Visual Recognition

Aaditya Singh
160002

Rohin Garg
160583

Object Tracking using Self Supervised Algorithms

April 17, 2019

1 Online Object Tracking

In this assignment we undertook the task of online object tracking via detection. In online tracking we only have the current and previous frames available unlike offline tracking where we have the future frames available as well.

2 Baseline

2.1 High-Speed Tracking-by-Detection Without Using Image Information

One of the simpler algorithms for object tracking via detection is [1], which uses detections from an object detection algorithm such as YOLOv3 [2] or Faster RCNN [3], and then takes the intersection over union (IOU) between the current tracks and current detections to predict the tracks for the next frame.

2.2 Simple Online RealTime Tracking

On the other hand, Simple Online RealTime Tracking (SORT) [4] is a slightly more complex but an enhanced tracking algorithm which also relies on the detections from YOLOv3 or Faster RCNN. It uses Kalman Filter to predict the future position of the current tracks and then minimizes the total IOU between those frames and the current detections using Hungarian algorithm.

SORT is one of the popular algorithms for object tracking which is widely implemented and easy to use, so we decided to use it as a baseline for our task. Specifically, we are using one such GitHub implementation of SORT [5] which uses YOLOv3 for detections.

3 Methods and Ideas tried

We used the following algorithms and ideas for object detection and tracking to try and improve upon the results of SORT. We visually analysed the outcome of the following methods (independent).

3.1 Conservation of object identities.

We tried to preserve the object-ids of the various tracks. SORT was not able to deal with this properly. Through minor changes, we were able to improve upon this.

3.2 Future Frame Prediction: Alternative to Kalman Filters

SORT uses Kalman Filter for the predicting future positions of the current track. We wanted to see if using the future frames predicted by Adversarial Video Generation improves upon this.

We created our own tracking pipeline which does not rely on Kalman Filters, and uses the frames predicted using the above model.

3.3 Domain Adaptation

Since the entire dataset is unlabeled, we thought of using domain adaptation to map the weights of YOLOv3, trained on COCO dataset, onto our dataset using backpropogation [8].

4 Conservation of object identities.

As our dataset consists of everyday traffic where distortions or sudden changes in the movements of various objects is quite likely, we found this implementation of SORT inconsistent in preserving the object-id's.

4.1 SORT implementation

In the original paper of SORT [4], IOUs are calculated between detections and predictions at every frame, and using them the detected objects are associated with the existing object tracks using the Hungarian Algorithm. If no object is associated with a track for T_Lost continuous frames,then the track is meant to be forgotten. The authors set this parameter T_Lost to 1. This fails to account for occlusion, or sudden movements of the object.

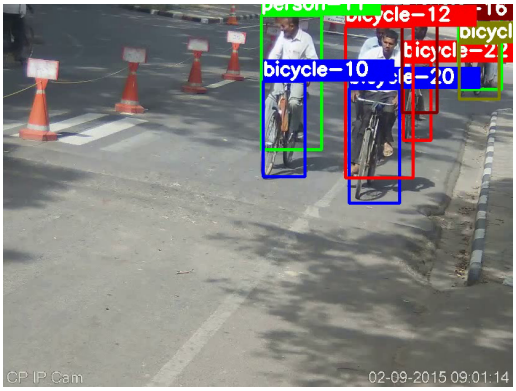

Figure 1: Before Speed Breaker

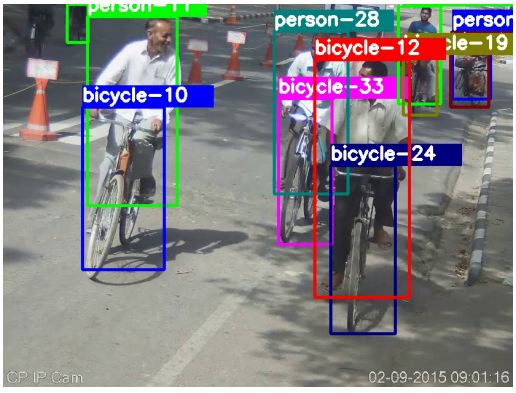

Figure 2: After Speed Breaker

4.2 Our Implementation

We changed the minimum number of frames for which a track remains active (in spite of no object associations) from 1 to 25 for preserving the object id's for a longer duration. This produced good results as the objects now retained their id's under the influence of distortions, specifically a speed-breaker for longer durations.

