Residual Transfer Learning for Multiple Object Tracking

Anonymous AVSS submission for Double Blind Review

Paper ID 181

Abstract

To address the Multiple Object Tracking (MOT) challenge, we propose to enhance the tracklet appearance features, given by a Convolutional Neural Network (CNN), based on the Residual Transfer Learning (RTL) method. Considering that object classification and tracking are significantly different tasks at high level. And that traditional fine-tuning limits the possible variations in all the layers of the network since it changes the last convolutional layers. Beyond that, our proposed method provides more flexibility in terms of modelling the difference between these two tasks with a four-stage training. This transfer approach increases the feature performance compared to traditional CNN fine-tuning. Experiments on the MOT17 challenge show competitive results with the current state-of-the-art methods.

1. Introduction

Multiple Object Tracking (MOT) is a critical task in the pipeline of a video monitoring system. A variety of approaches have been proposed to tackle MOT challenges such as abrupt object appearance changes, occlusions and illumination variations. However, these issues remained unsolved. Tracking-by-detection has become one of the most popular and effective methods to approach this challenge [4, 15, 18]. After a detector provides bounding boxes for the objects, the problems translate to how to model object appearance adapted to the variety of video scenarios and perform data association where the bounding boxes are paired to a trajectory.

Recent MOT algorithms [6, 11, 16, 21, 22] trend to improve tracking performance by enhancing the appearance feature quality. However, these trackers may have issues to represent an object appearance adapting itself to variation of video scenes, changes in pose, different illumination and occlusions. In particular, the trackers [6, 11] model the object appearance by extracting hand-crafted features, and the trackers do not perform well when complete and long occlusions change the objects appearance significantly.

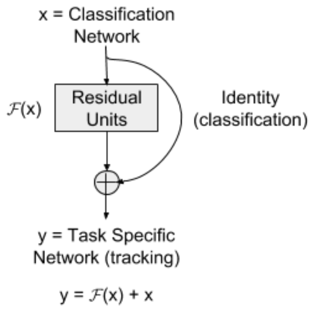
x = Classification Network
$\mathcal{F}$(x)
Residual Units
Identity (classification)
y = Task Specific Network (tracking)
y = $\mathcal{F}$(x) + x

Figure 1. Transfer Learning as a Residual Learning Problem

Lately, thanks to the increasing amount of annotated and available data, multi-object tracking has seen many improvements with deep learning approaches. With the success of deep learning in solving classification problems, more and more trackers such as [16, 21, 22] extract deep appearance features to describe objects and obtain significantly higher performance in both online and offline settings. Authors in [20] propose a novel and efficient way to obtain discriminative appearance-based tracklet affinity models. In this framework, each sample pair is passed to a Siamese CNN including two sub-CNNs to extract the feature vectors. Then, based on the feature vectors obtained from the last layer of both sub-CNNs in each video segment, temporally constrained metrics are learned online to update the appearance-based tracklet affinity model. Finally, the MOT problem is formulated as a Generalized Linear Assignment (GLA) problem which is solved by the soft-assignment algorithm. The approach in [3] performs the MOT task by an enhancing detection model. The object appearance features are extracted from a CNN fully-connected layer. To cope data association ambiguities in crowded scenarios, this approach extended the multiple hypothesis tracking approach by incorporating a novel enhancing detection model. This model analyses the correlation of detection-scene and detection-detection. The scene is modeled by using dense confidential detections and therefore handles false trajectories. The latter estimates the correlations between individual detections and improves the

