Self-supervised Fine-tuning of Tracktor for Multiple Object Tracking

Carolin Hecking-Veltman
TUM
carolin.hecking@tum.de

Adrian Zieger
TUM
adrian.ziegler@tum.de

Abstract

Tracktor is a tracking-by-detection multiple object tracking framework that constructs tracks using only an object detector. We propose to fine-tune the object detector to individual tracks of a scene in order to prevent identity switches and to re-identify tracks after occlusions.
Fine-tuning the regression head using a self-supervised training set does not improve the performance on average. By fine-tuning the classification head we are able to outperform Tracktor with no re-identification (re-id) method and match the performance of the siamese network used for re-id.

1. Motivation

Tracktor ([1]) is a framework that performs multiple object tracking in videos. The core idea of Tracktor is to use bounding boxes from the previous frame as region proposals for the object detector. By forward-passing these region proposals through the last part of the network, the bounding box from the previous frame is regressed to the person of the current frame in case the class scores are high enough. By that identity preservation is achieved.
In this project we propose to solve two fundamental challenges that Tracktor faces through self-supervised fine-tuning. The first challenge is that Tracktor occasionally regresses a bounding box to the wrong person in the next frame. That disrupts the identity within a track. We propose to fine-tune the regression head of the detector, that regresses the bounding box within Tracktor, to each track. This enables a distinction between people and avoids incorrect regression. Another challenge for Tracktor is the re-id of a person after it has been occluded. Tracktor uses a siamese network to reidentify people. In this project we propose to reuse the classification head of the detector for re-id by fine-tuning it to each track in a self-supervised manner. This is in line with Tracktor's paradigm that tracking is possible by only using an object detector.

2. Related Work

We evaluate our approach on the multiple object tracking benchmark ([3]). This project is based on and extends the Tracktor framework ([1]). As proposed in Tracktor we use a Faster-RCNN ([4]) as the object detector using a feature pyramid network ([2] as a backbone.
In [5] the authors fine-tune a network for each track in order to re-id vehicles, learning similarity with a triplet loss on observations from the preceding and following frames. In [6] one-shot learning is applied to person re-id using distances between group-based feature representations proposing the transfer local relative distance comparison on features obtained from e.g. color histograms.

3. Methods

Our approach consists of two main steps: Firstly, a training set is generated while the track is active. This includes the ROI-pooled individual feature maps, the bounding box and the label. Secondly, the tracks each have their own copy of the final layers of the regression or the classification head of the detector. These are fine-tuned using the training set. The two steps are adjusted based on whether the regression or the classification head is to be fine-tuned.

3.1. Regression Head Fine-Tuning

In the regression case the ground truth used for the fine-tuning of a track is the bounding box that is predicted

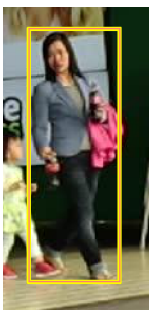


Figure 1. The training boxes shown in yellow are generated by sampling a random displacement in both dimensions. In the picture the training set with a batch size of 1024 is shown and it can be observed that the boxes are uniformly distributed around the orange 'ground truth'.

by the original detector of Tracktor. As training samples we use as input boxes that are randomly shifted around the ground truth corresponding to region proposals as shown in figure 3. The detector then regresses these to a bounding box around the person. The input additionally contains the ROI-pooled features corresponding to the saved box in that frame. This 'ground truth' is available at test time, however it does **not** correspond to the annotations of the MOT data set which are used to evaluate the performance of the tracking algorithm. In the regression case, only the final layer of the regression head is fine-tuned, since it provides sufficient parameters to learn the task.

3.2. Classification Head Fine-Tuning

To fine-tune classification we construct a training set that contains as positive samples the features and boxes of the person whose track's classification head is being fine-tuned with label 1. Additionally, it contains as negative samples the features and boxes of other people in the same frame with label 0.
In the classification case we fine-tune the three final layers of the classification head, while preserving the non-fine-tuned state of the layers that are shared with the regression head for the regression part.
When a track is set to inactive, its classification head is fine-tuned on the at most 40 previous frames. We train the layers with cross entropy loss. Since the number of epochs to reach sufficient class separation varies from track to track, the number of required epochs vary as well. To achieve faster training, we monitor the training process with regards to how well the classes are separated. We stop the training when two thresholds are passed: One for the lowest score of the positive samples and one for the minimum distance between a positive and a negative sample. For a new detection, all classification heads of inactive tracks are evaluated on it. If a classifier scores a detection above *min reid threshold* and scores all other detections in the frame at least *score gap* lower, the track corresponding to the classifier is added as a re-id candidate for this detection. Only if there is a re-id candidate for a detection, re-id is triggered and the track corresponding to the classifier with the highest score is extended.

