AI 537 Final Project Report

Wei-Chen Kao, Chi-Chieh Weng

1. Problem Statement

In this final project, we selected Multi-Object Tracking (MOT) as our topic. The purpose of this project is to give boundary boxes to each of our targeted objects in the first frame of the video and use the ByteTracker to track the subsequent video frame. The targeted objects in this project are the human body. Ideally, if there are 10 people in a frame, we will get 10 boundary boxes in that specific frame.

China implements the Social Credits System in their country. Under this system, the government is able to detect individual information, such as their credits, name, and honesty. When people walk pass in front of a monitor, the monitor can generate a boundary box and detect the specific person. This situation motivates us to think about how we can detect people in a scene. Since a scene can have more than a person showing up at the same time, this makes us want to do Multi-Object Tracking as our final project.

We used COCO (Common Objects in Context) as our dataset to train CNN (YOLOX). The reason we used COCO is that it has large-scale object detection, segmentation, and keypoint detection, and it consists of 328,000 images.

2. Approach

The whole approach of this project is to use ByteTracker to track multiple people in each frame. Before tracking, we have to use the dataset from COCO to train the CNN. After that, We utilized the BYTE (tracking BY associaTing Every detection box) method to detect the boundary boxes. In the BYTE algorithm, they separated the detection box into high scores (τ_high) and low scores (τ_low). They first consider the higher detection score threshold (τ_high) and store the fitting bounding box to the variable called Dhigh, then the bounding boxes which are lower than the threshold will be compared with the lower detection score threshold (τ_low). Then they used Kalman Filter to get the ground truth and get their bounding box value by their object detector. Through the IoU concept, they determine what tracks should be included in the next frame, and what tracks should be excluded. The pseudo-code of BYTE is on the next page.

Algorithm 1: Pseudo-code of BYTE.

Input: A video sequence V; object detector Det; Kalman Filter KF; detection score threshold τ_{high}, τ_{low}; tracking score threshold ϵ

Output: Tracks $\mathcal{T}$ of the video

Initialization: $\mathcal{T} \leftarrow \emptyset$
for *frame* f_k *in* V **do**
  /* Figure 2(a) */
  /* predict detection boxes & scores */
  $\mathcal{D}_k \leftarrow \text{Det}(f_k)$
  $\mathcal{D}_{high} \leftarrow \emptyset$
  $\mathcal{D}_{low} \leftarrow \emptyset$
  for d *in* $\mathcal{D}_k$ **do**
    if $d.score > \tau_{high}$ **then**
      $\mathcal{D}_{high} \leftarrow \mathcal{D}_{high} \cup \{d\}$
    end
    else if $d.score > \tau_{low}$ **then**
      $\mathcal{D}_{low} \leftarrow \mathcal{D}_{low} \cup \{d\}$
    end
  end
  /* predict new locations of tracks */
  for t *in* $\mathcal{T}$ **do**
    $t \leftarrow \text{KF}(t)$
  end
  /* Figure 2(b) */
  /* first association */
  Associate $\mathcal{T}$ and $\mathcal{D}_{high}$ using IoU distance
  $\mathcal{D}_{remain} \leftarrow$ remaining object boxes from $\mathcal{D}_{high}$
  $\mathcal{T}_{remain} \leftarrow$ remaining tracks from $\mathcal{T}$
  /* Figure 2(c) */
  /* second association */
  Associate $\mathcal{T}_{remain}$ and $\mathcal{D}_{low}$ using IoU distance
  $\mathcal{T}_{re-remain} \leftarrow$ remaining tracks from $\mathcal{T}_{remain}$
  /* delete unmatched tracks */
  $\mathcal{T} \leftarrow \mathcal{T} \setminus \mathcal{T}_{re-remain}$
  /* initialize new tracks */
  for d *in* $\mathcal{D}_{remain}$ **do**
    if $d.score > \epsilon$ **then**
      $\mathcal{T} \leftarrow \mathcal{T} \cup \{d\}$
    end
  end
end
Return: $\mathcal{T}$

3. Evaluation

Implementation details:

Libraries: PyTorch, OpenCV, YOLOX,

Open Source Code: https://github.com/ifzhang/ByteTrack

Thresholds: 0.7

Learning rate: 0.001

Epochs: 80

Batch size: 64

Dataset:

The video from MOT17 provided the ground truth of the bounding box, and we used two videos from MOT17 as our test videos. Since the human body is our only target, our class is equal to 1, which is human.