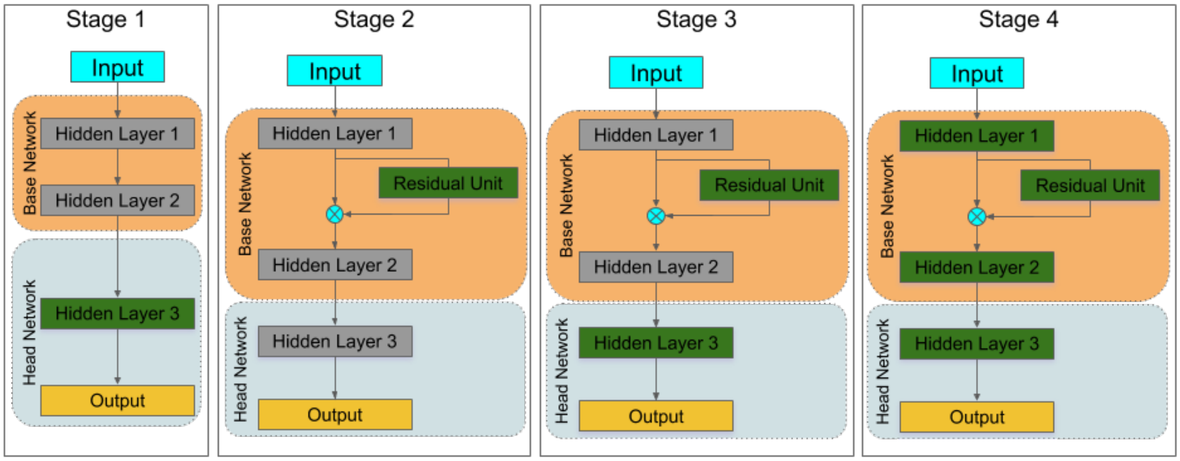

Figure 2. Training stages of Residual Transfer Learning method. Each stage only trains the layers shown in green, and fixes the layers in grey. The residual units are added at the second stage.

ability to deal with close object detections in crowded scenarios.

These methods learn object features by fine-tuning mechanisms which mostly change the last convolutional layers and do not leverage the difference between the classification and tracking tasks.

In this paper, instead of using the traditional fine-tuning way of transfer learning, we present a smart-training approach in four different stages. This alternative strategy is called Residual Transfer Learning (RTL), and it is done to better model the difference between two different tasks. As said before, more robust and discriminative appearance models will lead to higher performance, and the features extracted using RTL improve the appearance models.

The rest of the paper is organized as follows: Section 2 presents the limitations of traditional fine-tuning and the proposed solution with Residual Transfer Learning. In section 3, we detail the modifications made to CNN model, VGG-16 pretrained on ImageNet, and the features extracted to build the appearance models. The proposed RTL feature based multi-object tracker is described in section 4. In section 5, the experiments and evaluations of our method and other state-of-the-art trackers are presented. Finally, section 6 concludes the paper.

2. Transfer Learning

One of the main reasons deep learning is popular nowadays is the practicality of transferring learned models. For example, a model trained for classification can be hastily used for different tasks such as localization, object detection and instance segmentation. Transfer learning reduces the time it would take to train a model from scratch. The traditional way of transfer learning in CNN is to initialize a model, pretrained with a large dataset such as Imagenet, and then replace the last layers to *fine-tune* the new desired classes with specific datasets. This method mostly changes the classification or dense layers but not much the initial convolutional ones, since the classification layers are the ones giving the high abstract meaning to the classes. Traditional fine-tuning is efficient when the tasks are similar, when we want to simply narrow specific classes or target a dataset. The problem is that classification and tracking are different tasks, thus assuming the same layers can give good performance for both, even after fine-tuning, is inherently inaccurate.

2.1. Residual Transfer Learning

Here, we present a smart training alternative for transfer learning based on the concept of ResNet by He *et al.* [8]. In ResNet, a layer learns the estimate residual between the input and output signals. We cast transfer learning as a residual learning problem, since the objective is to close the gap between the initial network and the desired one as shown in Figure 1. Achieving this goal is done by adding residual units for a number of layers to an existing model that needs to be transferred from one task to another. The existing model can thus be able to perform a new task by adding and optimizing residual units. The main advantage of using residual units for transfer learning is the flexibility in terms of modelling the difference between two tasks.

2.2. Training

There is a major problem when adding residual units: their initialization. The ideal scenario is when the residual units approximate the identity function in the beginning. This problem can be overcome by first training the network