4. Results

We evaluate our results based on the MOT benchmark using the MOT17 data set with detections given by FRCNN. The code for all experiments is published on Github [1].

	MOTA	IDF1	MT	FP	FN	ID Sw.
Tracktor++	**61.7%**	**65.0%**	194	694	42031	**290**
Ft regressor reinit ts	61.6%	64.6%	**195**	**668**	**42020**	**290**
Ft regressor collect ts 6 it	61.6%	64.6%	**195**	669	**42020**	400

Table 1. This table compares scores of different variations of fine-tuning the detector to the baseline. First row: baseline Tracktor, second row: regressor is fine-tuned every 50 frames using a training set sampled entirely from the current frame, third row: regressor is fine-tuned every 5 frames on a training set that is collected over every two frames

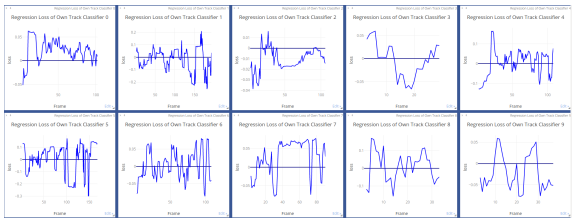
4.1. Regression Head Fine-Tuning

For regression we benchmark our results against Tracktor without fine-tuning, with all models using the siamese re-id suggested in Tracktor. We test a model that fine-tunes every 50 frames using a training set sampled in the current frame as well as a model that collects its training set along the sequence and fine-tunes every 5 frames. Both models use a batch size of 1024, learning rate of 10^{-5}, 6 iterations and the random displacement of the boxes is limited to 0.01 of their shortest edge. The scores of our two proposed fine-tuning approaches are shown in table 4.1.

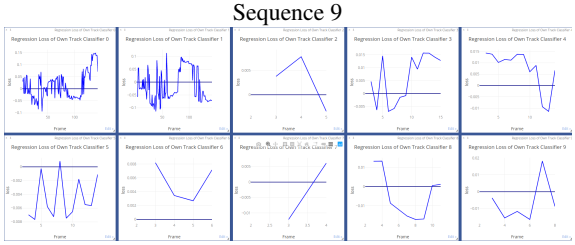
To fine-tune our regression head we use the boxes that are produced by the original object detector. At the same time these are the boxes that Tracktor, our baseline, produces. It is a challenge to outperform Tracktor, because during our fine-tuning process the regression head only knows a 'ground truth' given by Tracktor but not the real ground truth, i.e. the annotations. This is visualized in figure 2: The distance to the annotations of the boxes produced our model fluctuates around the distance of the boxes produced by the original detector. Fine-tuning every 50 frames using a training set entirely sampled in the current frame with the configuration described above does not improve the performance of Tracktor as shown in table 4.1.

We also tried fine-tuning the detector more often with a training set that is collected from different frames. This may allow it to average out noise in the bounding box predictions of the original detector for this person and thus adjust to their appearance. In our experiments this mechanism improved the IDF1 score on some sequences significantly but on average it decreased both the IDF1 and the MOTA score. This is exemplary shown in table 2. Figure 2 and table 2 show how the model performs on different sequences. In sequence 9 tracks exist for a large number of frames. It can be observed that the regression loss of the fine-tuned regressor fluctuates around the baseline. For this sequence, the fine-tuning increased the IDF1 score from

[1] https://github.com/CHeckingV/tracking_wo_bnw



Sequence 9



Sequence 5

Figure 2. This figure demonstrates the generalization of the fine-tuned regression head on detections of the same person for subsequent frames in the sequence. The graphs show the difference in regression loss between the original detector and the regression loss using the fine-tuned regression head. Each individual plot shows one track evaluating the corresponding fine-tuned regression head.

