Object Model Adaptation for Multiple Object Tracking

Author
miquel.escobar@upc.edu

Advisors
cventuraroy@uoc.edu, {andreu.girbau, ferran.marques, xavier.giro}@upc.edu

Abstract

Multiple object tracking is a broadly used task in multiple applications, all the way from bioengineering to security applications. In this paper we propose a variation of RVOS [7] by adding the center estimation of detected instances, by means of a second head in the decoder which is assigned the task of detecting the corresponding object's bounding box arithmetic center. We have trained the model using three variants of the cross-entropy loss, which has been adapted to tackle the class imbalance caused by the fact that the center of an object is represented by only one pixel of the image, and have obtained some promising results.

1. Introduction

This work is product of the Challenged Based Innovation subject at UPC, which aims to introduce undergraduate students to the world of research, and has been developed with the mentorship of professors Ferran Marqués, Xavier Giró, Carles Ventura and PHD candidate Andreu Girbau. In fact, the implementation of this model has been built on RVOS [7] public repository, created by the mentioned researchers, and my task has consisted on adding the functionality of center detection parallely to the already implemented object segmentation and tracking.

Multi object tracking is the process of locating one or more moving objects in a video, while differentiating and identifying each object instance at each time step. MOT has gained popularity over recent years, with a spike of published papers and state-of-the-art models, direct consequence of the publication of multiple purposely made datasets.

Most multi object tracking models are based on object segmentation which is often translated into detecting each entity's mask separately []. In our case, we only concentrate on the arithmetic center of the bounding box of the instances, which is independent from the shape of the object at each time instant.

2. Related work

A series of deep learning techniques have been published over the last few years in the field of object detection, tracking and segmentation. The following are worth highlighting, due to their relationship with this work.

Multi object tracking.
As mentioned often through the entire report, the base architecture and inspiration of this work comes from the End-to-End Recurrent Network for Video Object Segmentation project [7] . Its architecture provides support for multiple cases, such as the zero-shot and one-shot scenarios. The model is based on RSIS [5], which consists of a recurrent neural network that predicts the mask of each instance at every time step, and it adds the recurrence to the temporal domain which provides more information to the model that is translated into better results.

In Figure 1 we observe the architecture of the neural network that we will modify to produce the centers predictions we are looking for.

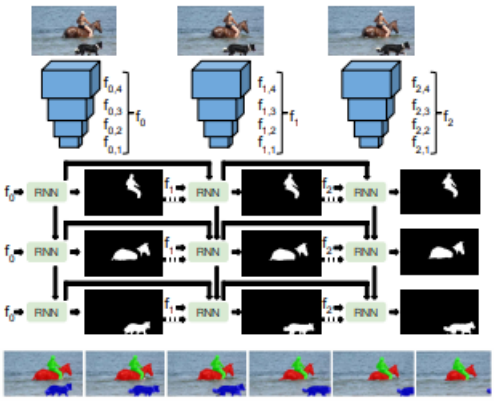


Figure 1: Architecture of the RVOS model.

Object detection.
There have been published various deep learning tech-

niques to detect objects, both represented as binary masks or, more commonly, as bounding boxes. Since we are working with a model that predicts the center pixel of the bounding box, we are more interested in the latter.

CornerNet [3] is one of these approaches, as it detects the object's bounding boxes as a pair of keypoints, the top-left corner and the bottom-right corner, and does it by using a single convolution neural network. By doing so, it can get rid of the anchor boxes used by other single-stage detectors. In figure 2 we observe an example of an output produced by their proposal.

Figure 2: Example of an output from the CornerNet model.

Another model that is worth mentioning is CenterNet [2], which is inspired in CornerNet: the idea they propose is to also explore the central part of a proposal, i.e., the region that is close to the geometric center, but with one extra keypoint. They extend the pair of keypoints into a triplet of them if a center keypoint is detected independently in the resulting bounding box's central region. Thus, detected objects are represented as a triplet of keypoints, as shown in Figure 3.

Figure 3: Example of an output from the CenterNet model.

3. Approach

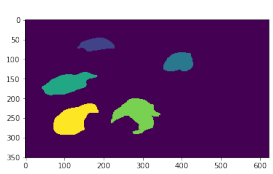
Our approach consists on representing the objects similarly to CornerNet and CenterNet, that is, simplifying the object's detection to a single keypoint corresponding to its center. This keypoint is the coordinates of the arithmetic center of the bounding box. Then, some transformations have been applied to make the data more balanced and tackle the class imbalance problem, as thoroughly explained in Section 3.1.

