SMAI PROJECT

TEAM 16 – HOW TO TRAIN YOUR DEEP MULTI-OBJECT TRACKER

TEAM MEMBERS

Aishwaryabharathi Upadhyayula	- 2020102060
Shriya Dullur	- 2020102006
Keerthi Pothalaraju	- 2020102010
Pallavi Pamulapati	- 2020101095

OBJECTIVE

Multi-object tracking (MOT) is a long-standing research problem. Object trackers are used to track the objects movement throughout a video sequence across multiple frames. dI;t has applications in autonomous driving and mobile robotics.

Through Tracking we become aware of surrounding object instances and can anticipate their future motion. Most of the existing methods to track objects uses detection to find the object instances in a frame and associate the responses over time to track the objects path.

Different methods are used to train object trackers, but most of them do not consider MOT evaluation measures.

The paper compares the following two multi-object tracking models

- ClearMot
- DeepMot

Multiple Object Tracking Using ClearMot

- The MOT’ s main task is to accurately estimate the bounding boxes for the object instances in the time frames. Then, associate these predicted boxes with the ground truths, over each time frame to track the path of the object.
- At a time-frame t, the predicted bounding boxes are compared with the ground-truth objects bounding boxes. These are compared using the Intersection Over-Union (IoU) distance metric and a distance matrix of size N*M is obtained. Where N is the number of bounding boxes(the rectangular boxes over every object) and M is the number of ground-truth objects in a time frame t.
- Optimal prediction to ground-truth assignment (a binary matrix) is obtained using the Hungarian Algorithm.
- Then we calculate the accuracy and precision of the model using popular MOT metrics like MOTA and MOTP. These metrics depend on the False Positives, False Negatives, Association matrix obtained from the Hungarian Algorithm, and Identity Switches in a time frame t.

Evaluation Metrics

- MOTA – Based on the mapping of predicted boxes and ground-truth boxes in each time frame. We compute True

Positives, False Positives, False Negatives, and identity switches that occur when tracker wrongfully swaps object identities or when a track is lost and is initialized with a different identity. MOTA measures accuracy based on these three errors.

$$\text{MOTA} = 1 - \frac{\sum_t (\text{FP}_t + \text{FN}_t + \text{IDS}_t)}{\sum_t M_t}$$

- MOTP – This averages the overlap between all correctly matches predictions and their ground truth.

$$\text{MOTP} = \frac{\sum_t \sum_{n,m} d_{tnm} a^*_{tnm}}{\sum_t |\, \text{TP}_t \,|}$$

Identity Switches – Object IDs might get swapped when two or more objects have similar attributes over time frames or when multiple objects coincide.

Data Set Used Mot16

Evaluating and comparing multi-target tracking methods is not trivial. One main reason for this is the ground truth, i.e., the perfect solution one aims to achieve, is difficult to define clearly. As the video may contain cropped targets, reflections in mirrors or windows, and objects that very closely resemble targets all impose intrinsic ambiguities, such that even humans may not agree on one ideal solution. Lack of pre-defined test and training data makes it difficult to compare different methods fairly.

MOTChallenge benchmark datasets are released to provide a fair comparison between tracking methods. In these MOT16 benchmark is used to evaluate the tracking methods (ClearMot and DeepMot). MOT16 contains a set of 14 sequences with more crowded scenarios, different viewpoints, camera motions, and weather conditions. And also, the annotations for all sequences are double-checked by researchers to ensure highest annotation accuracy.

Not only pedestrians but also vehicles, sitting people, and other significant object classes are annotated.

