State Of The Art Multi Object Tracking: An Overview

Gianmarco Cracco
gianmarco.cracco@studenti.unipd.it

Alessandro Manente
alessandro.manente.1@studenti.unipd.it

Abstract

After years of progress in the object detection and re-identification tasks, little attention was given on accomplishing the two in a single network to improve the speed of inference and accuracy. In this work we will try to give a better insight of the SOTA model, explaining in deep how it works.
At the end we will talk about our contribution that consist in a few experiments we managed to accomplish and the main weakness we found on the model, that is the lack of long term memory.

1. Introduction

Multi Object Tracking (MOT) has the goal to estimate trajectories of multiple moving objects of interest in videos. Until now the models with best performance were the ones that compute the bounding boxes separately for objects and the Re-ID features, but they are inevitably slow. With the maturity of multi-task learning, the one-shot methods have risen: since most features are shared for the two models, using a one-shot approach can dramatically reduce inference time.
Such models presented until now however suffer from the following three main problems:

1. **Use of Anchors:** anchors are at the basis of actual one-shot methods. However, experimental results show that they are not suited for learning Re-ID features for two main reasons: the first is that multiple anchors may be responsible for the identity of the same object; the second is, usually the feature map is downsampled by 8 times to balance accuracy and speed, and this results to be too coarse for Re-ID because of alignment problems. This is solved in [1], treating the task as a pixel-wise keypoint estimation, and identity classification on top of a high-resolution feature map.

2. **Multi-Layer Feature Aggregation:** Re-ID need to leverage low and high features level in order to handle both small and large object and through multi-layer aggregation this is made possible. This lead to a reduced number of identity switches.

3. **Dimensionality of the Re-ID Features:** previous Re-ID methods usually learn high-dimensional features, but it was found that lower-dimensional features are better for MOT, it helps reducing the risk of over-fitting to small data.

To overcome all these problems [1] adopts an $anchor - free$ object detection to estimate object centres on a high-resolution map, then a parallel branch estimating pixel-wise Re-ID features is used to predict objects' identities.
The resulting model is a complex neural network based on ResNet-34 with a variant of Deep Layer Aggregation (DLA) [2] on top of that. In addition, all convolution layers are replaced by Deformable Convolution Layer [3] in order to adapt to different scales and poses.
At the end of this network are attached 4, so called, *heads* responsible for the object detection and identity embedding.

2. Related Work

As said previously, the related works on MOT can be classified in 2 classes:

- **Two-Step method:** object detection and Re-ID are treated as two different tasks. A first CNN localizes all the objects of interest in the images with boxes, then an identity embedding network extracts Re-ID features and link the boxes.
  This is done using standard approaches for box linking that is: Intersection over Unions (IoU) (cf. Def.2), Kalman Filter and Hungarian Algorithm. The main advantage of this approach is that for each task the best model can be selected without compromises and obtaining best performances on public datasets, but this comes at the cons of slow performance because of the separate computations.

- **One-Shot Method:** exploits multi-task learning, so the idea is to share for the vast majority of the model the same weights and simultaneously accomplish object detection and identity embedding in a single network, this helps to reduce inference time. The main problem is that the tracking accuracy is usually lower