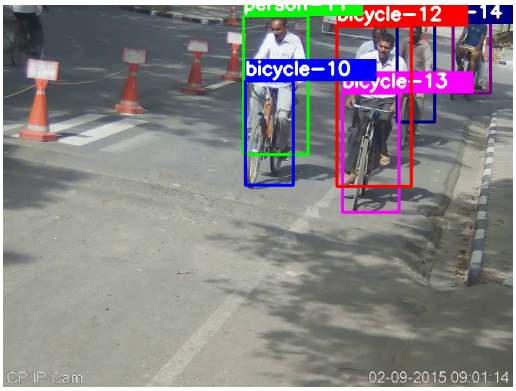

Figure 3: Before speed breaker

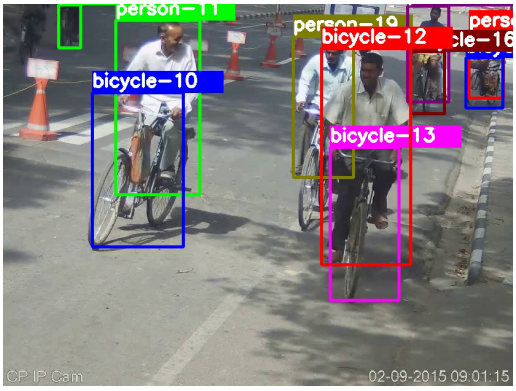

Figure 4: After speed breaker

5 Future Frame Prediction: Alternative to Kalman Filters

5.1 SORT implementation

SORT uses Kalman Filter for the predicting future positions of the current tracks with which it finds the IOUs of the current detections.
Kalman Filter approximates the inter-frame displacements of each object with a linear constant velocity model which is independent of other objects and camera motion. It keeps track of the state of an object as:

$$\mathbf{x} = [u, v, s, r, \dot{u}, \dot{v}, \dot{s}]^\top$$

where u and v represent the horizontal and vertical pixel location of the centre of

the target, while the scale s and r represent the scale (area) and the aspect ratio of the target's bounding box respectively.

The Kalman filter used in SORT assumes a linear velocity model for the various objects, which may not be true in reality. Moreover, it relies on a single past frame to do so, whereas it might be better to incorporate a few more past frames to predict the motion of the objects. Due to these reasons, we decided to predict the future frames instead of using Kalman filter.

5.2 Predicting future frames by Adversarial Video Generation

We learned how accurate and effective future frame prediction can be from the paper [6] and therefore we wanted to use it as a replacement for the Kalman Filter.

In this paper, multi-scale Generatve Adversarial Networks (GAN's) are used where the objective of the Generator is to produce a realistic estimate of the prediction while the Discriminator is being trained to weed out unrealistic examples. We train a combination of Generators and Discriminators at different scales so that they learn internal representations at various scales. The estimate at a lower scale is used as an input for the network at a higher scale.

Figure 5: Actual Frame

Figure 6: Predicted Frame

Figure 2: Multi-scale architecture

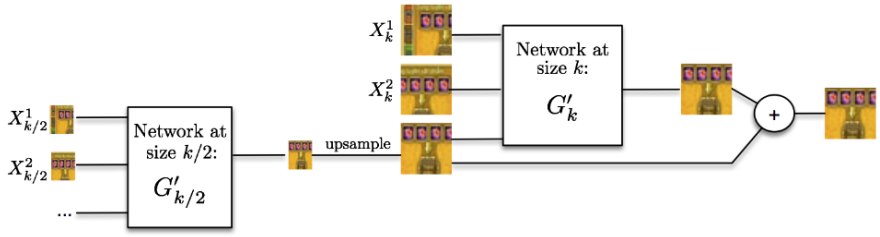

Figure 7: Multi-scale architecture Generative Adversarial Architecture.

5.3 Changes to the Tracking Algorithm

We changed the tracking algorithm to fit the future frame prediction:
For tracking on frame n:

- Using the previous 4 frames, we predict the possible n^{th} frame.
- Apply YOLOv3 detections on the predicted frame, to get the predicted bounding boxes.
- Associate these predicted boxes to the current tracks till $n-1$ frame using Hungarian algorithm on IOUs between the same.
- Now, on the actual frame, use YOLOv3 to detect the actual objects in the n^{th} frame.
- Use the Hungarian Algorithm to associate the detected boxes to the newly predicted future tracks using IOU.

