Chapter 1

Tube Proposal Network

One of the basic elements of ActionNet is **Tube Proposal Network**(TPN). The main purpose of this network is to propose **Tube of Interest**(TOIs). These tubes are likely to contain an known action and are consisted of some 2D boxes (1 for each frame). TPN is inspired from RPN introduced by FasterRCNN[**?**], but instead of images, TPN is used in videos as show in [**?**]. In full correspondence with RPN, the structure of TPN is similar to RPN. The only difference, is that TPN uses 3D Convolutional Layers and 3D anchors instead of 2D.

We designed 2 main structures for TPN. Each approach has a different definition of the used 3D anchors. The rest structure of the TPN is mainly the same with some little differences in the regression layer.

1.1 3D anchors as 6-dim vector

1.1.1 First Description

We started desinging our TPN inspired by [**?**]. We consider each anchor as a 3D bounding box written as $(x_1, y_1, t_1, x_2, y_2, t_2)$ where x_1, y_1, t_1 are the upper front left coordinates of the 3D and x_2, y_2, t_2 are the lower back left as shown in figure 1.1.

The main advantage of this approach is that except from x-y dims, dimension of time is mutable. As a result, the proposed TOIs have no fixed time duration. This will help us deal with untrimmed videos, because proposed TOIs would exclude background frames. For this approach, we use **n=4k=60** anchors for each pixel in the feature map of TPN. We have k anchors for each sample duration(5 scales of 1, 2, 4, 8, 16, 3 aspect ratios of 1:1, 1:2, 2:1 and 4 durations of 16,12,8,4 frames). In [**?**], network's anchors are defined according to the dataset most common anchors. This, however, creates the need to redesign the network for each dataset. We use the same anchors for both datasets, trying to generalize our approach. So the structure of TPN is:

- 1 3D Convolutional Layer

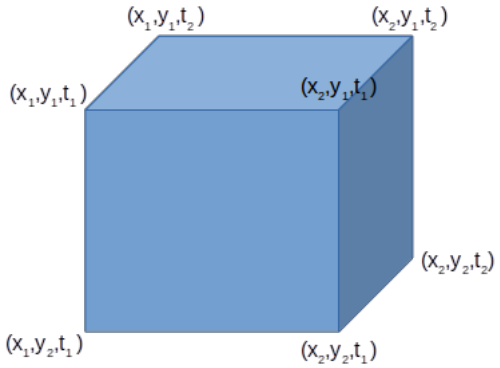

Figure 1.1: An example of the anchor $(x_1, y_1, t_1, x_2, y_2, t_2)$

- 1 classification layer outputs *2n scores* whether there is an action or not for *n tubes*.
- 1 regression layer outputs *6n coordinates* $(x_1, y_1, t_1, x_2, y_2, t_2)$ for *n tubes*.

which is shown in figure 1.2

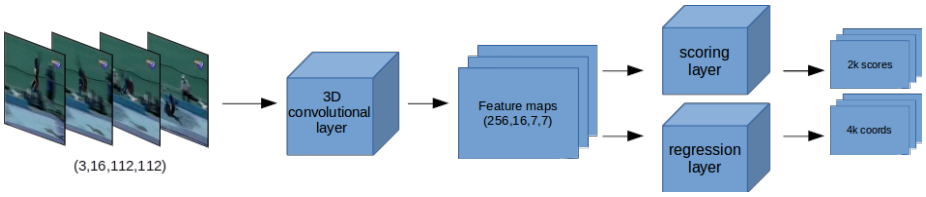

Figure 1.2: Structure of TPN

1.1.2 Training

As mentioned before, TPN extracts TOIs as 6-dim vectors. For that reason, we modify out groundtruth ROIs to groundtruth Tubes. We take for granted that the actor cannot move a lot during 16 frames, so that's why we use this kind of tubes. As shown in figure 1.3, these tubes are 3D boxes which include all the groundtruth rois, which are different for each frame.