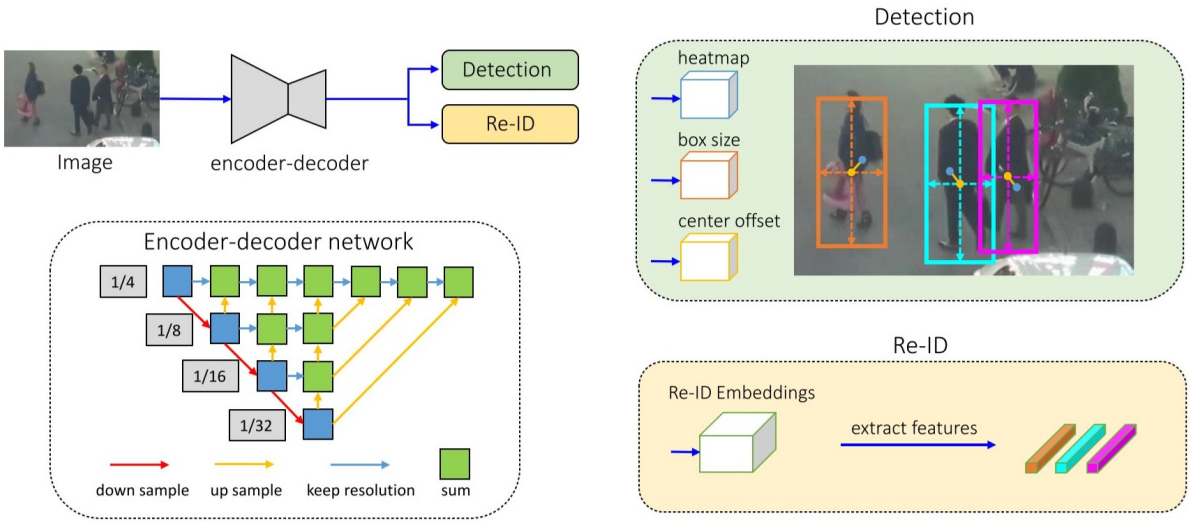

Figure 1: Overview of the proposed one-shot MOT model. The input is given to an encoder-decoder network to extract high resolution feature map, with a stride of 4. These outputs to two parallel heads to predict the bounding boxes and the Re-ID features. The predicted objects are fed to standard box linking techniques.

than the one obtained with two-step methods. The reason behind this is the anchor-based approach.

The way in which [1] addresses the problem is to implement a one-shot method that takes an anchor-free approach for detection and identity embedding.

3. Dataset

The data used by the authors of [1] to obtain SOTA results on MOT challenge is a combination of training images coming from 6 datasets for human detection, which are: ETH, CityPerson, CalTech, MOT17, CUHK-SYSU and PRW.
Each dataset is made of a train and validation set. They contain videos under the form of a sequence of images and for each image, which is a frame of the video, in an attached text file is given the centre and dimensions of the bounding box for each object of interest contained.
We also tried using these same datasets in our attempts to fine-tune the model but without success: we will explain better in Sec.5. For our final experiments, we used three different videos:

- **Real-world scenario:** similar to the ones used to train the model, it was chosen both for its limited duration (15 s) and because it presented a lot of moving obstacles in it, to try and test the lack of long term memory problem. More on this concept in Sec.5 and Sec.6; A small number of frame can be seen in Fig.2.
- **Videogame:** last video used is taken from a videogame with a medieval setting but still with a focus on people walking and standing in a city-like environment. Fig.4.
- **2D animation:** despite the objects being $2D$ characters the video represents people walking on the road, so the environment is similar to the training data; Fig.3.

4. Method

In a deeper way, we will report the model presented in [1]. As said previously it consists of a backbone network, called DLA-34 and four heads, three of them are responsible for the Object Detection Branch, while the remaining one is responsible for the Identity Embedding Branch.

4.1. Backbone

It has at its basis ResNet34 [4], since it gives a fair balance between speed and accuracy, using a combination of a deep architecture and skip-connections between layers. Then a first alteration is made substituting all convolutional layers with deformable convolutional layers [3], in order to increase the capabilities of the model to adapt to variations in scales and poses of the objects. At the end a variant of Deep Layer Aggregation (DLA) [2] has been applied. Both these two concepts are shown in Fig.5 and Fig.6 and better explained in section A.1 and A.2.

4.2. Heads

The main feature of the network is that the backbone network is shared between all the heads, making this model a

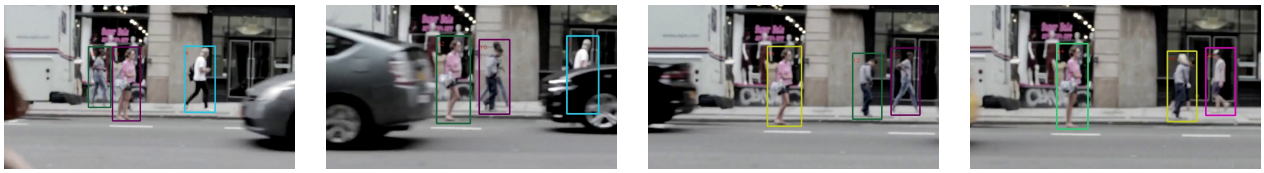

Figure 2: Evidences of lack of long term memory of the model. The woman standing in the background is here shown after her figure was covered by different object obstructing the view of the camera and it can be seen her ID switching in the sequence 10-12-14-24.

prime example of multi task learning.
The Object Detection Branch is made of three heads, which are implemented all in the same way: a $3x3$ convolution is applied to the output of the backbone network, followed by a $1x1$ convolution layer to generate final targets, and they are:

1. **Heatmap Head:** estimates the locations of object centres using heatmap based representation. The response in a given location is expected to be 1 if it coincides with the ground truth and decays exponentially with the increase of the distance between the object centre and the heatmap location.

2. **Centre Offset Head:** this improves the accuracy on the estimation of the localization of the objects. It is critical to achieve and sustain good performance of Re-ID.