Data Format in MOT16

Each video in MOT16 contains 2 text files, one for public detections and another for ground truths. These files are simple comma-separated value (CSV) files. Each line in the public detections file resents one object instance and contains 9 values as shown.

```
1, -1, 794.2, 47.5, 71.2, 174.8, 67.5, -1, -1
1, -1, 164.1, 19.6, 66.5, 163.2, 29.4, -1, -1
1, -1, 875.4, 39.9, 25.3, 145.0, 19.6, -1, -1
2, -1, 781.7, 25.1, 69.2, 170.2, 58.1, -1, -1
```

The first number indicates in which frame the object appears, while the second number identifies that object as belonging to a trajectory by assigning a unique ID (initially, set to -1). The Id is unique as each object can be assigned to only one trajectory. The next four numbers indicate the position of the bounding box in 2D image coordinates. The position of the bounding box is indicated by the top-left corner as well as width and height of the box. And the next number denotes their confidence score. The last two numbers are ignored.

Similarly, for the ground truths file, the 7th value (confidence score) acts as a flag whether the entry is to be considered. If the value is 0 that instance can be ignored during the evaluation process. 8th number indicates the type of the object (i.e., label of the object). Last number shows the visibility ratio of each bounding box.

```
1, 1, 794.2, 47.5, 71.2, 174.8, 1, 1, 0.8
1, 2, 164.1, 19.6, 66.5, 163.2, 1, 1, 0.5
2, 4, 781.7, 25.1, 69.2, 170.2, 0, 12, 1.
```

Distance Matrix Computation

The most common metric used for measuring the similarity between two bounding boxes is IoU. IoU is a number from 0 to 1 that specifies the amount of overlap between the predicted and ground-truth bounding box. It is given by Area of overlap divided by the Area of Union of two bounding boxes.

- An IoU of 0 means that there is no overlap between the boxes
- An IoU of 1 means that the union of the boxes is the same as their overlap indicating that they are completely overlapping.

Thus, distance between the boxes is given by 1 – IoU. This is also known as Jaccard Distance.

Hungarian Algorithm

To get the optimal association matrix between the detected bounding boxes and ground truths in a time frame t, we use Hungarian algorithm. The association matrix obtained by using Hungarian algorithm A* can be calculated as

$$\mathbf{A}^* = \underset{\mathbf{A}\in\{0,1\}^{N\times M}}{\operatorname{argmin}} \sum_{n,m} d_{nm} a_{nm}, \quad \text{s.t.} \sum_{m} a_{nm} \leq 1, \forall n;$$
$$\sum_{n} a_{nm} \leq 1, \forall m; \; \sum_{m,n} a_{nm} = \min\{N, M\}.$$

In the above equations, d_{nm} is the (n,m)th element of the computed distance matrix. N is the number of detected bounding boxes and M is the number of ground truths in a frame t. The matrix obtained is a binary association matrix.

After obtaining A*, we calculate the number of False positives, True Positives, False Negatives, ID switches etc and find out the MOT metrics MOTA, MOTP.

Disadvantage of ClearMot

As these evaluation measures are not differentiable, the existing methods only optimize the trackers hyperparameters that maximize MOTA and MOTP. In this current version, gradient descent cannot be used to optimize the tracker as these evaluation metrics are not differentiable.

In the paper **How to train your multi-object tracker** a new model DeepMOT is proposed which uses Deep learning methods like Recurrent Neural Networks (RNNs), Hungarian networks. Multi-object tracking evaluating metrics like MOTA, MOTP are made differentiable (dMOTA, dMOTP) which are used to derive a loss function. This loss function is further used in the back-propagation of RNN.

