BBM 418 - Computer Vision Laboratory

HACETTEPE UNIVERSITY
Department of Computer Engineering

Programming Assignment-3
Spring 2018-2019
Dr. Nazli Ikizler Cinbis
TA. Ozge Yalcinkaya

Due Date: 23:59pm on Monday, May 27th, 2019

Single Object Tracking with Regression Networks

In this assignment, you will implement a basic single object tracker by training the network on the given videos. You will use a two-frame architecture so the network will use two features of frames at the same time. You will utilize the given video dataset for training and testing the tracker by considering the ground truth bounding box annotations.

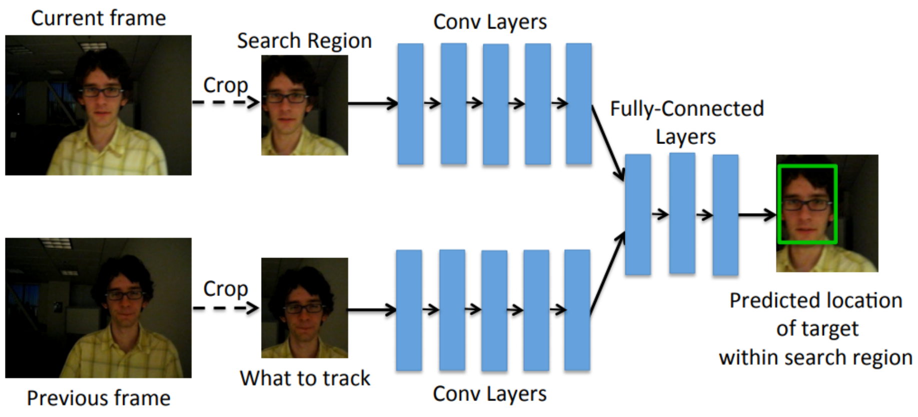

Figure 1: Overview of the tracker.

Background

What you are going to do is, trying to predict the position of the object at the current frame by leveraging the position of it at the previous frame. For this purpose, your network will learn to predict a bounding box from the given combination of features. This combination includes the patch of the object in the previous frame, and the possible position of the object in the current frame. The possible regions are within a search region such as the enlarged version of the previous bounding box. Here, the assumption is: object could not move so fast from frame to frame, so it should be close to its previous position.

The network basically learns the following: given the object (with a context, since we will use the enlarged version of the bounding box) and a search region, the object should move to specified location (the ground-truth bounding box relative to the search region).

What to track? The crop of the object in the previous frame with a context (2 times enlarged bounding box from the center). This crop tells the network which object is being tracked.

Where to look? To find the target object in the current frame, the tracker should know where the object was previously located. Since objects tend to move smoothly through space, the previous location of the object will provide a good guess of where the network should expect to currently find the object. Therefore, you will crop the current frame using the search region, which is 2 times enlarged bounding box of the previous frame from the center.

Network output. The goal of the network is to regress to the location of the target object within the search region. However, the annotations are given over the entire image. Your network will output the coordinates of the objects in the current frame, relative to the search region. Therefore, you should arrange the bounding box position according to cropped search region. (see Figure 2). The network's output consists of the coordinates of the top left and bottom right corners of the bounding box.