For training procedure, for each video, we randomly select a part of it which has duration 16 frames. For each video, we train TPN in order to score all the anchors using IoU criterion (which is explained in next paragraph) and we use Cross Entropy Loss as a loss function. For regression, we use smooth-L1 loss. For regression targets, we use the same implementation as Faster R-CNN does,

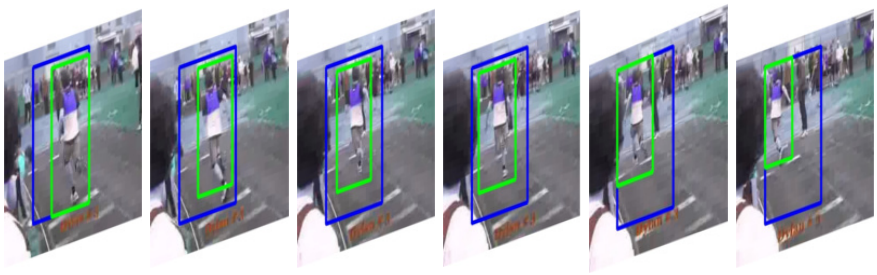

Figure 1.3: Groundtruth tube is coloured with blue and groundtruth rois with colour green

but for 3 domains again. So we have:

$$\begin{array}{ccc} t_x = (x - x_a)/w_a, & t_y = (y - y_a)/h_a, & t_z = (z - z_a)/d_a, \\ t_w = log(w/w_a), & t_h = log(h/h_a), & t_d = log(d/d_a), \\ t_x^* = (x^* - x_a)/w_a, & t_y^* = (y^* - y_a)/h_a, & t_z^* = (z^* - z_a)/d_a, \\ t_w^* = log(w^*/w_a), & t_h^* = log(h^*/h_a), & t_d^* = log(d^*/d_a), \end{array}$$

where x, y, z, w, h, d denote the 3D box's center coordinates and its widht, height and duration. Variables $x, x_a,$ and x^* are for the predicted box, anchor box, and groundthruth box respectively (likewise for y, z, w, h, d).

Modified Intersection over Union(mIoU) During training, we get numerous anchors. We have to classify them as foreground anchors or background anchors. Foreground anchors are those which contain some action. We need a criterion for evaluating them if we know the groundtruth tubes. We can see an extend of Intersection over Union criterion, which is used in Object Detection algorithms. So, we will consider as foreground those which have Intersection Over Union ≥ 0.5 .
One first approach would be to consider instead of areas, the volume of 2 candidate tubes. So IoU would be:

$$IoU = \frac{\text{Volume of Overlap}}{\text{Volume of Union}}$$

Intuitively, the above criterion is good for evaluating 2 tubes if they overlap but it has one big drawback: it considers x-y dimesions to have same importance with time dimension, which we do not desire. That's becase firstly we care to be accurate in time dimension, and then we can fix x-y domain. As a result, we change the way we calculate the Intesection Over Union. We calculate seperately the IoU in x-y domain (IoU-xy) and in t-domain (IoU-t). Finally, we multiply them in order to get the final IoU. So the formula for 2 tubes $(x_1, y_1, t_1, x_2, y_2, t_2)$ and $(x'_1, y'_1, t'_1, x'_2, y'_2, t'_2)$ is:

$$IoU_{xy} = \frac{\text{Area of Overlap in x-y}}{\text{Area of Union in x-y}}$$

$$IoU_t = \frac{max(t_1, t_1') - min(t_2, t_2')}{min(t_1, t_1') - max(t_2, t_2')}$$
$$IoU = IoU_{xy} \cdot IoU_t$$

The above criterion help us balance the impact of time domain in IoU. For example, let us consider 2 anchors: a = (22, 41, 1, 34, 70, 5) and b = (20, 45, 2, 32, 72, 5). These 2 anchors in x-y domain have IoU score equal to 0.61. But they are not exactly overlaped in time dim. Using the first approach we get 0.5057 IoU score and using the second approach we get 0.4889. So, the second criterion would reject this anchor, because there is a difference in time duration.

1.1.3 Adding regressor

The output of TPN is α-highest scoring anchors moved according to their regression prediction. After that, we have to translate the anchor into tubes. In order to do so, we add a regressor system which gets as input TOIs' feature maps and returns a sequence of 2D boxes, each for every frame. The only problem is that the regressor needs a fixed input size of featuremaps. This problem is already solven by R-CNNs which use roi pooling and roi align in order to get fixed size feature maps from ROIs with changing sizes. In our situation, we extend roi align operation, presented by Mask R-CNN, and we call it **3D Roi Align**.