3. **Box Size Head:** estimates the dimensions of the object bounding box. It is not directly related to Re-ID operations.

The **Identity Embedding Branch** is made of only one head, implemented with a convolutional layer with 128 kernels to extract identity embedding features for each location. It is responsible for generating features that can distinguish different objects.

4.3. Loss Function

The loss functions are the following:

- **Heatmap Loss:** for each box $b^i = \left(x_1^i, y_1^i, x_2^i, y_2^i\right)$ in the image, the object centre $\left(c_x^i, c_y^i\right) = \left(\frac{x_1^i + x_2^i}{2}, \frac{y_1^i + y_2^i}{2}\right)$ is computed and the location on the heatmap $\left(\tilde{c}_x^i, \tilde{c}_y^i\right)$ is obtained dividing the stride of 4. The heatmap response at location (x, y) is $M_{xy} = \sum_{i=1}^{N} \exp^{-\frac{\left(x - i_x^i\right)^2 + \left(y - \tilde{c}_y^i\right)^2}{2\sigma_c^2}}$ with N number of objects in the image and σ_c the standard deviation. The loss is a pixel-wise logistic regression with focal loss (cf. Sec.A.3:

$$L_{\text{heatmap}} = -\frac{1}{N} \sum_{xy} \begin{cases} \left(1 - \hat{M}_{xy}\right)^{\alpha} \log\left(\hat{M}_{xy}\right) & \text{if } M_{xy} = 1 \\ \left(1 - M_{xy}\right)^{\beta} \left(\hat{M}_{xy}\right)^{\alpha} \log\left(1 - \hat{M}_{xy}\right) & \text{otherwise} \end{cases}$$

$\hat{M}$ is the estimated heatmap while α, β are the parameters.

- **Offset and Size Loss:** for each box $b^i = \left(x_1^i, y_1^i, x_2^i, y_2^i\right)$ in the image, the size is $s^i = \left(x_2^i - x_1^i, y_2^i - y_1^i\right)$ and the offset is $o^i = \left(\frac{c_x^i}{4}, \frac{c_y^i}{4}\right) - \left(\left\lfloor \frac{c_x^i}{4} \right\rfloor, \left\lfloor \frac{c_y^i}{4} \right\rfloor\right)$. We enforce l_1 losses for the two heads:

$$L_{\text{box}} = \sum_{i=1}^{N} \left(\left\| o^i - \hat{o}^i \right\|_1 + \left\| s^i - \hat{s}^i \right\|_1 \right)$$

- **Identity Embedding Loss:** this task is treated as a classification one. All objects instances with same identity belongs to one class. For each box $b^i = \left(x_1^i, y_1^i, x_2^i, y_2^i\right)$ in the image a object centre on the heatmap is obtained. A feature vector E_{x^i, y^i} is extracted at location and is mapped to a class distribution vector $p(k)$. Given $L^i(k)$ the one-hot representation of the Ground Truth (GT) class label, the softmax loss is computed:

$$L_{\text{identity}} = -\sum_{i=1}^{N} \sum_{k=1}^{K} L^i(k) \log(p(k))$$

with K number of classes.

4.4. Online Tracking

The inference of the model and the box tracking is accomplished through the following pipeline: the network takes as input an image of size $1088x608$. Then given the predicted heatmap, non-maximum suppression (NSM) (cf. Def.1) is performed to extract peak keypoints and only those with heatmap scores larger than a threshold are kept. Bounding boxes are also computed based on estimated offsets and box sizes. Identity embeddings are also extracted. To achieve box linking a number of tracklets based on the estimated boxes are initialized. In the following frames, boxes are linked to existing tracklets according to distances

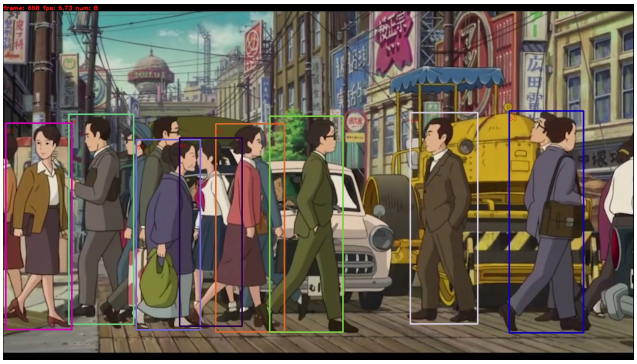

Figure 3: Video of animation, it can be seen that the model detects easily most objects although all training data were from real life footage.