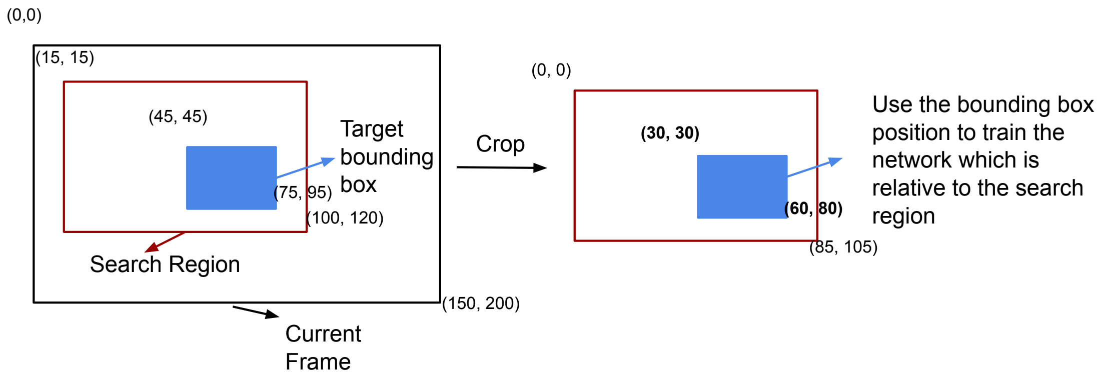

Figure 2: Bounding box of the target object relative to the search region.

Dataset

The dataset includes 263 training, 25 validation and 25 test videos. Each video folder includes the related frames as images. In the corresponding annotation folder of the video, each line consists of frame id and bounding box of the object as "frame_id x1 y1 x2 y2" where the coordinates gives the top left and bottom right corners of the bounding box.

1 Training

1. Prepare your data loader with consecutive frame pairs from training set of the dataset such as $(t_0, t_1), (t_1, t_2), (t_2, t_3)$ etc. Shuffle the loader so that the network will get random pairs of frames from random videos. You can use the same dataloader for validation as well. But for testing, the process will be different.

2. Crop the first frame, e.g. t_0, by using the padded ground-truth bounding box which is the 2 times enlarged version of it from the center. This way, you will get **what to track** with a context.

3. Extract features from VGG-16 pool5 layer (the last layer before FC layers). Note that you will not update the weights of VGG-16 since you are using it as a feature map extractor. Apply *global average pooling* to the output so that you can get $1x512$ vector (AvgPool2d in PyTorch). You will apply the previous (in PA2) data transformations such as resizing and normalization (not flipping). However, if you use random cropping, be sure that you are applying exactly **the same cropping**, in terms of cropping position, both to target object patch and the search region patch. Otherwise, the network cannot learn. In addition, you should find the new position of the **target's bounding box** in the search region since you are applying some transformations as resizing and cropping to the image, which will change the coordinates of the objects.

4. Crop the second frame, t_1, by using the search region which is the 2 times enlarged version of the previous frame's (t_0) bounding box from the center. This way you will get **where to look**.

5. Repeat the Step 3.

6. Concatenate those two features so that the input dimension will be $1x(512 + 512)$. Save those features for each pair in the entire dataset.

7. Use fully connected layers such as FC(1024,1024)-ReLU-FC(1024, 1024)-ReLU-FC(1024, 4) that maps input features to ground-truth bounding box of the second frame t_1 relative to the search region (see Figure 2), being an output vector in the size of $1x4$, which consists of the coordinates of the top left and bottom right corners of the bounding box. Therefore, the last FC layer will map the inputs to a $1x4$ vector. You will train only these fully connected layers (from scratch) in this assignment. For this part, you have to build a small network (FC layers) by yourself, so do a research about PyTorch network class, sequential module and forward function.

8. Your loss function will be MSE between the predicted coordinates of the bounding box and the ground-truth coordinates (which are relative to the search region).

9. You will update the weights with ADAM optimizer and plot the loss graphics both for validation and training.

   Bonus: Experiment with different network structures e.g.: Try to experiment with different size/number of FC layers, add dropout **(suggested)** between the layers etc. But you should **report and examine** each experiment that you conduct in detail both for the entire assignment and the bonus part.

   Warning: Before training your network, definitely check the training/validation batches by visualizing the target object, the search region and the bounding box coordinates of the target on the search region, since the network will map the inputs to these coordinates. Be careful about two things: (1) target object's bounding box should be fixed as relative to the search region (Figure 2), (2) the coordinates should be arranged according to the resized/cropped search region as well. Therefore, visualize the bounding box of the target object over the final input to check your data (see Figure 3).

