is adaptive goturn and red one is original goturn,

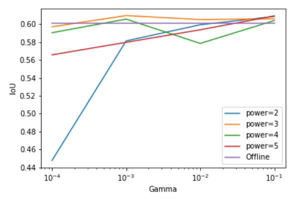


(a) Learning Rate Variation

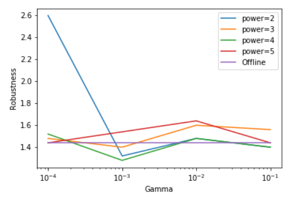


(b) Gamma Variation

Figure 5: Exponential Decaying

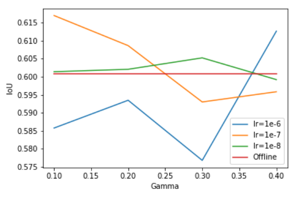


(a) IoU

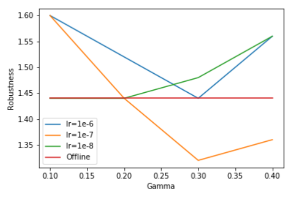


(b) Robustness

Figure 6: Inverse Decaying

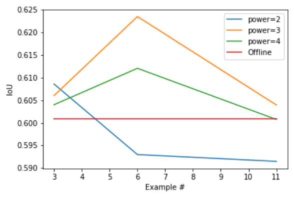


(a) IoU

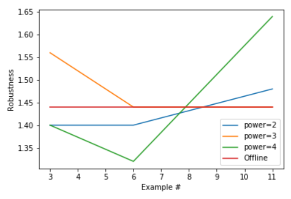


(b) Robustness

Figure 7: Sigmoid Decaying

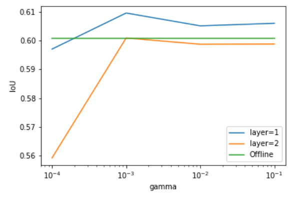


(a) IoU

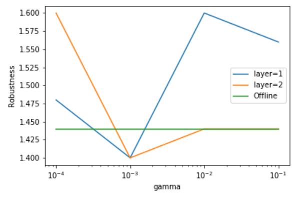


(b) Robustness

Figure 8: Example Number Variation



(a) IoU



(b) Robustness

Figure 9: Online Training with Different Number of Layers

	$\gamma = 10^{-1}$	$\gamma = 10^{-2}$	$\gamma = 10^{-3}$	$\gamma = 10^{-4}$
Exp.	0.602973	0.610683	0.600804	0.605757
Inv.	0.606028	0.605142	0.609605	0.597058
Sig.	0.585762	0.593478	0.576786	0.612657

Table 1: IoU vs. γ

	$\gamma = 10^{-1}$	$\gamma = 10^{-2}$	$\gamma = 10^{-3}$	$\gamma = 10^{-4}$
Exp.	1.44	1.48	1.52	1.48
Inv.	1.56	1.6	1.4	1.48
Sig.	1.6	1.52	1.44	1.56

Table 2: Robustness vs. γ

	$lr = 10^{-6}$	$lr = 10^{-7}$	$lr = 10^{-8}$
Exp.	0.609928	0.598713	0.608319
Inv.	0.537948	0.606028	0.603091
Sig.	0.585762	0.617008	0.601406

Table 3: IoU vs. Learning Rate

	$lr = 10^{-6}$	$lr = 10^{-7}$	$lr = 10^{-8}$
Exp.	1.44	1.44	1.44
Inv.	2.32	1.56	1.48
Sig.	1.6	1.6	1.44

Table 4: Robustness vs.Learning Rate

Furthermore as the layers that were being trained increased past the last fully connected layer performance dropped continually dropped as the network is overfitting to the previous frame proportional to the number of weights being trained. This is a good indication that there needs to be some means of discriminating outputs bounding box regions with a high correlation to the targets features versus regions with a low correlation. Thus allowing for filtering of training samples online. We can see that the usage of the sample generation with noisy transformation actually provides an increase in the accuracy. This is because the tracker is immediately being shown different perspectives of the target and hence it reduces the overfitting that can occur from training only on the previous frame perspect.

