Chapter 1

Connecting Tubes

1.1 Description

As mentioned before, TPN gets as input a sequence of 16 frames and proposes TOIs. However, most actions in videos lasts more that 16 frames. This means that, in overlaping video clips, there will be consequentive TOIs that represent the entire action. So, it is essential to create an algorithm for finding and connecting these TOIs. Our algorithm is inspired by [], and uses a score in order to decide if a sequence of TOIs is possible to contain an action. This score is a combination of 2 metrics:

Actioness, which is the TOI's possibility to contain an action. This score is produced by TPN's scoring layers.

TOIs' overlapping, which is the IoU of the last frames of the first TOI and the first frames of the second TOI.

The above scoring policy can be described by the following formula:

$$S = \frac{1}{m}\sum_{i=1}^{m} Actioness_i + \frac{1}{m-1}\sum_{j=1}^{m-1} Overlap_{j,j+1}$$

For every possible combination of TOIs we calculate their score as show in figure 1.1. The above approach, however, needs too much memory for all needed calculations, so a memory usage problem is appeared. The reason is, for every new video segments we propose *k TOIs* (16 during training and 150 during validation). As a result, for a small video seperated in **5 segments**, we need to calculate **16^5 combinations at least**.

In order to deal with this problem, we create a greedy algorithm in order to find the candidates tubes. Inituitively, this algorithm after a new video segment keeps tubes with score higher than a threshold, and deletes the rest. So, we don't need to calculate combinations with very low score. This algorithm is described below:

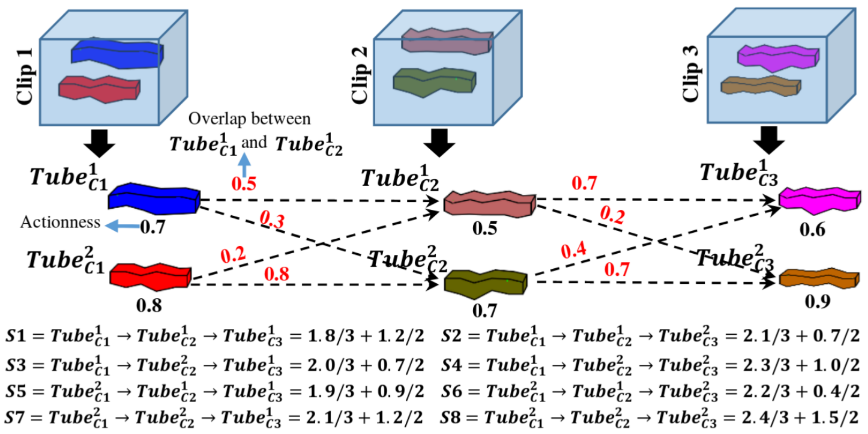
Clip 1
Clip 2
Clip 3
Overlap between
$Tube_{C1}^{1}$ and $Tube_{C2}^{1}$
$Tube_{C1}^{1}$
$Tube_{C2}^{1}$
$Tube_{C3}^{1}$
$Tube_{C1}^{2}$
$Tube_{C2}^{2}$
$Tube_{C3}^{2}$
Actionness
0.7
0.8
0.5
0.7
0.6
0.9
0.5
0.3
0.2
0.8
0.7
0.2
0.4
0.7
$S1 = Tube_{C1}^{1} \to Tube_{C2}^{1} \to Tube_{C3}^{1} = 1.8/3 + 1.2/2$
$S2 = Tube_{C1}^{1} \to Tube_{C2}^{1} \to Tube_{C3}^{2} = 2.1/3 + 0.7/2$
$S3 = Tube_{C1}^{1} \to Tube_{C2}^{2} \to Tube_{C3}^{1} = 2.0/3 + 0.7/2$
$S4 = Tube_{C1}^{1} \to Tube_{C2}^{2} \to Tube_{C3}^{2} = 2.3/3 + 1.0/2$
$S5 = Tube_{C1}^{2} \to Tube_{C2}^{1} \to Tube_{C3}^{1} = 1.9/3 + 0.9/2$
$S6 = Tube_{C1}^{2} \to Tube_{C2}^{1} \to Tube_{C3}^{2} = 2.2/3 + 0.4/2$
$S7 = Tube_{C1}^{2} \to Tube_{C2}^{2} \to Tube_{C3}^{1} = 2.1/3 + 1.2/2$
$S8 = Tube_{C1}^{2} \to Tube_{C2}^{2} \to Tube_{C3}^{2} = 2.4/3 + 1.5/2$

Figure 1.1: An example of calculating connection score for 3 random TOIs

1. Firstly, initialize empty lists for the final tubes, their scores, active tubes, their overlapping sum and actioness sum where:

    - Final tubes list contains all tubes which are the most possible to contain an action, and their score list contains their corresponding scores.
    - Active tubes list contains all tubes that will be match with the new TOIs. Their overlapping sum list and actioness sum list contain their sums in order to avoid calculating then for each loop.

    Also, we initialize threshold equal to 0.5 .

2. For the first video segment, we add all the TOIs to both active tubes and final tubes. Their scores are only their actioness because there are no tubes for calculating their overlapping score. So, we set their overlaping sum equal to 0.

3. For each next video, firstly we calculate their overlapping score with each active tube. Then, we empty active tubes, overlapping sum and actioness score lists. For each new tube that has score higher than the threshold we add to final tubes and to active tubes.

4. If the number of active tubes is higher than a threshold (1000 in our situation), we set the threshold equal to the score of the 100th higher score. On top of that, we update the final tubes list, removing all tubes that have score lower than the threshold.

5. After that, we add in active tubes, the current video segment's proposed TOIs. Also their actioness scores in actioness sum list and zero values in corrensponding positions in overlaps sum list (such as in the 1st step).

6. We repeat the previous 3 steps until there is no video segment left.

1.2 Some results