Figure 4: Here the model has more difficulty than in the case on the left, although the objects are more real-life looking.

measured by Re-ID features and IoU's. Kalman Filter is used to predict locations of tracklets in current frame. If the distance is too big, the corresponding cost is set to infinity that prevents from linking the detections with large motion. To handle appearance variations, the appearance features of the trackers are updated at each time step.

5. Experiments

Our primary efforts in this experimental part were dedicated to fine-tune the model in order to obtain new and different results. Our plan was to try to play a little bit with both the parameters and hyper-parameters of the final layers contained in the heads, in such a way we could have drastically reduced the time required to train the model. To do so we managed to freeze all those layers, but unfortunately the code provided kept giving us errors making it impossible to train our model. In addition to that, obtaining our results on the test set would be very difficult since MOTchallenge requires to upload the results to their server in order to get the performance with the limitations of 3 days between an upload and the next one, for a maximum of 4 attempts. For these reasons and the lack of time we had to give up on trying to fine-tune the original model.
At the end, we tried running the `demo` function provided by the code of [1], so we fed the model with three specific video input and tried to study the results. More specifically:

- **Real-world scenario footage:** this video is emblematic to show the main problem with the model developed so far. As can be easily seen in it and from the frames in 2 it appears that the model suffers from the lack of long-term memory, in fact whenever something covers the woman standing in the background her ID changes.
  Changing the number of frames after which the linking algorithm delete a given unused tracklet, we were able to reduce the amount of ID switch changes for the woman in the background to 3, but still no appreciable result was obtained choosing a buffer size greater than 90 frames, in fact it would always switch the ID two times. This suggests that a better technique for linking should be implemented with a focus on the memory of it.

- **Videogame footage:** in this video can be seen that a lack of contrast and lack of difference in colours between people and background lead the model to often not recognise people or switch their IDs. However, it works quite well given the fact that the model never saw a clip coming from a videogame.

- **2D animation footage:** the model can follow $2D$ animated characters quite well, and shows problems only when the pose of characters is too far from the standing one.

The output videos can be downloaded from this link.

6. Lack of memory: future proposal

In this last section, we would like to propose a solution to the lack of long-term memory that affects the online linking approach used in [1].
We propose to substitute the online linking algorithm with a neural network for classification that takes as input the tracklets of the estimated boxes and classify them sequentially (the last layer will make use of softmax, in order to obtain a probability). Once all the tracklets from the same frame are processed by the network if a tracklet has been classified as a never seen before class (that is, no class reach a certain threshold of probability), the final layer is extended by one unit and random weights are initialised to that unit. Then, to update the classification capabilities of the network to the new information, the final layer is fine-tuned using the tracklets of the last frame as the training set.
In order to keep the focus of the network on identities that

are still present in the frames, during the training through the loss, we penalise the weights associated to classes that haven't been seen for at least a given number of frames. We suggest that the frames should be at least 60 (which equals to 2 s, assuming the default frame rate is 30fps) if the aim of the model is still to track people in urban environment, due to possible occlusions caused by passing cars or other people.
Furthermore to let the model forget about classes not present in the frames for a given threshold of time, the weights corresponding to those old classes should be set equal to 0.
Unfortunately we could not implement this proposal into the existing code due to the lack of computational power and time.

7. Conclusions

Here we presented an overview of a One-Shot anchor-free multiple object tracking. The reasons behind the failures of previous works are studied and found in the use of anchor-based approach. In fact, it is clear from the evidence that multiple anchors assigned to different parts of an object may be responsible for wrong estimation of identity.
Testing the model with various sources, we showed its strength but also its weakness, which is the lack of long-term memory, and for that we suggested a solution. We hope someone with better resources could implement it to try it out.

References

[1] Y. Zhang, C. Wang, X. Wang, W. Zeng, and W. Liu, "A simple baseline for multi-object tracking," 2020.

[2] F. Yu, D. Wang, E. Shelhamer, and T. Darrell, "Deep layer aggregation," 2017.

[3] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable convolutional networks," 2017.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[5] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," 2017.

A. Technical Appendix

A.1. Deformable Convolutional Layers

Deformable convolutional layers [3] adds an offset to the grid sampling location in standard convolution, and such offsets are learned through additional convolutional layers. While standard $2D$ convolution consists of sampling using a regular grid R over input feature map and then summation of sampled values weighted by w. So for each location p_0 on the output feature map y we have:

$$y(p_0) = \sum_{p_n \in R} w(p_n) \cdot x(p_0 + p_n)$$

where p_n enumerates locations in R.
In deformable convolution, the regular grid R is augmented with offsets $\{\Delta p_n | n = 1, \dots, |R|\}$ and so the previous equation becomes:

$$y(p_0) = \sum_{p_n \in R} w(p_n) \cdot x(p_0 + p_n + \Delta p_n).$$

It is worth noticing that the sampling is implemented as:

$$x(p) = \sum_q G(q, p) \cdot x(q)$$

$$G(q, p) = g(q_x, p_x) \cdot g(q_y, p_y)$$

where G is the bilinear interpolation kernel, $g(a, b) = max(0, 1 - |a - b|)$, and this is needed because the offset is typically fractional.

