Report: Implementation of GOTURN

Michele Sezgin

Department of Computer Science, Smith College

CSC 370: Seminar: Computer Vision and Image Processing

Professor Nick Howe

December 18th, 2022

Abstract

Object tracking is an important and relevant computer vision task that has a variety of applications in sports, security, autonomous driving, etc. GOTURN is a fast single object tracking algorithm developed in 2018 that is capable of tracking at up to 100 frames per second (FPS) with certain computer architectures. This paper focuses on training an instance of Nrupatunga's 2020 GOTURN implementation with limited training data in Google Colaboratory. Colab's runtime and GPU constraints did not allow for significant results to be produced within the timeframe of the project. Future work will focus on further analysis of this implementation, and analysis of GOTURN variants with different convolutional layer structures and training data.

Introduction

Object tracking is an important computer vision task that has seen great strides in recent years due to increasingly powerful computer architectures and the increasing popularity of deep convolutional neural networks (CNNs). Uses of object tracking include security systems, sports analytics, and autonomous vehicles, among many other important applications. The ideal object tracker is fast (with the ability to track objects in real time), accurate, general, and robust. This can be challenging for many reasons. Accurate trackers are typically computationally expensive, making them slow. Additionally, training an algorithm on enough data to make it general is again computationally expensive. Lastly, it is difficult to create a robust tracker because of the many challenging aspects of real-world tracking scenarios. Objects may become fully or partially occluded for portions of the video, changes in lighting may make it difficult to locate an object, and multiple objects may be moving and interacting with each other in the frame at once.

Previous work

Generic Object Tracking Using Regression Networks (GOTURN) is a fast, general, and accurate single object tracking algorithm developed in 2018. The algorithm was developed by David Held, Sebastian Thrun, and Silvio Savarese at the Stanford University Department of Computer Science. GOTURN utilizes a variant of the CaffeNet architecture, which is a variant of Alexnet. The algorithm is trained on thousands of videos and images of many different classes, using an artificial motion algorithm to simulate object motion, diversifying the training set and making the algorithm more robust. The training datasets include the ALOV300++ dataset and the images labeled with bounding boxes in the ImageNet dataset. Limitations include the inability to track objects that are moving too quickly, the inability to track objects that are occluded for periods of time, and the inability to reliably track the same object when another moving object occludes the object of interest. Despite these limitations, the algorithm still achieves superior performance in many tracking applications.

Methods

Due to computational and time limits, I attempted to run Nrupatunga's pytorch implementation of GOTURN in Google colab. The goal of this project was to determine the accuracy of the GOTURN algorithm when trained solely on the ALOV300++ dataset (as opposed to both the ALOV300++ and ImageNet datasets), to see if I could replicate the video training-only loss calculated in the original GOTURN paper. An additional goal was to make experimentation with the GOTURN algorithm more accessible by outlining a methodology to run a version of GOTURN in colab. I uploaded the files from Nrupatunga's repository into Google Drive, and then mounted them in Colab. The repository was last updated in 2020, so

there were a few outdated dependencies and pieces of code to debug. Additionally, the file structure has been customized to fit my needs for this project. Google colab limits on GPU access and computation time necessitated shrinking the dataset size and number of training epochs in order to produce results within the time period for this project. Because of this, only the first two videos in each ALOV300++ category were selected for training the model.

Results

Within the constraints of this project, training data shows the algorithm loss decayed as follows:

Epoch	Percentage completed	Loss (L2, in pixels)
0	100	92.2
1	100	69
2	32	66.9

In the first epoch, loss (calculated using euclidean distance) stabilized to an average of 92.2 pixels, meaning the bounding box prediction was off by 92.2 pixels by the last batch in the epoch. In the second epoch, loss dropped substantially to 69 pixels on average. The dataset images varied in size, with some images being relatively large (ex: 1280x720) and others relatively small (ex: 480 x 360). Regardless, a bounding box error of 69 pixels is likely quite noticeable. The third epoch loss was at 66.9 pixels 32% of the way through the data before the runtime timed out.

Discussion

This implementation of GOTURN was able to be trained successfully, though slowly, in Google Colab for at least a few epochs, achieving an average error in bounding box prediction of 66.9 pixels partially through the third epoch.

Implementing algorithms with large CNN architectures can be difficult with limited computational resources. Future work could investigate ways to experiment with training the GOTURN algorithm in a computationally efficient and economical way. This could include shrinking the dataset further, reducing image resolutions, or finding a different smaller dataset to train models with. Future work could also include implementing versions of GOTURN with convolutional layers taken from the original AlexNet and Visual Geometry Group (VGG) net iterations, to see if these achieve similar or even superior performance to the CaffeNet version. With the appropriate computing resources, ideally other future models could be trained on other datasets (including the original ImageNet dataset) to push the limits of the generality of the object tracker. Further analysis to be performed includes methods specified in the original GOTURN paper, such as comparing tracker performance to other trackers in the VOT 2014 tracking challenge, and comparing runtime performance on test videos to other trackers. The generality of the original GOTURN implementation was also assessed by comparing loss for common and uncommon classes in the training data, which I'd be curious to assess with my implementation. My implementation was not tested for accuracy on any test videos or image sequences, but ideally, both videos that have been used for testing the original GOTURN algorithm and those that have not would be used for testing. I would be particularly interested to test the performance of the tracker in sports contexts to combine my interests in sports analytics and computer vision. However, given that the tracker does not do well with occlusion and fast

movement, both of which are highly prevalent in sports, I hypothesize that the tracker would not do well in these contexts. One way to potentially combat the tracker's inability to track fast-moving objects would be to expand the search region, at the cost of sacrificing computational efficiency. Other parameter and algorithm specifications to experiment with include loss function, training epochs, batch size, dropout, and artificial motion creating techniques.

References

Held, D., Thrun, S., & Savarese, S. (2016, August 16). *Learning to track at 100 fps with deep regression networks*. arXiv.org. Retrieved December 18, 2022, from https://arxiv.org/abs/1604.01802

Nrupatunga. (2020). *GOTURN tracking implemented in PyTorch.* Retrieved December 18, 2022, from https://github.com/nrupatunga/goturn-pytorch#about-the-project

Supplementary Materials

Sezgin, M. (2022). *GOTURN-Implementation*. Retrieved December 18, 2022, from https://colab.research.google.com/drive/15_krNkxAJaMMymdR6JNtI9GugR6Z8FjN?authuser=1#scrollTo=XWUcnTYcFL3u

Sezgin, M. (2022). *my-goturn*. Retrieved December 18, 2022, from https://drive.google.com/drive/u/1/folders/1Cdq96PZosWavCndi3TC3CJoFAZn7BPML