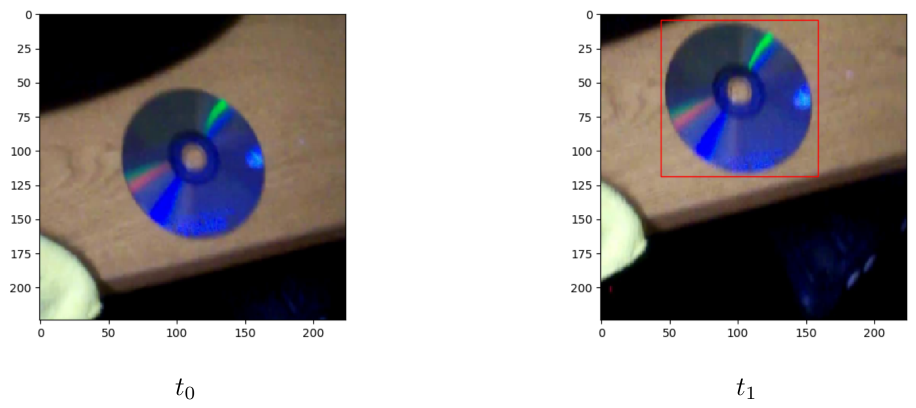

Figure 3: The enlarged object patch on t_0 and the search region with the target object position on t_1.

Note: First of all, you may extract the features of the frames and save them to reduce complexity. After that, your data loader can load the related features of the frames for the training part. Otherwise, for each epoch, you will extract the same features from the VGG-16 network several times which will be a slow process.

2 Testing

1. For each test video, you will initialize the bounding box of the first frame from the ground-truth. Do not shuffle the test pairs.

2. Then, your network will predict the bounding boxes of the rest. In other words, for other frame pairs, you will use the previous frame's **predicted** bounding box locations both for padded bounding box from Section 1 Step 2 and the search region from Section 1 Step 4.

3. This way, for each test video, your network will start with a ground truth bounding box (target object), and hopefully will track it along the frames.

3 Evaluation

- Give MSE loss graphs for the training process (training and validation). For the test set, give the average MSE of all pairs of frames from test videos.

- You should give visualized predictions of 3 test videos (the best, the average and the worse result) where you put the predicted bounding boxes on each frame. Upload those videos/GIFs to your drive folder and give the viewable link in your report. Make comments for each result.

The Implementation Details

1. You should pay attention to code readability such as comments, function/variable names and your code quality: 1) no hard-coding 2) no repeated code 3) cleanly separate and organize your code 4) use consistent style, indentation 5) write deterministic algorithms
2. Implement your code with Python 2.7 and use libraries from Anaconda. You can install any library that is not in Anaconda as well, such as OpenCV.
3. You should use the latest PyTorch as the deep learning framework. You can use Google Colab to run your experiments.

What should you write in the report?

- Give explanations for each step.
- Give experimental/visual results, used parameters and comments on the results in detail.
- Give your model's loss plot both for training and validation set during training.
- Put the results of different hyper-parameters (learning rate, batch size), the effect of them, with the loss plots and visualized bounding box predictions.
- A basic structure might be: 1) Introduction (what is the problem, how do you approach to this problem, what is the content of your report) 2) Implementation Details (the method you followed and details of your solution) 3) Experimental Results (all results for separate parts with different parameters and your comments on the results) 4) Conclusion (what are the results and what are the weaknesses of your implementation, in which parts you have failed and why, possible future solutions)
- You should write your report in LaTeX
- You should give visual results by using a table structure.

What to Hand In

Your submission format will be:

- README.txt *(give a text file containing the details about your implementation, how to run your code, the organization of your code, functions etc.)*
- code/ *(directory containing all your code)*
- report.pdf

Archieve this folder as **b<studentNumber>.zip** and submit to `https://submit.cs.hacettepe.edu.tr`.

Academic Integrity

All work on assignments must be done individually unless stated otherwise. You are encouraged to discuss with your classmates about the given assignments, but these discussions should be carried out in an abstract way. That is, discussions related to a particular solution to a specific problem (either in actual code or in the pseudocode) will not be tolerated. In short, turning in someone else's work, in whole or in part, as your own will be considered as a violation of academic integrity. Please note that the former condition also holds for the material found on the web as everything on the web has been written by someone else.