6. Conclusion

We have shown that a hybrid version of an offline trained tracker can outperform the original version we achieved an overall increase in accuracy by **2.4%** without any loss in robustness. This improvement is highly correlated to the fine-tuning of all the parameters for the online training. However, if compared to the loss in speed of **50%** from 75 fps to 35 fps; it is marginal. In order to solve the loss of speed, we are prepared to use LSTM layers followed by the convolutional layers for only feedforward testing while retaining temporal information. In addition, our model is based on GOTURN using the prior knowledge for online training and which has a higher likelihood of leading to overfitting of the previous frame. In order to solve this, we are also interested in exploring a local generic object detect relying only on the current frame by applying K Nearest Neighbor algorithm with optimization. Furthermore as Bertinetto et al. [3] only use the original input they have a feature set that accurately represents the target but lack sequential frame information that could strength the tracker's awareness. Thus by further modifying GOTURN's architecture to incorporate a similarity function between the original target features and the current frame it will impose a heuristic on the pixels related to the features of the target. These are all interesting avenues of future work that will be explored.

7. Team Contributions

- Ibrahim Akbar - 1/4
- Yen Chang - 1/4
- Chao Long - 1/4
- Eli Uc - 1/4

References

[1] Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. Multiple object recognition with visual attention. *arXiv preprint arXiv:1412.7755*, 2014.

[2] Loris Bazzani, Hugo Larochelle, Vittorio Murino, Joanne Ting, and Nando D Freitas. Learning attentional policies for tracking and recognition in video with deep networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 937–944, 2011.

[3] Luca Bertinetto, Jack Valmadre, João F. Henriques, Andrea Vedaldi, and Philip H. S. Torr. Fully-convolutional siamese networks for object tracking. *CoRR*, abs/1606.09549, 2016.

[4] Luka Čehovin, Aleš Leonardis, and Matej Kristan. Visual object tracking performance measures revisited. *IEEE Transactions on Image Processing*, 25(3):1261–1274, 2016.

[5] Kai Chen and Wenbing Tao. Once for all: a two-flow convolutional neural network for visual tracking. *CoRR*, abs/1604.07507, 2016.

[6] Martin Danelljan, Gustav Hager, Fahad Shahbaz Khan, and Michael Felsberg. Learning spatially regularized correlation filters for visual tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4310–4318, 2015.

[7] Jialue Fan, Wei Xu, Ying Wu, and Yihong Gong. Human tracking using convolutional neural networks. *Trans. Neur. Netw.*, 21(10):1610–1623, October 2010.

[8] Yarin Gal. Uncertainty in deep learning. *University of Cambridge*, 2016.

[9] Daniel Gordon, Ali Farhadi, and Dieter Fox. Re3: Real-time recurrent regression networks for object tracking. *arXiv preprint arXiv:1705.06368*, 2017.

[10] David Held, Sebastian Thrun, and Silvio Savarese. Learning to track at 100 fps with deep regression networks. In *European Conference on Computer Vision*, pages 749–765. Springer, 2016.

[11] 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 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014.

[12] Matej Kristan, Jiri Matas, Ales Leonardis, Tomas Vojir, Roman P. Pflugfelder, Gustavo Fernández, Georg Nebehay, Fatih Porikli, and Luka Cehovin. A novel performance evaluation methodology for single-target trackers. *CoRR*, abs/1503.01313, 2015.

[13] 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.

[14] Hanxi Li, Yi Li, and Fatih Porikli. Deeptrack: Learning discriminative feature representations online for robust visual tracking. *CoRR*, abs/1503.00072, 2015.

[15] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. In *Advances in neural information processing systems*, pages 2204–2212, 2014.

[16] Hyeonseob Nam and Bohyung Han. Learning multi-domain convolutional neural networks for visual tracking. *CoRR*, abs/1510.07945, 2015.

[17] Hyeonseob Nam and Bohyung Han. Learning multi-domain convolutional neural networks for visual tracking. In *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*, pages 4293–4302. IEEE, 2016.

[18] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. *arXiv preprint*, 1612, 2016.