The idea is to add a new head to the pre-existing RVOS model in order to detect the center of each instance. Various implementations have been carried out, both based on the (ResNet-101 [1]) encoder's and the decoder's features, which are represented in Figure 1. After the first tests, we have disregarded the option of only using the encoder's features (see Section 3.3.1) as it did not result in good models and the executed model trainings did not seem to reduce the loss significantly. On the other hand, using the features obtained by the decoder (see Section 3.3.2) of the model resulted in successful trainings and much better results. Both MLP and convolutional layers have been added at the end of the decoder, but only the latter have resulted in the outputs we are looking for. The final chosen architecture for the center detection layer is explained in Section 3.3.2.

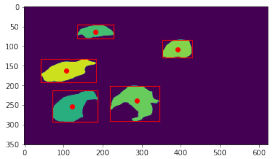
3.1. Center as a Gaussian distribution

The dataset we have used, Youtube-VOS [8], as detailedly explained in section 4.1, is composed by annotations of the masks of each instance. Since we want to work with the centers of the instances, which we will need represented as a keypoint, a transformation must be applied in order to obtain them.

We are taking the center of each instance as the arithmetic center of the mask's bounding box, that being $c_x = \frac{right-left}{2}$ and $c_y = \frac{top-bottom}{2}$, where right, left, top and bottom correspond to the coordinates of the bounding box. In Figure 4 we observe the transformation from masks into the bounding boxes and their corresponding centers, marked in red circles.



(a) Example of an instance ground truth mask.



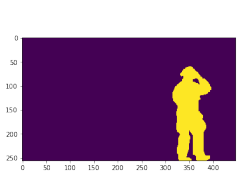
(b) The detected bounding boxes and their corresponding centers.

Figure 4: Computation of bounding box centers from object masks.

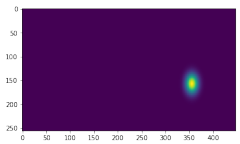
After computing the center, what we are left is with a ground truth of only one true pixel (the center), and the rest is set to false. This is very bad for training purposes, given such a huge class imbalance, and thus we tackle this problem by representing the center as a Gaussian distribution centered at the bounding box center and with the variance being to the width and height of the object, as expressed in equation 3.1.

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2} \tag{1}$$

Where x is the center 2D-vector (c_x, c_y), and σ is the covariance matrix $\sigma = \begin{pmatrix} w & 0 \\ 0 & h \end{pmatrix}$, being $w = right - left$ the width and $h = top - bottom$ the height of the object's bounding box. For instance, the left image of Figure 5 is transformed into the center representation of the right, where the distribution variance is higher on the y-axis than on the x-axis given the shape of the object.



(a) Example of an instance ground truth mask.



(b) The resulting Gaussian center representation obtained from the instance mask.

Figure 5: Computation of the center's Gaussian representation from the object's mask.

3.2. Loss functions

Various loss functions can be used for our purpose. We are treating with labels of continuous value, that is, the ground truth is composed of probabilities and so is non-binary. For a given instance, the model assigns the probability of each pixel being its center. We want to fit this probability to the value obtained off of the Gaussian center representation.

In order to do so, a lot of loss functions have been tested out:

- **Mean squared error.** It was tested as a first approach, but as expected it did not work properly and was hardly reduced during training. Given the nature of the data we are dealing with are probabilities $p \in [0, 1]$, this loss function was not expected to work. Furthermore, as it has been discarded for training, it is later used as a metric to measure the obtained results (see Section 4.2). Equation 2 shows how this loss function is computed.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{2}$$

- **Kullback–Leibler divergence.** Given the data we are treating consists of probability distribution, and this function allows us to compare two distributions, it seemed it could fit well for our purpose. In equation 3 we observe the computation of this loss, where p is the Gauss distribution centered on $\mu = (C_x, c_y)$ and with covariance $\sigma = \begin{pmatrix} w & 0 \\ 0 & h \end{pmatrix}$, and q is the 2D-prediction of the distribution.

$$L(q, p) = \sum_{i=1}^{N} p(x_i) \cdot \log \frac{p(x_i)}{q(x_i)} \tag{3}$$

$$L(q, p) = \sum_{i=1}^{N} y_i \cdot \log \frac{y_i}{\hat{y}_i} \tag{4}$$

- **Cross-entropy variants.** Three different variants of the cross-entropy loss have been regarded, which tackle differently the issue of class imbalance. In this case, the loss converged during training and the obtained results were satisfactory. It is worth noting that in order to implement this loss into the training, the three variants required binary masks, and thus the Gaussian probabilities of the ground truth had to be thresholded into booleans. In order to do so, the resulting Gauss distribution is normalized dividing it my its maximum value (which by definition is less than 1), and then thresholded by the value $t = 0.5$, obtaining the binary mask around the center of the object and still respecting the variance at each axis.