3D Roi Align 3D Roi align is a modification of roi align presented by Mask R-CNN[]. The main difference between those two is that Mask R-CNN's roi align uses bilinear interpolation for extracting ROI's features and ours 3D roi align uses trilinear interpolation for the same reason. Again, the 3rd dimension is time. So, we have as input a feature map extracted from ResNet34 with dimensions (64,16,28,28) and a tensor containing the proposed TOIs. For each TOI, we get as output a feature map with size (64, 16, 7, 7).

On top of that, for each proposed TOI we give its feature map as input to a regressor. This regressor, returns $16 \cdot 4$ predicted translations, 4 for each frame. We keep only the predicted translations, for the frames that are $\geq t_1$ and $< t_2$.

Finally, for the frames, contained by the anchors, we set a 2D box (x_1, y_1, x_2, y_2) where x_1, y_1, x_2, y_2 are the reggressed values from the anchor, and the frames which are not contained, we set a zero-ed 2D box. The previous regressor is also trainable. It is consisted of 1 2D convolutional layer followed by a Relu function and another Linear Layer as shown in figure 1.4. After getting proposed TOIs from TPN, we pick, randomly, 16 tubes which will be input in the regressor. Finally, we find the traslation for each rois and, again, we use smooth-L1 loss for loss function.

1.1.4 Validation

Validation procedure is a bit similar to training procedure. We randomly select 16 frames from a validation video and we examine if there is at least 1

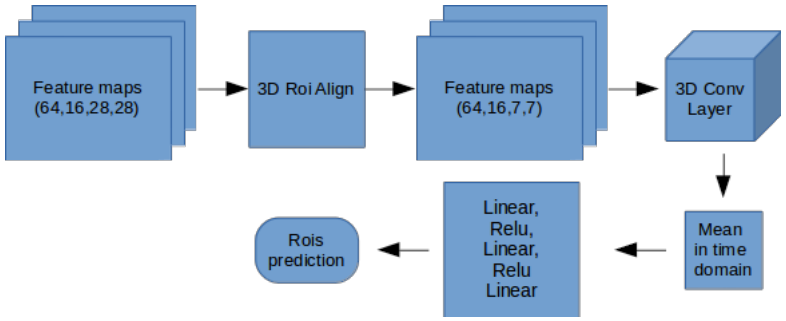

Figure 1.4: Structure of Regressor

proposed TOI which overlaps ≥ 0.5 with the groundtruth tubes. If there is, we consider this tube as True Positive (TP) else as False Negative (FN).

After we run one epochfor the whole validation dataset we calculate the **recall** metric which is:

$$recall = \frac{\text{TP}}{\mathbf{TP + FN}}$$

In order to count overlaps we use 2 kinds of Iou metrics.

1. 3D IoU which was mentioned before, which counts tubes' Volumes' Intersection of Union (without the modification we made during training).

2. the mean ROIs' IoU for each frame.

Respectively, we calculate 2 kinds of recall. The first one tells as how good were our proposed TOIs. The second one tells us how many tubes, we managed to detect during our proposals. **We count another one recall, which tells us how many from the good Proposed TOIs managed to correspond to an actual tube. In other words, this metric shows us the performance of the regressor.**

In order to get good proposal, after TPN we use Non-Maximum Suppresion (NMS) algorith. This algoriths removes at the proposed TOIs which have overlap ¿ 0.7 with the high-scoring proposed TOIs. As we can see in table , we get better recall scores when using NMS algorithm.

1.1.5 Improving TPN score

After first test, we came with the idea that in a video lasting 16 frames, in time domain, all kind of actions can be seperated in the following categories:

1. Action starts in the n-th frame and finishes after the 16th frame of the sampled video.

2. Action has already begun before the 1st frame of the video and ends in the n-th frame.

3. Action has already begun before the 1st frame of the video and finishes after the 16th video frame.

4. Action starts and ends in that 16 frames of the video.

On top of that, we noticed that most of actions, in our datasets, last more that 16 frame. So, we added 1 scoring layer and 1 reggression layer as shown in figure 1.5. These two layers have anchors with fixed time duration. Their purpose is to be trained only in x-y domain, keeping time duration steady.