In our evaluation metrics, we used Intersection over Union detection divided by all frames and evaluated the average performance of all the bounding boxes. Based on the knowledge of IoU, we also use average confidence to determine whether the bounding box is trustworthy.

Simple versions of the approach:

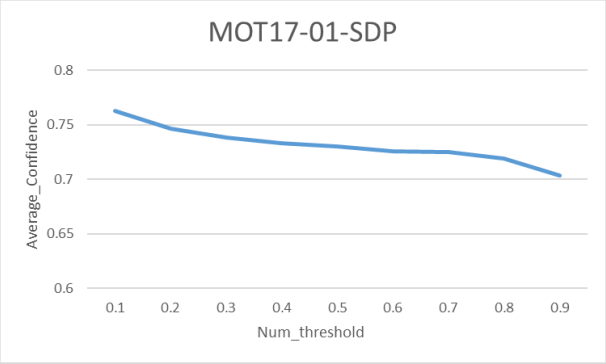
By testing a simple version, we use MOT17-07-SDP video as our testing model. According to the source code from GitHub, it provided 4 different kinds of models which are Bytetrack_x_mot17, Bytetrack_l_mot17, Bytetrack_m_mot17, and Bytetrack_s_mot17. In our experiments, we used these four ByteTrackers and compared their value of average confidence. The below table is the result that we ran based on the source code model. We can see that x_version has the best performance of confidence, while s_version has the worst performance of confidence. That is, using x_version will have 76.2% confidence to claim that the bounding box is corresponding to the ground truth bounding box.

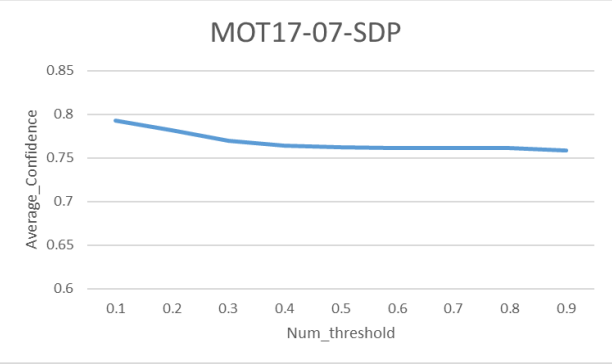
	x_version	l_version	m_version	s_version
Avg_Confidence	0.76205	0.759529	0.750963	0.733891

Qualitative evaluation:

In this part, we modify the value of Num_threshold, and this value is to determine how many bounding boxes we are going to have in each frame. The x-axis is the value of Num_threshold, and the y-axis is the average of the confidence. To obtain the value of confidence, it used the background knowledge of Intersection over Union. The confidence value is how many percent that the code believes the bounding box that we got was actually the ground truth bounding box.

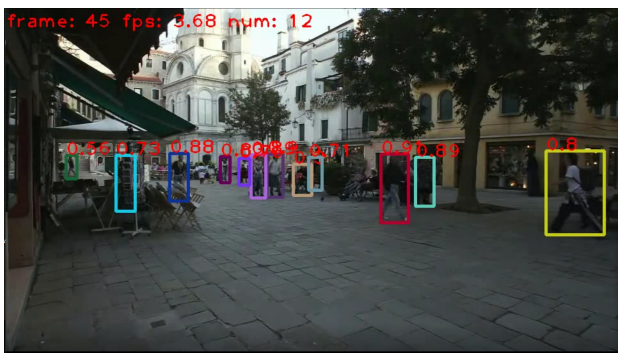
When we set the Num_threshold as 0.9, we can obviously notice that the value of the Average confidence is the lowest value compared to other thresholds. Since setting a higher Num_threshold is to get more possible bounding boxes from each frame, means some of the bounding boxes may have less confidence regarding the ground truth bounding box. In Figure 1 and Figure 2 on the next page, we can see that the number of bounding boxes is different even if they are in the same frame.