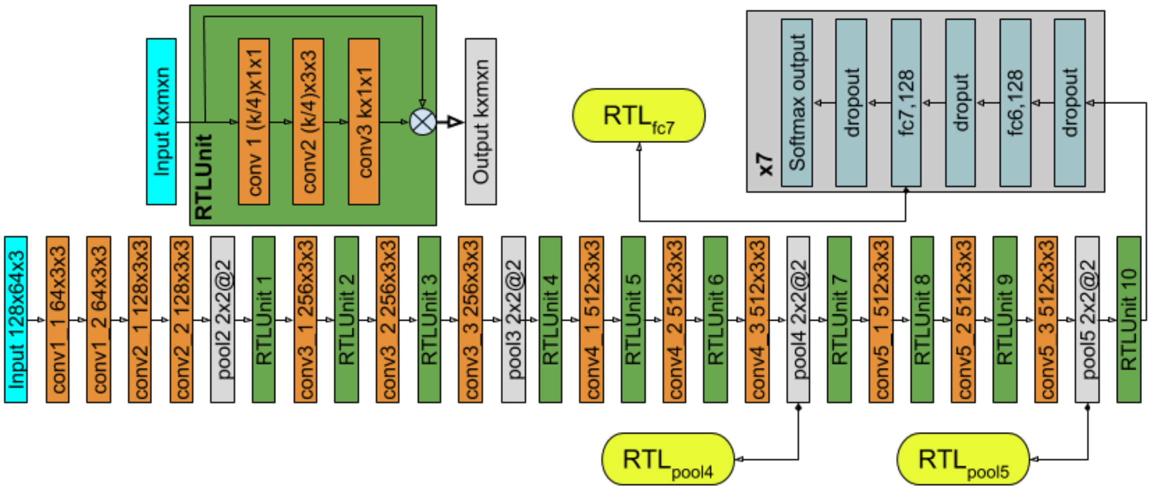

Figure 3. Network architecture for Multiple-Object tracking. A total of 10 RTLunits are added to the 16-layer VGG model. The operations and parameters are shown in each box. Convolutional layers are represented by number of filters x kernel size. Pooling layers use kernel size 2x2 and stride of 2. Each RLTunit contains 3 convolutional layers and a shortcut. The output of RTLUni10 is divided into 7 part models simultaneously without parameter sharing. The outputs of fc7, pool5 and pool4 are used to build the appearance models RTL$_{fc7}$, RTL$_{pool5}$ and RTL$_{pool4}$ respectively.

with a new *head* - the layers towards the end of the network, such as classification - and then training the residual units, keeping the head of the network unchanged. The advantage of this approach training stage is significantly less training loss, by having one source of error instead of two per stage.

We propose 4 training stages that ensure a better feature adaptation of the network as shown in Figure 2.

Stage 1: The first stage is intended to learn the high level representations of the new specific task at hand. We replace the head of the original network and initialize it randomly, training only the new head and keeping the network's base unchanged. The residual units are not added at this stage.

Stage 2: In the second stage, we add the residual units between the convolutional layers and initialize them randomly. In order to approximate to the identity function, we fix the base and head of the network and train only the residual units. The residual units are the only source of loss during this stage.

Stage 3: In the third stage, we train the head and the residual units conjointly. This allows us to learn both high and low representations.

Stage 4: The last stage is optional, as we noticed that the loss function is low enough after the third stage. Nonetheless, further improvement performance can be achieved by training the whole network.

3. Residual Transfer Learning for Multiple-Object Tracking

Using RTL, we transfer the VGG-16 layer model, pretrained on ImageNet for object classification, into a tracking feature extractor. We overlooked ResNet for its extreme depth because our target dataset is small and do not warrant such a deep model for higher performance. Two sets of modifications are made. The first is to change the input size of the network and the second is to train 7 different stripes of a person. The final network architecture is shown in Figure 3.

3.1. Base model modifications

VGG16 is a pretrained model for 1000-way object classification with an input of 224x224 pixels. It consists of 5 convolutional groups (*conv1-conv5*) and two fully connected layers (*fc6-fc7*) followed by a classification layer. Each convolutional group is passed trough a max-pool layer (*pool1-pool5*). We replace the fully connected layers *fc6* and *fc7* as well as the classification layer. This allows us to change the input size to 128x64, a more suitable size for people tracking. With a smaller receptive field, we remove the first max-pool layer (*pool1*), producing 512 filter maps of 7x3 at the output of max-pool layer *pool5*.

3.2. Head model modifications

At test time, appearance models are built with the features extracted by the *embedding* layer. We build three embedding layers from *pool4*, *pool5* and *fc7*. To accomplish this, we split the output of *pool5* in 7 parts, one for each horizontal stripe of the feature map. And we use two fully connected layers of size 128 for each part feature map (*partx/fc6,partx/fc7*). Dropout layers are interleaved with fully connected layers followed by a n-way classification layer. We train all the part models simultaneously by summing the classification loss for each part. This helps learning a shared low-level representation via the convolutional layer groups, and a specific representation via the fully connected layers.

3.3. Residual Units and their placement

The residual units are added across the layers for two reasons. First, to distribute the burden of error compensation on each specific layer. And second, to reduce the number of parameters per residual unit, decreasing the computational and convergence time. We use a residual unit similar to [8]. Each residual unit is composed of three convolutional layers. The number of filters of the first layer is a quarter (1/4) of the input layer, the kernel size is 1x1 and a stride of 1. The second layer uses the same number of filters as the fist layer, the kernel size is 3x3 and stride of 1. The third layer uses a kernel size of 1x1 and has the same number of kernels as the input of the residual unit. The output of the third layer is then added to the input of the residual unit. Finally, RELU is applied to the combined output.