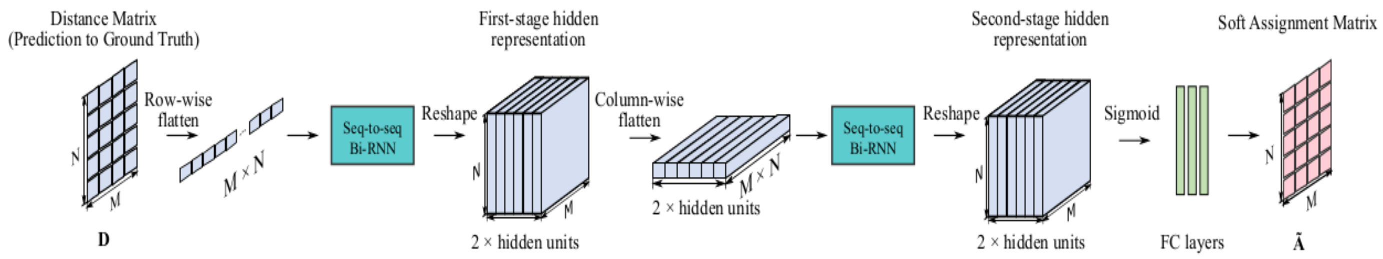
Multiple Object Tracking Using DeepMot

We follow a two-step strategy. First, we compute the soft assignment matrix using the deep Hungarian net algorithm. And then we approximate the CLEAR-MOT measures (MOTA and MOTP) as a combination of differentiable functions of the soft assignment matrix and distance matrix. Then used a combination of these parameters as the loss function to perform gradient descent.

Deep Hungarian Net (DHN)

DHN produces a soft assignment matrix (Ã) and is differentiable with respect to distance matrix (D). To accomplish this, we used two bi-directional recurrent neural networks (Bi-RNNs). For the first bi-rnn we pass the flattened distance matrix row wise, and it outputs the first stage hidden representation of size N*M*2h, where h is the size of the bi-rnn hidden layers. This gives the row-wise intermediate assignments. We then flatten this first stage hidden representation column-wise, to input to a second bi-rnn that produces the second stage hidden representation of size N*M*2h.

These two bi-rnns have the same hidden size but weights might be different. The output of the second bi-rnn is passed through the three fully connected layers and finally the sigmoid activation gives the soft assignment matrix of size N*M.



Distance Matrix Computation:

Jaccard distance is used along with Euclidean center-point distance in order to get a differentiable difference matrix. So that information can be back-propagated.

$$d_{nm} = \frac{f(\mathbf{x}^n, \mathbf{y}^m) + \mathcal{J}(\mathbf{x}^n, \mathbf{y}^m)}{2}$$

Here f is the Euclidean center-point distance normalized w.r.t image size and J is the Jaccard distance.

Differentiable MOTA and MOTP

To make MOTA and MOTP differentiable (i.e., dMOTA and dMOTP) we express FN, FP, IDS as functions of D and Ã. To find soft approximations of FP and FN, we construct matrices C^r and C^c by appending a column and a row to Ã respectively, filled with threshold value. Next, we perform the row-wise SoftMax and column-wise SoftMax. The resulting sum of the elements in $C^r_{n, M+1}$ and $C^c_{N+1, m}$ gives us the soft approximations of FP and FN respectively.

$$\tilde{\mathrm{FP}} = \sum_n \mathbf{C}^r_{n,M+1}, \quad \tilde{\mathrm{FN}} = \sum_m \mathbf{C}^c_{N+1,m}$$

To compute the soft approximations of IDS and MOTP, we construct two additional binary matrices B^{TP} and B_{-1}^{TP}, whose nonzero entries represent the true positives in the time frames t-1 and t respectively. To get the number of ID switches we take a l1 norm of the matrix obtained by the dot product of B^{TP} and binary complement of B_{-1}^{TP}.

From the above calculations we get dMOTA and dMOTP as:

$$dMOTA = 1 - \frac{\tilde{\mathrm{FP}} + \tilde{\mathrm{FN}} + \gamma \tilde{\mathrm{IDS}}}{M} \qquad dMOTP = 1 - \frac{\| \mathbf{D} \odot \mathbf{B}^{\mathrm{TP}} \|_1}{\| \mathbf{B}^{\mathrm{TP}} \|_0}$$

Since we should train the tracker to maximize MOTA and MOTP, the loss function used is

$$\mathcal{L}_{\mathrm{DeepMOT}} = (1 - dMOTA) + \lambda(1 - dMOTP)$$

Here λ is the loss balancing factor. We choose a large λ when precision of the model is the major concern and small λ value, when accuracy of the model is the major concern.