The three implemented cross-entropy variants, which have been the ones used for the final three models are the following:

Binary cross-entropy loss (BCE).
In this case, we could say that we "ignore" the class imbalance and simply use the basic binary cross-entropy loss. We would expect that the model trained with this loss predicts too many false negatives due to the class disparity.

$$BCE = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot p(y_i) + (1 - y_i) \cdot log(1 - p(y_i)) \tag{5}$$

Focal loss (FL).
In order to tackle the class imbalance problem, we substitute the binary-cross entropy with a focal loss. The original focal loss is defined in equation 6.

$$FL = -\alpha(1 - p(y_i))^{\gamma} \cdot log(p(y_i)) \tag{6}$$

We use the parameter $\gamma = 1$ and replace the α parameter with the computation of the mean between the losses of foreground and background pixels, which are computed separately, thus assigning the same relevance to each of the classes for each of the objects. This is computed as shown in equation 9.

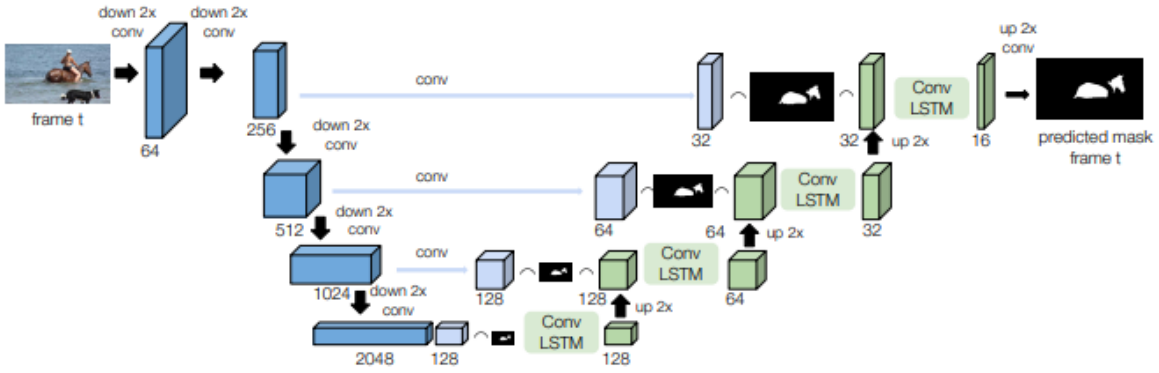


Figure 6: Architecture of the model's encoder and decoder. In the left, in blue, the components corresponding to the encoder. On the right and in green, the components corresponding to the decoder.

$$FL_0 = -\frac{1}{\sum_{i=1}^{N}(1-y_i)}\sum_{i=1}^{N}(1-y_i)\cdot log(1-p(y_i)) \tag{7}$$

$$FL_1 = -\frac{1}{\sum_{i=1}^{N} y_i}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) \tag{8}$$

$$FL = \frac{L_0 + L_1}{2} \tag{9}$$

Modified focal loss (Mod. FL).

In the two previous variants, the Gauss distribution per se is not being used completely. That is because we must threshold the values in order to obtain the needed binary mask. In this case, what we do is to ponderate the loss of each pixel by its distance to the center, which is denoted by the Gaussian ground truth distribution. Thus, for foreground pixels, we ponderate the loss by $g(y_i)$ (more important as it is closer to the center), and for background pixels by $1-g(y_i)$ (more important if it is further from the center). This is computed as shown in equation 12.

$$L_0 = -\frac{1}{\sum_{i=1}^{N}(1-y_i)}\sum_{i=1}^{N}(1-y_i)\cdot log(1-p(y_i))\cdot(1-g(y_i)) \tag{10}$$

$$L_1 = -\frac{1}{\sum_{i=1}^{N} y_i}\sum_{i=1}^{N} y_i \cdot p(y_i)\cdot g(y_i) \tag{11}$$

$$L = \frac{L_0 + L_1}{2} \tag{12}$$

3.3. Model

The model we propose is fully based on the RVOS architecture. The input consists of the set of RGB image frames of the video sequence, as well as the centers of the objects at the frame where each instance appears. These centers are obtained as explained in Section 3.1. Then the model passes the input to the encoder (see Section 3.3.1), generating a set of features as an output, which is then used as an input to the decoder of the model (see Section 3.3.2), which returns the prediction of the center assigning a probability to each pixel of the 2D-image. This architecture is shown in Figure 6.