4. RLT feature based tracker

We define tracklet Tr_i between consecutive frames $< m, n >$ as a chain of tracked objects called nodes N_i^t, $m < t < n$, where i represents the ID of object and N represents the object bounding-box at time t.

$$Tr_i = \{N_i^m, N_i^{m+1}, .., N_i^{n+1}, , N_i^n\} \qquad (1)$$

The appearance of a tracklet is modeled based on the RTL features of all nodes N_i belonging to the tracklet from sub-sample *pool4*.

4.1. Tracklet Representation

The representation of tracklet Tr_i using the learned RTL features is defined as follow:

$$\nabla_{Tr_i} = mean(RTL_i^t) \quad t \in < min(m, n - \Delta), n > \qquad (2)$$

where RTL_i^t is the RTL feature extracted from node N_i^t. Assuming that the appearance of recent nodes is more informative than those in the past, we limit to model the tracklet representation by nodes N_i^t appearing in current time-window Δ: $min(m, n - \Delta) < t < n$. In the experiment, Δ is set to 50 frames.

4.2. Data association

The framework tries to calculate the similarity scores of tracklets in every time-window Δt. The similarity matrix S=$\{m_{ij}\}$ is constructed where i=1..Nb, j=1..Nb, and Nb is the number of tracklets in the current time-window [$t - 2\Delta$t,t]. If tracklet j is a candidate of tracklet i, the similarity of the pair is calculated using Euclidean distance $m_{ij} = Eud(\nabla_{Tr_i}, \nabla_{Tr_j})$, otherwise it is set to zero in the similarity matrix. Once the cost matrix is computed, the optimal association pairs, which minimize the data association cost in S, are determined using Hungarian algorithm.

5. Experiments

In this section, we present the experimental results and analysis the performance of the proposed online MOT framework on the challenging dataset MOT17 [14]. Our approach is compared with hand-crafted feature as well as traditional fine-tune deep learning feature based trackers. In order to have a fair comparison of trackers, we use a publicly available detection and evaluation method on website [1].

5.1. Dataset

The benchmark dataset MOT17 consists of 7 sequences for training and 7 sequences for testing. Each sequence is provided with 3 sets of detections: DPM, Faster-RCNN, and SDP. The dataset contains a large diversity of outdoor scenarios, fixed or moving cameras, crowded scenarios, person-to-person occlusions and low illumination.

5.2. Implementation details

Using the ground-truth provided by the MOT17 challenge, we extract all the non-occluded people (*accuracy=1*), giving us a total of 519 tracks across all training sets, and $\sim 65K$ training images. Therefore, the n-way classification layer we use is 519. We train the network using stochastic gradient descent with momentum of 0.9 and batch size of *MK* where *M* = 64 persons and *K* = 2 images per person. We shuffle the person order after each training person was selected once. The learning rates for stages 1, 2, 3 and 4 are correspondingly (0.1, 0.1, 0.01, 0.0001). The number of iterations for each stage are respectively (10*N*,10*N*,20*N*,20*N*) with *N* = 500. N is the number of batches needed to cover all 65*K* images with the given batch size. We remark that all images may not be used for training, since we randomly use sample data for each person. In other words, we train for as many images in the training set and not all the images during an epoch.

[1] https://motchallenge.net/data/MOT17/

Trackers	Method	MT↑	ML↓	MOTA↑	MOTP↑	FP↓	FN↓	IDSw↓	Frag↓
pool4 + *Euclidean*	online	34.13	21.60	51.8	80.9	6,795	23,577	855	1,066
$\text{RTL}_{\text{Pool4}}$ + *Euclidean*	online	35.42	21.60	51.5	80.8	6,986	23,517	910	1,078
$\text{RTL}_{\text{Pool5}}$ + *Euclidean*	online	35.64	21.81	51.4	80.8	7,158	23,388	904	1,064
RTL_{fc7} + *Euclidean*	online	34.99	21.17	51.7	80.7	7,131	23,232	901	1070

Table 1. Comparisons of baseline model (pool4 without RTL) and RTL models with Euclidean distance. The experiment is evaluated on training sequences of MOT17. The best values are marked in red.