Training DeepMot

Step 1: Randomly sample a pair of consecutive frames from the training video sequences.

Step 2: The two images along with their ground-truth bounding boxes are considered as one training instance.

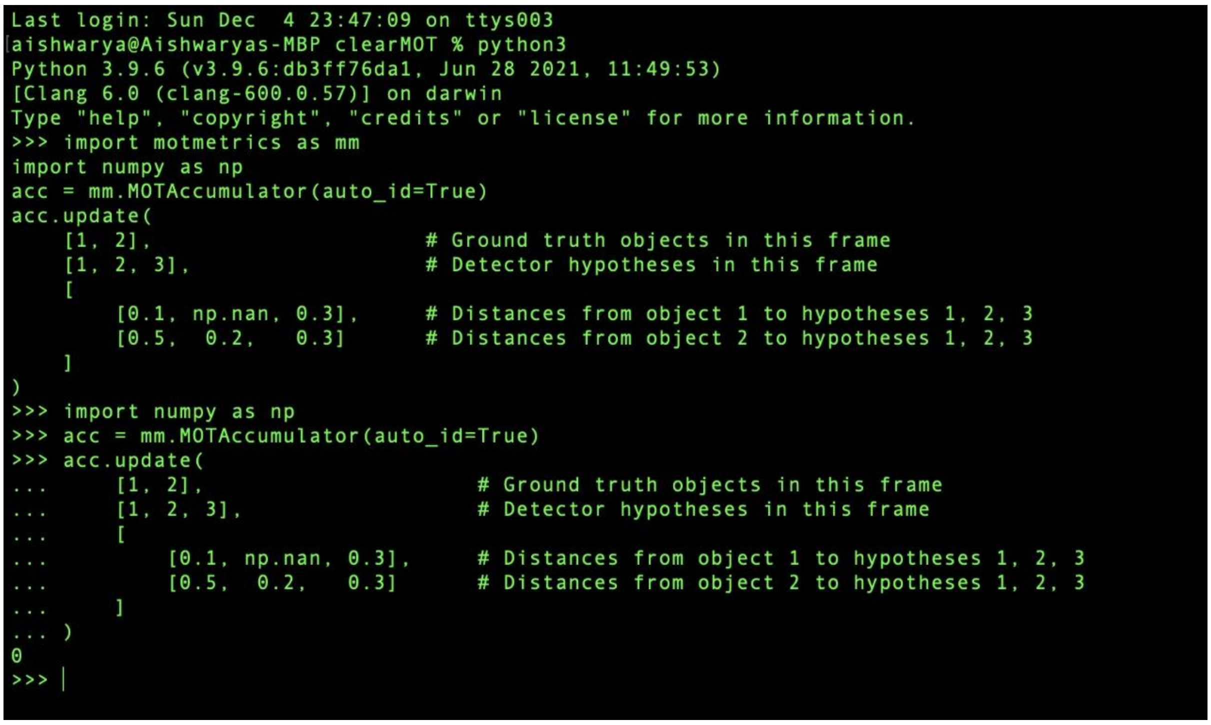
Step 3: Initialise the tracks with ground-truth bounding boxes at time t and forward pass them to get track’ s bounding box predictions

Step 4: Then, the distance matrix D is computed and use the DHN to compute the A˜

Step 5: The loss function is calculated and used to update the weights of the tracker.