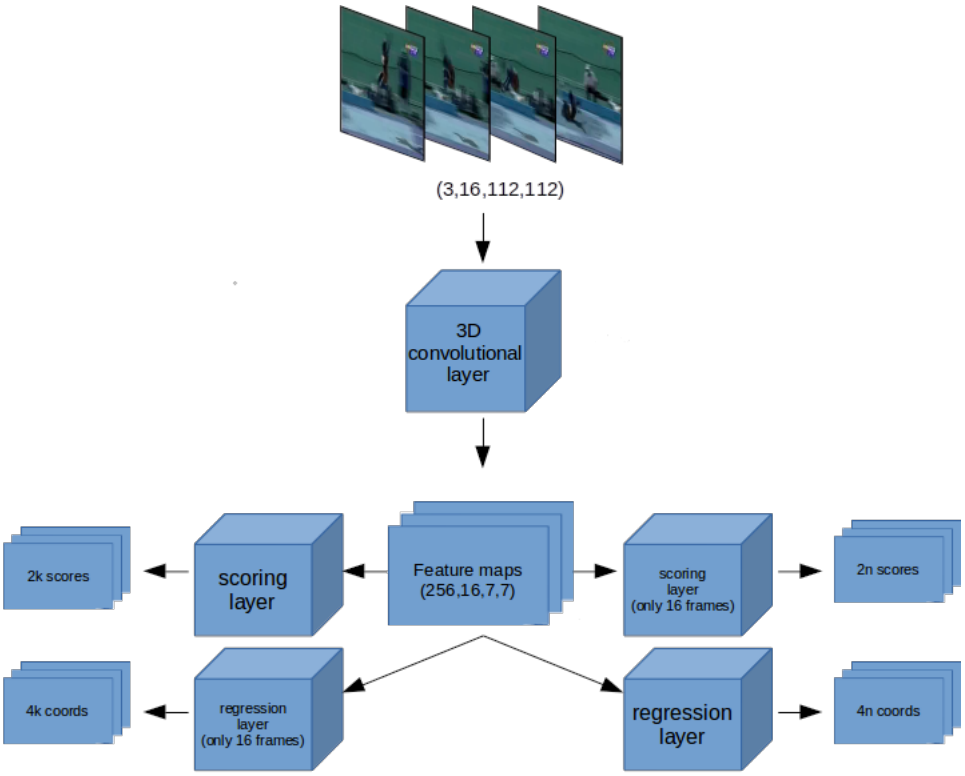

Figure 1.5: TPN structure after adding 2 new layers, where k = 5n.

Training and Validation procedures remain the same. The only big difference is that now we have from 2 difference system proposed TOIs. So, we first concate them and, then, we follow the same procedure. For training loss, we have 2 different cross-entropy losses and 2 different smooth-L1 losses, each for every layer correspondly. The regressor does not change at all.

1.1.6 Changing Regressor

As the above results show, when we translate a TOI into a sequence of ROIs, recall reduces about 20-30%. As a result, we should find a solution, in

order to deal with this proble.

From 3D to 2d The first idea we thought, was to change the first Convolutional layer from 3D to 2D. This means that we consider features not to have temporal dependencies for each fra. As we can see in the figure **??**, we got worse results, so, we rejected this idea.

Remove time-mean layer The second idea was to remove the mean layer in time dimension. This means that, the input of the first linear layer after the 3D convolutional layer gets as input all the features outputed from 3D convolution.

Use max pooling instead of mean layer As we noticed from the above figures, our system has difficulty in translating 3D boxes into 2D sequence of ROIs. So, that makes us rethink the way we designed our TPN.

1.1.7 Changing training procedure

Until now, we trained TPN and regressor together, using one total loss, which was the sum of all the sublosses. Now we use a new approach. At first, we train TPN for 40 epochs. Then, we freeze TPN and we train the regressor for 20 epochs.

1.2 3D anchors as 4k-dim vector

In this approach, we set 3D anchors as 4k coordinates (k = 16 frames = sample duration). So a typical anchor is written as $(x_1, y_1, x'_1, y'_1, x_2, y_2, ...)$ where x_1, y_1, x'_1, y'_1 are the coordinates for the 1st frame, x_2, y_2, x'_2, y'_2 are the coordinates for the 2nd frame etc as presented in []. In figure 1.6 we can an example of this type of anchor.

The main advantage of this approach is that we don't need to translate the 3D anchors into 2D boxes. However, it has a big drawback, which is the fact that this type of anchors has fixed time duration. In order to deal with this problem, we set anchors with different time durations, which are 16, 12, 8 and 4. Anchors with duration < sample duration (16 frames) can be written as 4k vector with zeroed coordinateds in the frames bigger that the time duration. For example, an anchor with 2 frames duration, starting from the 2nd frame and ending at the 3rd can be written as (0, 0, 0, 0, $x_1, y_1, x'_1, y'_1, x_2, y_2, x'_2, y'_2$, 0, 0, 0, 0) if sample duration is 4 frames.

