**Towards Real-Time Multi-Object Tracking**

**Detailed Analysis**

A few pointers to summarize the details covered on 20/5/2020:

* There has to be some “detection-by-tracking” paradigm. What is this? How is it different from “tracking-by-detection”?
* Tracking-by-detection characteristics (basic idea):

1. Detection model for target localization: This would create a bounding box around each target per frame. Since there have to be multiple targets, multiple targets need to be localized.
2. Appearance embedding model for data association: That particular target will (can) have different locations in different frames. Now, for each frame, we’ll have a bounding box around the target. If somehow we can “associate” these detections so as to obtain a trajectory (path) of the object(s) of interest, we’re done!

Intuition regarding what model can be built for such a problem:

* Will only having a bounding box around each object work?
  + If we have an assumption: the relative speed of an object w.r.t. other object doesn’t change, then it “may” work. This we cannot assure. Henceforth, overlaps may arise and our detection algorithm will make inaccurate predictions like predicting the presence of a single object when overlaps arise.
  + Further, can we differentiate objects on the basis of, say, shape or size of the bounding boxes?
* These observations lead us to the conclusion that we need to include the features of whatever is enclosed in the box, and one of the components of our prediction would be prediction of what is there in the bounding box -> a **classifier**!
* Further, the bounding box may be predicted by the model at a location different from the ground truth. Solution? Simple, use a **regressor**.
* Tracking would involve association. This would involve taking our data to a higher dimension, the continuity perspective. Use an **embedding layer**.

We now have the components of our model. What can we now do?

* Detect and then track- We first train our model for detection, making use of a classifier for predicting the class of the object and regression for training the model for the coordinates of the bounding box. We then feed the predictions into an embedding neural network so as to obtain the trajectory w.r.t. the location of a particular object in different frames. But, this is basically a 2-step process, which does not take into consideration at all regarding what steps are repeating, and is not trainable in a single go. Hence, appropriate optimisation can be performed to reduce the running time to great extent.
* A one-after-other framework, in which we rather generate feature maps first, performing classification and regression, for we know that the relative location of the object would be same in the feature map, as in the original image. Then, we can feed the re-sampled features into the embedding layer. This is obviously better; the model can be trained and weights updated in a single go. Still, the scope of improving the run-time persists! After all, tracking has to be “real-time”.
* Can we still do it better? Can we take our embedding layer near classification and regression layer and perform association along with detection? This is what has been proposed in this research paper.

21/5/2020:

Weighting strategies:

Performance of multi-task learning systems is strongly dependent on the “relative weighting” between each task’s loss.

* Uniform (baseline) strategy: Assigns all the tasks equal (or, identical) loss weights.
* App. Opt (baseline): Uses a set of approximate loss weights; better than uniform strategy in our case, because embedding loss has a much larger scale than regression and classification losses.
* Uncertainty strategy: Weighs multiple loss functions by considering the homoscedastic uncertainty of each task.

Homoscedastic: The error term is the same across all values of the independent variables. (Is this the reason why in fig. 2, those loss functions which are getting fused are exactly the same, or is it just a typing mistake?- it’s a typing mistake)

* MGDA-UB: Multiple Gradient Descent Algorithm- Upper Bound

Multiple Gradient Descent Algorithm is used, however, it fares poorly as the number of tasks increases. To make it scalable, the authors (Sener & Koltun) proposed an upper-bound (UB) for the multi-task loss and optimised it.

* Normalisation Strategy: In this, the relative losses are normalised by dividing by the running deviation.

Q. Why is MGDA-UB strategy assigning too much weight to the embedding loss?

Q. Loss weights are automatically learnt => better tracking. What is the reason for it?

<http://openaccess.thecvf.com/content_cvpr_2016/papers/Shrivastava_Training_Region-Based_Object_CVPR_2016_paper.pdf> : About Online Hard-Example Mining