Other model architectures have been tested out, such as using only the features of the decoder (by applying an MLP or convolution layer on the last feature),

3.3.1 Encoder

We use the architecture proposed by [7], which consists of a ResNet-101 [1] model pre-trained on ImageNet [4]. This architecture does instance segmentation by predicting a sequence of masks. The input x_t of the encoder is an RGB image, which corresponds to frame t in the video sequence, and the output $ft = \{f_{t,1}, f_{t,2}, ..., f_{t,k}\}$ is a set of features at different resolutions. The architecture of the encoder is illustrated as the on the left in 6.

3.3.2 Decoder

The decoder is composed by recurrent ConvLSTM [6] layers, which can treat the different resolutions of the input features $f_t = \{f_{t,1}, f_{t,2}, ..., f_{t,k}\}$, where $f_{t,k}$ are the features extracted at the level k of the encoder for the frame t of the video sequence. Each ConvLSTM layer also receives the mask of the previous frame at the corresponding resolution as an input.

The output of the decoder is a set of center distribution predictions $\{Ct, 1, , ..., Ct, i, ..., Ct, N\}$, where $C_{t,i}$ is the segmentation of object i at frame t, which are computed by

either applying a 2D-convolution to the last hidden state h and up-sampling it to match the input image resolution, or by combining all hidden states (except the last) of the decoder by using a convolution and up-sampling layer to each hidden state and combining them with the last one by applying a mean, and finally applying a 2D-convolution layer and an up-sampling to match the input image resolution, exactly like in the first option.

Out of the two variants to make the predictions $\{Ct, 1, , ..., Ct, i, ..., Ct, N\}$, the first one (which only uses the last hidden state) has obtained significantly better results, and thus all the results exposed in this document correspond to this model version.

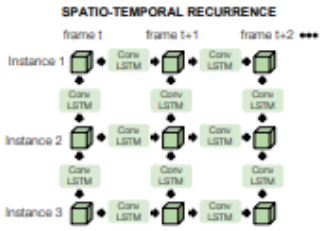


Figure 7: Architecture of the model's decoder, using an spatiotemporal recurrent network.

4. Experiments

4.1. Dataset

We use the YouTube-VOS dataset [8], the first large-scale benchmark that supports multiple video object segmentation tasks, including Semi-supervised Video Object Segmentation and Video Instance Segmentation.

It has the following features:

- 4000+ high-resolution YouTube videos
- 90+ semantic categories
- 7800+ unique objects
- 190k+ high-quality manual annotations
- 340+ minutes duration

4.2. Metrics

Once the models have been trained and we obtain predictions for the test dataset, these must be evaluated using objective metrics. A very important condition is that the metrics must be different than the loss functions used, in order for them to be more reliable. We want to measure how well does the model predict the center of the image: in order to do so, we use two metrics that compute the euclidean distance between the real center and the center of mass of our 2D-prediction, and one that measures the intersection over union between both. The latter is useful given that if we take the center of mass of our prediction, we are partly neglecting the distribution (two very differently distributed predictions can have the same center of mass): by computing the intersection over union, we can compare which of the models predicts better the regions of the foreground pixels, even though the metric in itself is not of much interest as we are interested in the center and not in a mask of the object.

Mean absolute error (MAE)

The first metric is to compute the mean absolute error for all predictions. In this case, the error is measured as the euclidean distance between the real center y_i and the predicted center $\hat{y}_i$.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{13}$$

Mean squared error (MSE)

This metric is computed in order to penalize the models which produce more outliers. A model with a lower MAE but a higher MSE would mean that it is better generally at predicting the center, but that it is also more prone to getting it completely wrong. Equation 14 shows how this measure is computed.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{14}$$

Intersection over Union (IoU)

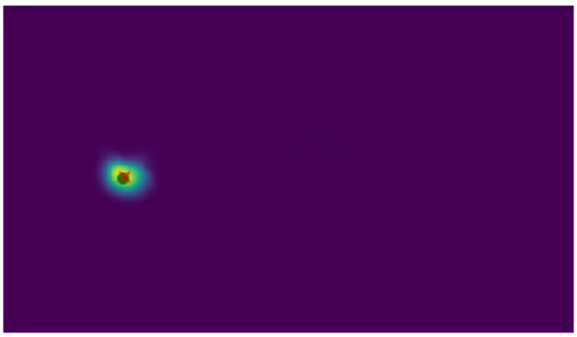
As explained at the beginning of this section, the IoU is used to analyze whether or not the distribution of the predictions match the ground truths, as the quality of the center measured by MAE and MSE only takes into consideration the center of mass of the prediction. Equation 15 shows how this metric is computed.