This new approach led us to change the structure of TPN. The new one can is presented in figure 1.7. As we can see, we added scoring and regression layers for each duration.

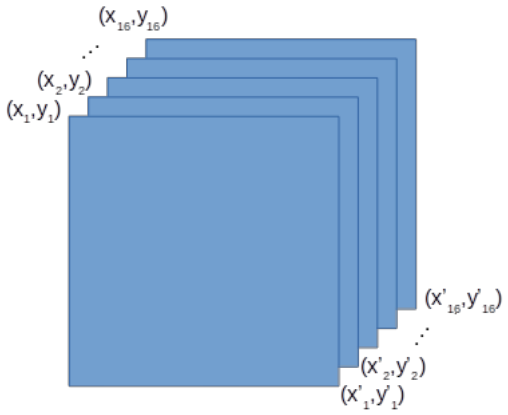

Figure 1.6: An example of the anchor $(x_1, y_1, x'_1, y'_1, x_2, y_2, ...)$

1.2.1 Training

In following figures we can see recall performance for sample duration = 16 when using max or avg pooling. From the above results, we can see that using max pooling achieves better results.

1.2.2 Adding regressor

In full correspondance with the previous approach, we added an regressor for trying to find better results. We

1.2.3 Trying to improve performance

1.2.4 Changing training procedure

1.2.5 Changing sample duration

After trying all the previous version, we noticed that we get about the same recall performances. So, we thought that we could try to reduce the sample duration. On top of that, we trained our network for sample duration = 8 and 4 frames.

.....

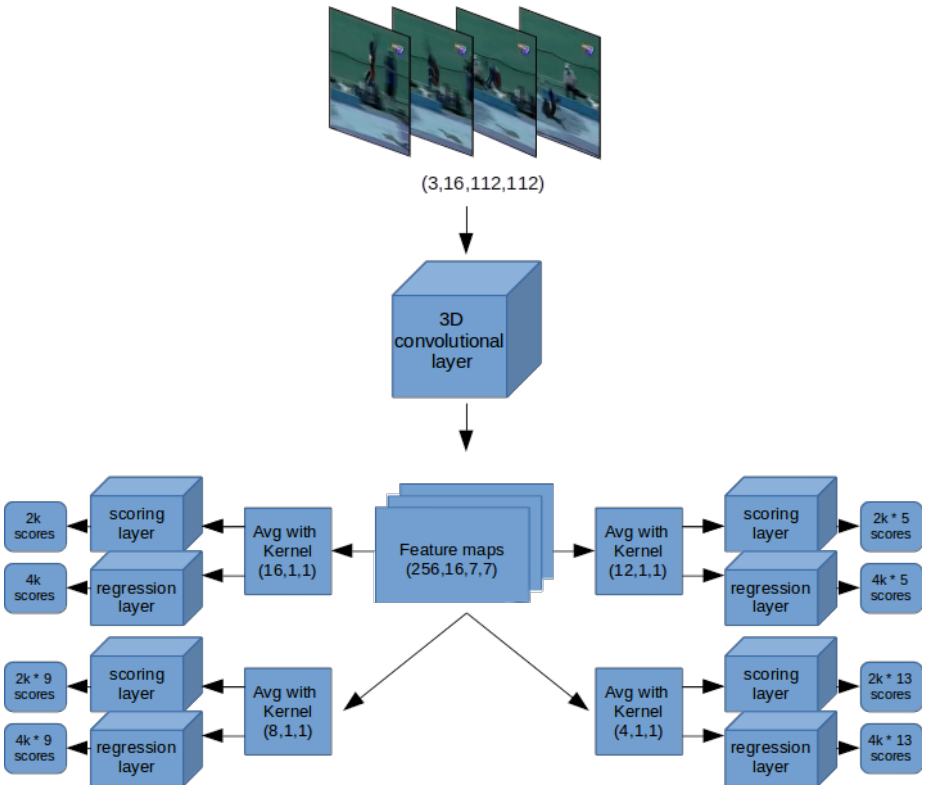

Figure 1.7: The structure of TPN according to new approach