The rest of the Algorithm is similar to the original SORT.

5.4 Implementation

We found a GitHub implementation of this paper [7], which we used to train the GAN on our dataset. In this paper, Generatve Adversarial Networks (GAN's) are used where the objective of the Generator is to produce a realistic estimate of the prediction while the Discriminator is being trained to weed out unrealistic examples.
We train a combination of Generators and Discriminators at different scales so that they learn internal representations at various scales. The estimate at a lower scale is used as an input for the network at a higher scale.

We found a GitHub implementation of this paper [7], which we used to train the GAN on our dataset.

5.5 Visual Analysis of the Output

We observed that this approach wasn't able to retain object id's for the same time as SORT. This was probably due to the fact that the predicted frames from the GAN's consisted of distortions which signifies moving objects, but YOLOv3 wasn't able to find good bounding boxes in such cases.

Moreover, the tracking was slower than SORT which was expected, as we are forward propagating approximately twice through YOLOv3 which consumes significant amount of time (for our implementation).

6 Unsupervised Domain Adaptation

SORT uses YOLOv3 pre-trained on COCO dataset. We wanted to train YOLOv3 on our dataset for obtaining better detections, but this wasn't possible as our dataset wasn't annotated. Therefore, we shifted to Domain Adaptation for this purpose.

The goal of domain adaptation is to transfer the knowledge of a model to a different but related data distribution. The model is trained on a source dataset and applied to a target dataset (usually unlabeled). In this case, the model is trained on regular COCO dataset, but we wanted to get good performance on our dataset. We learned about unsupervised Domain Adaptation from the paper [8].

The paper [8] proposes a deep feature extractor and a deep label predictor which together form a standard feed forward architecture. Unsupervised domain adaptation is achieved by adding a domain classifier, which is connected to the feature extractor via a gradient reversal layer that multiplies the gradient by a certain negative constant during the backpropagation based training.

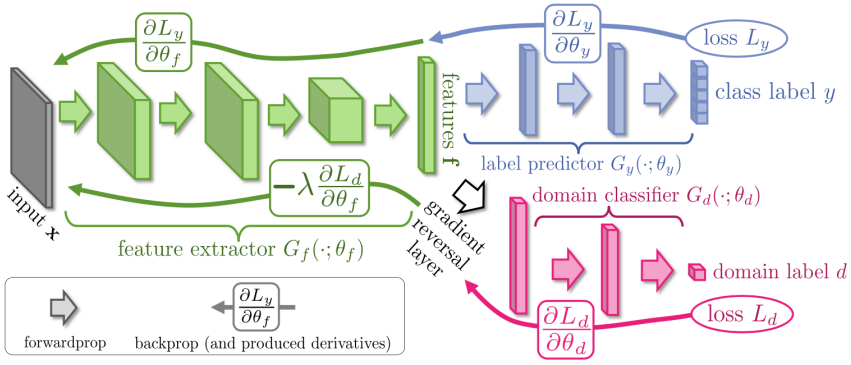

Figure 8: The proposed architecture for unsupervised domain adaptation.

We found a GitHub implementation of this paper [9] and used it to modify the weights of YOLOv3. We wanted to use these modified weights for the standard SORT. However, we found that the detections got worsened after using those weights.

7 References

[1] High-Speed Tracking-by-Detection Without Using Image Information

https://arxiv.org/abs/1506.01497

[2] YOLOv3: An Incremental Improvement

https://pjreddie.com/media/files/papers/YOLOv3.pdf

[3] Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8078516

[4] Simple Online And Realtime Tracking

https://arxiv.org/pdf/1602.00763.pdf

[5] GitHub: pytorch_objectdetecttrackg

https://github.com/cfotache/pytorch_objectdetecttrack

[6] Deep Multi-Scale Video Prediction Beyond Mean Square Error

https://arxiv.org/pdf/1511.05440.pdf

[7] GitHub: Adversarial_Video_ Generation

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.367.6279&rep=rep1&type=pdf

[8] Unsupervised Domain Adaptation by Backpropagation

https://arxiv.org/pdf/1409.7495.pdf

[9] Github: pytorchdomainadaptation

https://github.com/jvanvugt/pytorch-domain-adaptation