$$IoU = \frac{\text{Area of overlap}}{\text{Area of union}} = \frac{< y_i, \hat{y}_i >}{\sum_{i=1}^{N} y_i + \sum_{i=1}^{N} \hat{y}_i} \tag{15}$$

4.3. Results

We have trained the model for the three different losses exposed in Section 4.2 (BCE, FL and Mod. FL). The obtained results in table 1 show that the best model at fitting the position of the center is the one trained with the Focal Loss (FL), even though the one with Modified Focal Loss seems to pick up slightly better on the distribution of the center, as it presents a higher IoU. Both losses that have been customized for this case outperform the basic Binary Cross Entropy Loss.

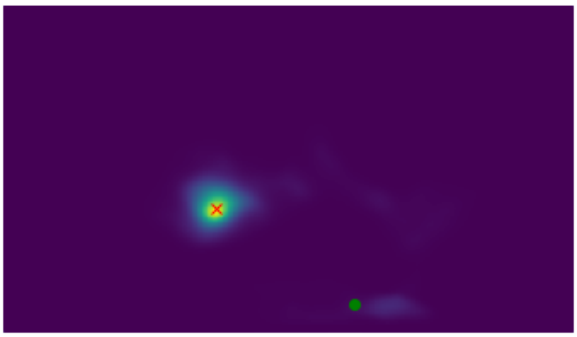
(a) Original RGB image.



(b) Predicted center distribution.

Figure 8: A good prediction of the center of the object by the FL model. The real center is marked with a dark green circle, the predicted center is marked with a red cross.

(a) Original RGB image.



(b) Predicted center distribution.

Figure 9: A bad prediction of the center of the object by the FL model. The real center is marked with a dark green circle, the predicted center is marked with a red cross.

	IoU	*MAE*	*MSE*
BCE	0.087	33.72	3376.09
FL	0.133	**28.98**	**2496.29**
Mod. FL	**0.149**	30.19	2583.05

Table 1: Results for each of the loss functions used in training the model.

It is worth noting that the treated images are of size 448×256, and thus an euclidean distance of ≈ 30 is relatively close. As a reference, the expected distance between two random points would be of around ≈ 260. Thus, the best model FL with $MAE = 28.98$ performs around 10 times better than a uniformly random model would.

In Figure 8 we can observe a good prediction made by the FL model, at a distance of $2px$, while in Figure 9 we observe a bad prediction of the same model at a distance of $140px$, which seems to be confused by another object of the same class.

5. Future work

Analyzing the obtained results from this work, there exists room for implementation of several modifications and improvement. We list the ideas that can be further developed from this work:

- Some short trainings have been executed for a model combining the original task of RVOS [7] (multiple object segmentation) and the center detection, by combining their losses. These have reduced the loss function during training, indicating that there exists potential for using the center detection head as an auxiliary task that could improve the task of the original model.

- Changing the task to the detection of not only the center, but also the bounding box of the object, similarly to [2] and [3].

6. Conclusions

In this work it has been shown that the model presented by [7] can be successfully modified to perform other tasks than multi object segmentation. It has also been shown that treating the problem of class imbalance with a proper loss function that adapts well to it can translate into significantly better results.

Finally, I would like to add that this introduction to the world of research has enormously helped me to learn about the difficulties you can find when exploring a path which is not marked and how to tackle them, together with dealing with real-world cases and data, which require scalable and reproducible implementations of the used code.

7. Acknowledgements

I would like to end this report by thanking both the collaboration of Ferran Marqués, Xavier Giró, Carles Ventura and Andreu Girbau throughout the development of all the work and the resources provided by UPC's Image Processing Group.

References

[1] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks, 2016.

[2] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. Centernet: Keypoint triplets for object detection, 2019.

[3] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints, 2019.

[4] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015.

[5] Amaia Salvador, Miriam Bellver, Victor Campos, Manel Baradad, Ferran Marques, Jordi Torres, and Xavier Giro i Nieto. Recurrent neural networks for semantic instance segmentation, 2019.

[6] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai kin Wong, and Wang chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting, 2015.

[7] Carles Ventura, Miriam Bellver, Andreu Girbau, Amaia Salvador, Ferran Marques, and Xavier Giro i Nieto. Rvos: End-to-end recurrent network for video object segmentation, 2019.

[8] Ning Xu, Linjie Yang, Yuchen Fan, Dingcheng Yue, Yuchen Liang, Jianchao Yang, and Thomas Huang. Youtube-vos: A large-scale video object segmentation benchmark, 2018.