		MOTA	IDF1
Sequence 5	Tracktor	**55.83%**	**63.39%**
	Ft regressor collect ts 6 it	55.40%	60.75%
Sequence 9	Tracktor	**63.13**	53.62%
	Ft regressor collect ts 6 it	63.04	**57.98%**

Table 2. The approach of fine-tuning the regressor often using a training set compiled from the features and bounding boxes is compared to Tracktor for the two sequences 5 and 9.

53% to 58%. In sequence 5 most tracks only exist for a very small number of frames. In this case the model does not benefit from the training set collection which is reflected in a significantly decreased IDF1 score and the decreased MOTA score. Overall it can be stated that fine-tuning the regressor using the available detections as a ground truth for the self-supervised training set does not improve the performance of Tracktor.

4.2. Classification Head Fine-Tuning

We compare our fine-tuned classification head to two models: Firstly, Tracktor not using any re-id and secondly Tracktor using a siamese network for re-id. We use a learning rate of $5 \cdot 10^{-4}$ and stop training when sufficient class separation is achieved as described in 3.2. As *min reid threshold* we set 0.95 and as *score gap* 0.2. The higher the thresholds the less often re-id is triggered. We decided for a conservative approach to avoid false positives. To circumvent the class imbalance in the self-supervised data set we upsample the positive examples. As for the siamese network, we set the *inactive patience* to 10 frames.

	Sequence	MOTA	IDF1
Tracktor without re-id	02	40.7%	43.0%
	04	**64.5** %	**71.3** %
	05	55.4 %	60.7 %
	09	63.0 %	**58.0 %**
	10	70.2 %	64.8 %
	11	**68.0 %**	**63.7** %
	13	71.2 %	71.2 %
Tracktor siamese re-id	02	**40.8%**	**45.9%**
	04	**64.5** %	71.2 %
	05	**55.8 %**	**63.4 %**
	09	**63.1 %**	53.6 %
	10	**70.3** %	65.3 %
	11	**68.0** %	63.3 %
	13	**71.6** %	72.7 %
Tracktor fine-tuned re-id	02	**40.8%**	44.9 %
	04	**64.5** %	71.1%
	05	55.7 %	61.6 %
	09	63.0 %	**58.0 %**
	10	**70.3** %	**65.4 %**
	11	**68.0 %**	**63.7** %
	13	**71.6 %**	**73.9%**

Table 3. We evaluate our re-id against Tracktor without any re-id and Tracktor using the siamese network for re-id. Our system consistently reaches higher IDF1 and MOTA scores than Tracktor without any re-id. Compared to the re-id with siamese our approach reaches on four out of seven sequences a higher IDF1 score. Siamese with reid only reaches a a marginally better MOTA score on some sequences, namely on sequence 05 and 09.

In table 3 we compare the two baseline models with our approach for each sequence individually. We beat Tracktor without re-id for all sequences except for the IDF1 score on sequence 04. Interestingly, Tracktor **without** re-id outperforms using the siamese network with regards to IDF1 score on more sequences compared to our method, namely on sequence 04, 09 and 11. This shows that our method has a better precision than the siamese network. Generally, the siamese network only beats our method with regards to the IDF1 score on three out of seven sequences and with regards to the MOTA score on two out of seven sequences. However, on the other five sequences the MOTA scores a equal. This shows that the siamese network is able to re-id more detections correctly, but at the cost of lower precision.

References

[1] Philipp Bergmann, Tim Meinhardt, and Laura Leal-Taixé. Tracking without bells and whistles. *CoRR*, abs/1903.05625, 2019. 1

[2] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. *CoRR*, abs/1612.03144, 2016. 1

[3] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler. MOT16: A benchmark for multi-object tracking. *arXiv:1603.00831 [cs]*, Mar. 2016. arXiv: 1603.00831. 1

[4] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015. 1

[5] Yuhao Xu and Jiakui Wang. A unified neural network for object detection, multiple object tracking and vehicle re-identification, 2019. 1

[6] W. Zheng, S. Gong, and T. Xiang. Towards open-world person re-identification by one-shot group-based verification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(3):591–606, March 2016. 1