A.2. Deep Layer Aggregation

Once defined an aggregation [2] as the combination of different layers in the network, we say that a group of aggregations is *deep* if it is compositional, nonlinear and earliest aggregated layer passes through multiple aggregations. Layers are grouped into blocks and further grouped into stages by their feature resolution.
The Iterative Deep Aggregation (IDA) version is used in the model, so we will specifically talk about it.
We divide stacked blocks of the network into stages according to feature resolution. Aggregation begins at the shallowest, smallest scale and then iteratively merges deeper, larger scales. So shallow features are refined as they are propagated through different stages of aggregation.
The iterative deep aggregation function I of layers $x_1, \dots, x_n$ is defined as:

$$I(x_1, \dots, x_n) = \begin{cases} x_1 & \text{if } n = 1 \\ I(N(x_1, x_2), \dots, x_n) & \text{otherwise} \end{cases}$$

where N is the aggregation node defined in the following way:

$$N(x_1, \dots, x_n) = \sigma\left(\text{BatchNorm}\left(\sum_i W_i x_i + b\right)\right)$$

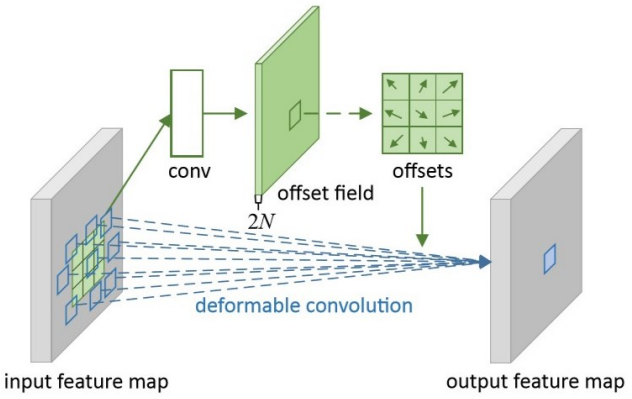

Figure 5: Illustration of deformable convolution network.

σ is the non-linear activation function while W_i and b are the weights in the convolution.
The main function of aggregation nodes is to combine and compress inputs, by learning to select and project important information to maintain the same dimension at their output as a single input.

A.3. Focal Loss

A modulating factor $(1-p)^{\gamma}$ is added to the cross entropy loss, with tunable focusing parameter $\gamma \geq 0$ [5]:

$$\mathrm{FL}\left(p\right) = -\left(1-p\right)^{\gamma}\log\left(p\right)$$

A.4. Network's Inference

The following concepts are used to perform inference:

Definition 1. Non-Maximum-Suppression - NMS
It is an algorithm used to filter proposed boxes for a given classification and its pseudocode is:

Algorithm *Non-Maximum-Suppression*

$B_{nms} \leftarrow \emptyset$
for $b_i \in B$ **do**
  $discard \leftarrow False$
  for $b_j \in B$ **do**
    if same$(b_i, b_j) > \lambda_{\mathbf{nms}}$ **then**
      if score$(c, b_j) > score(c, b_i)$ **then**
        $discard \leftarrow True$
  if not $discard$ **then**
    $B_{nms} \leftarrow B_{nms} \cup b_i$

The algorithm selects the proposal with highest confidence score e move it to a new void proposal list. It then compares the proposal with all the others through IoU and if this value is grater than a threshold it is removed from B. Repeat the process until B is empty.

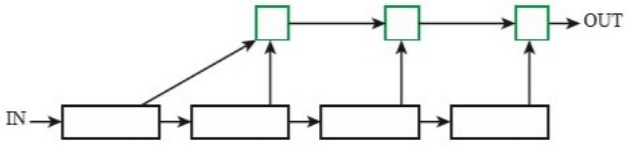

Figure 6: illustration of iterative deep aggregation (IDA).

Definition 2. Intersection over Union - IoU
Also called the Jaccard index, it is a measure of similarity of two sets and is computed as:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

and if A and B are both empty it is defined as $J(A,B) = 1$. Notice that it holds $0 \leq J(A,B) \leq 1$.