Computing Hardware:

The hardware requirements were proposed in the paper [2]. The model is trained on 8 NVIDIA Tesla V100 GPU with a batch size of 48, with an initial learning rate of 0.001, with 1 epoch warmup and cosine annealing schedule, and a total training period of roughly 12 hours.



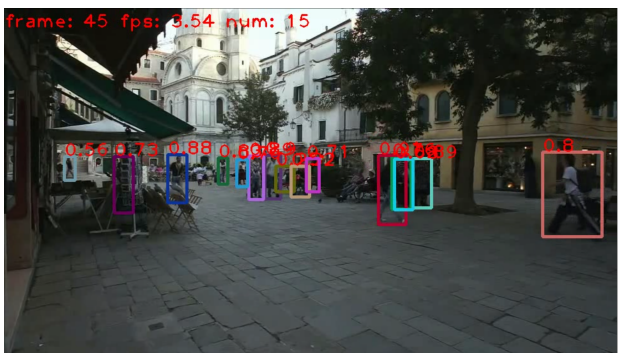
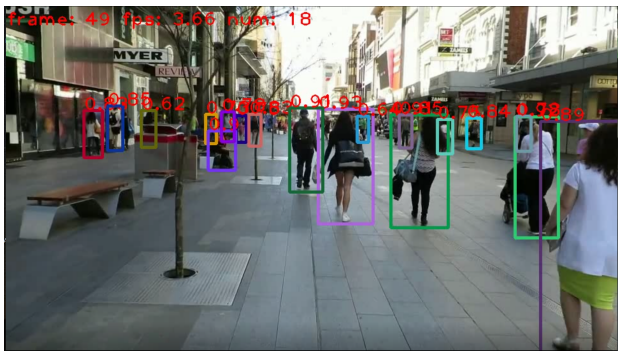


Figure 1. (MOT17-01-SDP with threshold 0.1)

Figure 2. (MOT17-01-SDP with threshold 0.9)



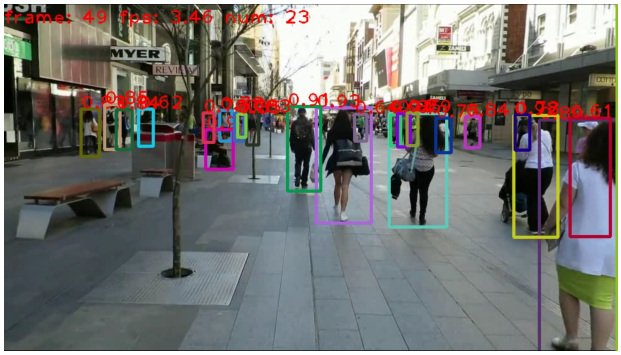


Figure 3. (MOT17-07-SDP with threshold 0.1)

Figure 4. (MOT17-07-SDP with threshold 0.9)

4. Team Contribution

Wei-Chen Kao: Researching Algorithm, Report Writing, Compare results

Chi-Chieh Weng: Environment setup, Report Writing , Find Dataset, Compare results

5. Reference

1. Lin, T., Maire, M., Belongie, serge , Bourdev, lubomir , Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. lawrence, & Dollar, P. (2015, February 21). *Microsoft COCO: Common Objects in Context*. https://arxiv.org/pdf/1405.0312.pdf
2. Zhang, Y., Sun, P., Jiang, Y., Yu, dongdong , Weng, F., Yuan, Z., Luo, P., Liu, W., & Wang, X. (2022, April 7). *ByteTrack: Multi-Object Tracking by Associating Every Detection Box*. https://arxiv.org/pdf/2107.08430.pdf
3. https://github.com/ifzhang/ByteTrack
4. https://paperswithcode.com/dataset/coco