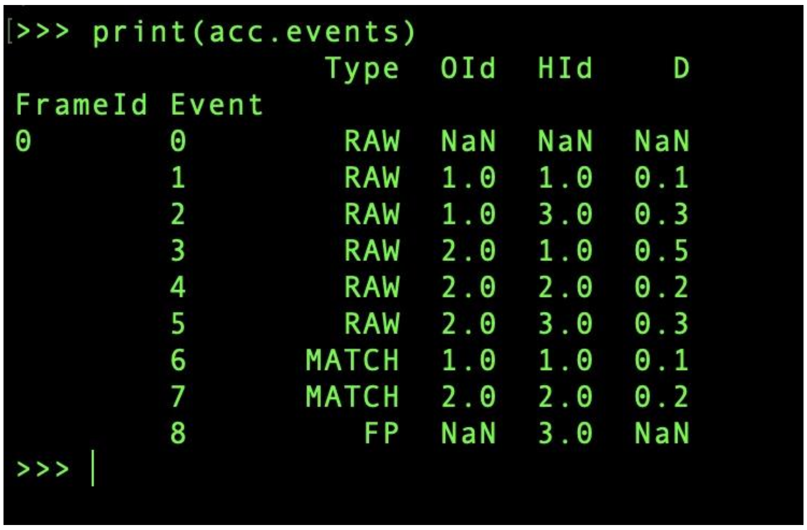
RESULTS

For a sample test case like this, giving inputs as follows :



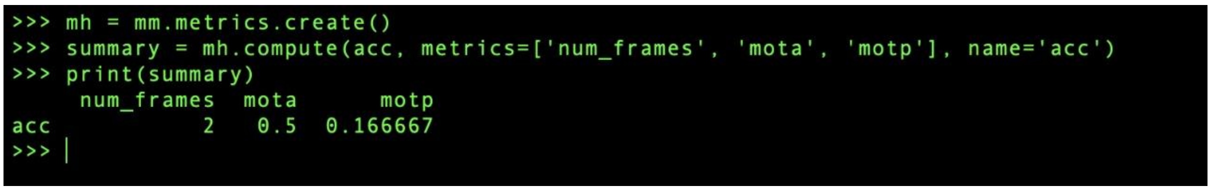
```
Last login: Sun Dec  4 23:47:09 on ttys003
[aishwarya@Aishwaryas-MBP clearMOT % python3
Python 3.9.6 (v3.9.6:db3ff76da1, Jun 28 2021, 11:49:53)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import motmetrics as mm
import numpy as np
acc = mm.MOTAccumulator(auto_id=True)
acc.update(
    [1, 2],                     # Ground truth objects in this frame
    [1, 2, 3],                  # Detector hypotheses in this frame
    [
        [0.1, np.nan, 0.3],     # Distances from object 1 to hypotheses 1, 2, 3
        [0.5,  0.2,   0.3]      # Distances from object 2 to hypotheses 1, 2, 3
    ]
)
>>> import numpy as np
>>> acc = mm.MOTAccumulator(auto_id=True)
>>> acc.update(
...     [1, 2],                     # Ground truth objects in this frame
...     [1, 2, 3],                  # Detector hypotheses in this frame
...     [
...         [0.1, np.nan, 0.3],     # Distances from object 1 to hypotheses 1, 2, 3
...         [0.5,  0.2,   0.3]      # Distances from object 2 to hypotheses 1, 2, 3
...     ]
... )
0
>>> |
```

Outputs are



```
[>>> print(acc.events)
                Type  OId  HId    D
FrameId Event
0       0        RAW  NaN  NaN  NaN
        1        RAW  1.0  1.0  0.1
        2        RAW  1.0  3.0  0.3
        3        RAW  2.0  1.0  0.5
        4        RAW  2.0  2.0  0.2
        5        RAW  2.0  3.0  0.3
        6      MATCH  1.0  1.0  0.1
        7      MATCH  2.0  2.0  0.2
        8         FP  NaN  3.0  NaN
>>> |
```

Mota ,motp values computed



```
>>> mh = mm.metrics.create()
>>> summary = mh.compute(acc, metrics=['num_frames', 'mota', 'motp'], name='acc')
>>> print(summary)
     num_frames  mota      motp
acc           2   0.5  0.166667
>>> |
```