Methods	Trackers	MT↑	ML↓	MOTA↑	MOTP↑	FP↓	FN↓	IDSw↓	Frag↓
Offline	EDMT17 [3]	21.6	36.3	50.0 ± 13.9	77.3	32,279	247,297	2,264	3.260
	FWT [9]	21.4	35.2	51.3 ± 13.1	77.0	24,101	247,921	2,648	4,279
	JCC [10]	20.9	37.0	51.2 ± 14.5	75.9	25,937	247,822	1,802	2,984
	MHT_DAM [12]	20.8	36.9	50.7 ± 13.7	77.5	22,875	252,889	2,314	2,865
	IOU17 [2]	15.7	40.5	45.5 ±13.6	76.9	19,993	281,643	5,988	7,404
Online	Ours	17.6	36.77	45.6 ± 13.3	75.9	34,084	268,860	4,276	6,583
	PHD_DCM [7]	16.9	37.2	46.5 ±13.8	77.2	23,859	272,430	5,649	9,298
	EAMTT [17]	12.7	42.7	42.6 ±13.3	76.0	30,711	288,474	4,488	5,720
	GMPHD_KCF [13]	8.8	43.3	39.6 ± 13.6	74.5	50,903	284,228	5,811	7,414
	GM_PHD [5]	4.1	57.3	36.4 ± 14.1	76.2	23.723	330,767	4,607	11,317

Table 2. Quantitative analysis of our MOT framework on MOT17 challenging dataset with state-of-the-art methods. The tracking results of these methods are public on MOTchallenge website. The best values in both online and offline methods are marked in red.

5.3. Evaluation metrics

In order to evaluate the performance of our multi-object tracking method, we adopt the widely used CLEAR MOT metrics [1]: including multiple object tracking accuracy (MOTA↑) and the multiple object tracking precision (MOTP↑) which punish on false positives (FP↓), false negatives (FN↓) and identity switching (IDSw↓). The tracking-time metrics are also computed: the number of trajectories in ground-truth (GT), the ratio of mostly tracked trajectories (MT↑), a ground truth trajectory that is covered by a tracking hypothesis for at least 80% is regarded as mostly tracked), the ratio of mostly lost (ML↓), a ground truth trajectory that is covered by a tracking hypothesis for at most 20% is regarded as mostly lost) and the number of times a trajectory is fragmented (FM↓). Where ↑ indicates that higher scores lead to better results, and ↓ shows that lower scores correspond to better results.

5.4. Comparison of features

We present the quantitative comparison of the features performance from *pool4*, RTL_{pool4}, RTL_{pool5} and RTL_{fc7} in Table 1. From this table, we can see that $\text{RTL}_{\text{pool4}}$ and $\text{RTL}_{\text{pool5}}$ perform better than pool4, 1.29% and 1.49% increase respectively. We can attribute this improvement to RTL since pool4 is the baseline model and uses the traditional method of fine-tuning. We use RTL_{pool4} as the features for tracklet representation. The reason is that the higher order of layers we use, the network is prone to overfitting since we have a small amount of data to train with.

5.5. Comparisons with the state-of-the-art methods

The table 2 show the quantitative comparison between our approach and nine state-of-the-art trackers on benchmark dataset MOT17. These compared methods are categorized into offline and online tracking. The tracking performances are computed over 7 testing sequences with 3 different detections: DPM, F-RCNN and SDP. The values shown in table 2 are computed by the mean of each metric (ML, ML, MOTA, MOTP) and by the sum of each indicator (FP, NP, IDSw, Frags). As the discussion in [19], metrics MT and ML are proved to be closer to user expectations than the others. Therefore, the performance of all compared trackers are sorted in descent order by the MT metric. Generally, offline trackers with their beforehand information of objects and scenarios have better performance than online trackers. Comparing two trackers including $EDMT17$ - the best offline tracker and our approach - the best online tracker, we can see the modest increases of around 4.5% of metrics MT and MOTA, 1.4% of metric MOTP and a slight decrease of 0.4% of metric ML.

Our approach is also compared with four other online tracking methods. The results show that our approach has the best performances on the metrics (MT, ML, FN and IDSw) and the second best performances on the metrics (MOTA, Frags). On MOTP metric, there is a slight decrease of 1.3% (from 77.2 % to 75.9%) when comparing our approach and the best performance belonging to tracker PHD_DCM.

It is shown in the table 2 that the performance of state-of-the-art trackers are modest on this challenging benchmark dataset. The best results (only 21.6% and 50% measured by ML and MOTA metrics, respectively) belong to tracker $EDMT17$ which works in the offline mode.

6. Conclusions

In this paper, we have presented a method to render transfer learning more efficient. This four-stage training method alleviates the difference between object classification and tracking and improves tracklet representation. In addition, the ability of our approach to generalize well from small amount of data is crucial for practical applications because recurrent data collection in large amount to train models is inconvenient.

References

[1] K. Bernardin and R. Stiefelhagen. Evaluating multiple object tracking performance: The clear mot metrics. *EURASIP Journal on Image and Video Processing*, 2008(1):246309, May 2008.

[2] E. Bochinski, V. Eiselein, and T. Sikora. High-speed tracking-by-detection without using image information. In *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6, Aug 2017.

[3] J. Chen, H. Sheng, Y. Zhang, and Z. Xiong. Enhancing detection model for multiple hypothesis tracking. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2143–2152, July 2017.

[4] W. Choi. Near-online multi-target tracking with aggregated local flow descriptor. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, pages 3029–3037, Washington, DC, USA, 2015. IEEE Computer Society.

[5] V. Eiselein, D. Arp, M. Ptzold, and T. Sikora. Real-time multi-human tracking using a probability hypothesis density filter and multiple detectors. In *2012 IEEE Ninth International Conference on Advanced Video and Signal-Based Surveillance*, pages 325–330, Sept 2012.

[6] L. Fagot-Bouquet, R. Audigier, Y. Dhome, and F. Lerasle. Improving multi-frame data association with sparse representations for robust near-online multi-object tracking. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VIII*, pages 774–790, 2016.

[7] Z. Fu, F. Angelini, S. Naqvi, and J. Chambers. Gm-phd filter based online multiple human tracking using deep discriminative correlation matching. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[9] R. Henschel, L. Leal-Taix, D. Cremers, and B. Rosenhahn. Improvements to frank-wolfe optimization for multi-detector multi-object tracking. *arXiv:1705.08314*, 2017.

[10] M. Keuper, S. Tang, Z. Yu, B. Andres, T. Brox, and B. Schiele. A multi-cut formulation for joint segmentation and tracking of multiple objects. In *arXiv:1607.06317*, 2016.

[11] H. Kieritz, S. Becker, W. Hbner, and M. Arens. Online multi-person tracking using integral channel features. In *2016 13th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 122–130, Aug 2016.

[12] C. Kim, F. Li, A. Ciptadi, and J. M. Rehg. Multiple hypothesis tracking revisited. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, pages 4696–4704, Washington, DC, USA, 2015. IEEE Computer Society.

[13] T. Kutschbach, E. Bochinski, V. Eiselein, and T. Sikora. Sequential sensor fusion combining probability hypothesis density and kernelized correlation filters for multi-object tracking in video data. In *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–5, Aug 2017.

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

[15] A. Milan, S. Roth, and K. Schindler. Continuous energy minimization for multitarget tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(1):58–72, Jan 2014.

[16] A. Sadeghian, A. Alahi, and S. Savarese. Tracking the untrackable: Learning to track multiple cues with long-term dependencies. *CoRR*, abs/1701.01909, 2017.

[17] R. Sanchez-Matilla, F. Poiesi, and A. Cavallaro. *Online Multi-target Tracking with Strong and Weak Detections*, pages 84–99. Springer International Publishing, Cham, 2016.

[18] S. Schulter, P. Vernaza, W. Choi, and M. K. Chandraker. Deep network flow for multi-object tracking. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2730–2739, 2017.

[19] F. Solera, S. Calderara, and R. Cucchiara. Towards the evaluation of reproducible robustness in tracking-by-detection. In *2015 12th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6, Aug 2015.

[20] B. Wang, L. Wang, B. Shuai, Z. Zuo, T. Liu, K. Luk Chan, and G. Wang. Joint learning of convolutional neural networks and temporally constrained metrics for tracklet association. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2016.

[21] L. Wang, N. T. Pham, T. T. Ng, G. Wang, K. L. Chan, and K. Leman. Learning deep features for multiple object tracking by using a multi-task learning strategy. In *2014 IEEE International Conference on Image Processing (ICIP)*, pages 838–842, Oct 2014.

[22] F. Yu, W. Li, Q. Li, Y. Liu, X. Shi, and J. Yan. *POI: Multiple Object Tracking with High Performance Detection and Appearance Feature*, pages 36–42. Springer International Publishing, Cham, 2016.