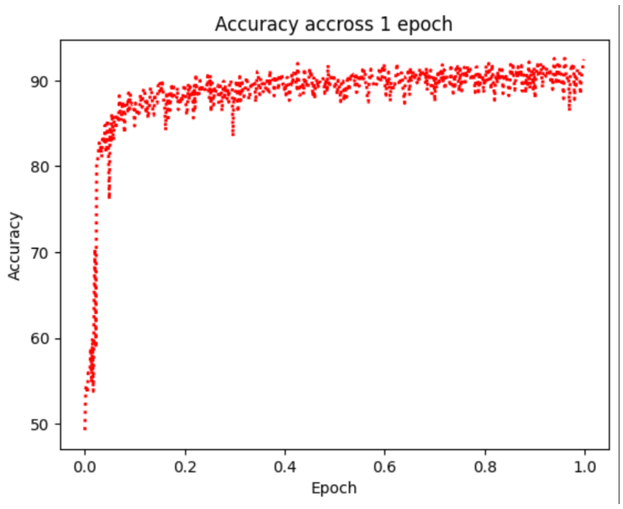
Accuracies obtained for MOT16 dataset using CLEARMOT

MOTA – 59.97

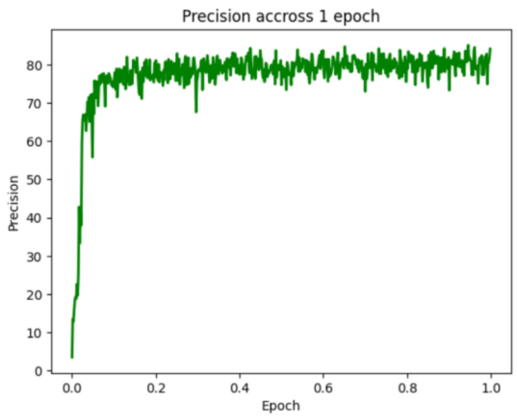
MOTP – 89.5

IDF1 – 70.84

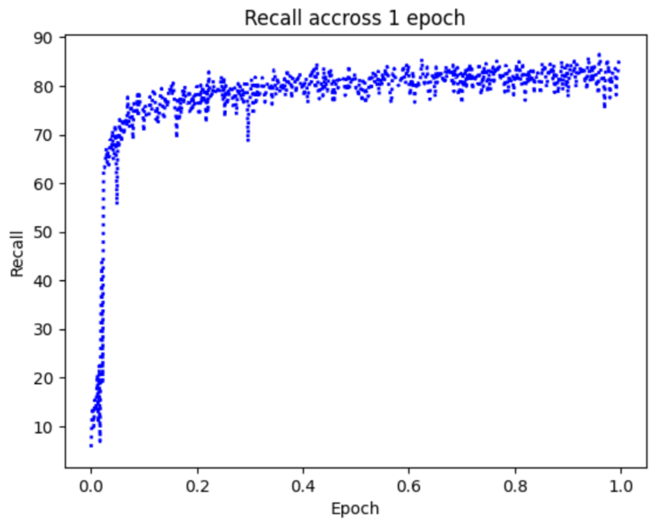
Accuracies obtained while training the DHN model on MOT16 data



Precision values obtained while training the DHN model



Recall values obtained while training the DHN model



WORK DIVISION

Data collection - Pallavi , Shriya

ClearMOT model – Aishwarya, Shriya, Pallavi

DHN model – Keerthi, Aishwarya, Shriya

Metric calculation and comparison – Pallavi, Aishwarya, Keerthi

Report and Readme – Pallavi, Shriya, Keerthi

PROJECT GITHUB LINK

https://github.com/Itreek/Team-16.git

REFERENCES

[1906.06618] How To Train Your Deep Multi-Object Tracker (arxiv.org)

https://motchallenge.net/data/MOT16/

https://arxiv.org/pdf/1603.00831.pdf

https://www.researchgate.net/publication/26523191_Evaluating_multiple_object_tracking_performance_The_CLEAR_MOT_metrics