Fast Simultaneous People Detection and Re-identification in a Single Shot Network

Wiebe Van Ranst and Toon Goedemé

EAVISE, Technology Campus De Nayer, KU Leuven, Belgium
{firstname.lastname}@kuleuven.be

Abstract

A traditional re-identification pipeline consists of a detection and re-identification step, i.e. a person detector is run on an input image to get a cutout which is then sent to a separate re-identification system. In this work we combine detection and re-identification into one single pass neural network. We propose an architecture that can do re-identification simultaneously with detection and classification. The effect of our modification has only a negligible impact on detection accuracy, and adds the calculation of re-identification vectors at virtually no cost.

The resulting re-identification vector is strong enough to be used in speed sensitive applications which can benefit from an additional re-identification vector in addition to detection. We demonstrate this by using it as detection and re-identification input for a real-time person tracker. Moreover, unlike traditional detection + re-id pipelines our single-pass network's computational cost is not dependent on the number of people in the image.

1. Introduction

Re-identification (re-id), the task of matching person identities between different appearances of the same person is a well studied problem. Current state of the art techniques use deep convolutional neural networks (CNNs) trained using a loss function like triplet loss [11, 26], contrastive loss [8] and on-line instance matching (OIM) loss [30]. The principle generally remains the same: the neural network transforms an input image into an (embedding) vector which contains distinctive features of the subject in the image (people in our case). This vector is then embedded in a space where vectors of same identities (eg. coming from pictures of the same person) are close to each other, and vectors of different identities far from each other. A distance measure (often cosine or Euclidean distance) in this vector space can then be used to measure how similar two objects are to each other.

Figure 1. We create a neural network that can do both detection and re-identification at the same time. Left shows the Euclidean distance of a tracked person compared to all other output cells, right shows detection output and center uses both detection and re-identification to create a tracker.

Re-id can be useful in many applications: In the field of surveillance, it can be used to search for a person in a huge quantity of video sequences coming from different cameras. Other applications include recognizing people in photos which is useful for all kinds of purposes (eg. social media, photo albums). Apart from that, re-id can also be used to make person tracking more robust. Matching identities in consecutive frames in a video sequence (with an optional interval) can reaffirm tracks created by a tracker, it makes it possible to overcome total occlusions and allows the tracker to be used in applications where non-overlapping multi camera handover is required. By treating the tracking problem as a clustering problem it can also be used as a tracker by itself.

The mapping from image to id-vector is generally done from a cropped image containing one person to the embedding space. A two stage pipeline is used: first a person detector is used to locate all people in the image, after which cutouts of the image containing each a single person are sent to a re-id network which transforms them into an embedding space. To detect people in a video frame an object detection algorithm is used which can be based on hand

crafted features (like ACF [5] or DPM [7]) or use a CNN pipeline (like YOLO [23], SSD [18], FPN [17] or Faster R-CNN [25]).

Indeed, to get state-of-the-art results a computationally intensive two-step CNN pipeline is mainly used. For off-line processing pipelines that require highly accurate results this is often acceptable. However, for systems that need to work in real-time on the other hand (e.g. a tracker) this often rules out re-id altogether.

This raises the question: Is it not possible to combine this pipeline into one fast neural network that can fulfill both objectives at the same time? I.e. we envision a neural network that at the same time outputs detections and also mappings of these detections into an embedding space. So far, combining these steps has only been attempted in a primitive way by Xiao *et al.* [29] with the more computationally intensive Faster R-CNN [25].

In this paper we investigate how this can be done using a single one pass fully convolutional network. In this paper we introduce YOLO-REID, a single shot convolutional neural network based on YOLOv2 that is able to do both person detection and re-identification at the same time introducing only a negligible increase in computational cost and small penalty in detection accuracy. We make the choice of the YOLOv2 architecture because of its compactness (only 23 convolution layers), which gives it the ability to meet our real-time constraints. We of course do not expect the same accuracy as other re-id approaches like [12] which use the deeper Inception or ResNet networks. If better accuracy is desired, the proposed approach can however similarly be used in deeper networks.

To the best of our knowledge we are the first to attempt a combination of a single-shot-detector and re-identification. This approach would have unique properties: it would be able to do detection and re-id at the same time without much extra cost. By running a detector you essentially get re-id for free. It would be possible to output an embedding vector of every person in the image without adding any extra cost independent of the number of people in frame. We demonstrate our approach by applying it to a tracker. The output is visually shown in figure 1.

The remainder of the paper is structured as follows: Section 2 will explain some of the related work. Section 3 will go into further detail on the architecture of our network and how we train it. In section 4 we demonstrate our work by using it as the basis of a tracker, section 5 compares our approach to both itself and others in terms of tracking, detection and re-identification accuracy. Our conclusions can be found in section 6.

2. Related Work

One of the motivations of this work is to use re-identification to create a reliable tracker. We will first briefly discuss the related work about tracking. Next, we will explain related work about re-identification and detection.

Multi-object tracking can be divided into two approaches. On the one hand there is tracking by detection which only looks at the output of an object detector, and tries to associate detection boxes to create accurate tracks [2, 21]. The recent SORT [2] uses a simple Kalman tracker combined with the Hungarian algorithm and an association metric based on detection overlap to create an accurate multi-person tracker. Other approaches [3] use only a simple IoU measure that is able to achieve state-of-the-art results thanks to the recent advances in object detection accuracy. Off-line approaches have also been proposed [1, 13, 14, 20, 22]. Because they have all detection data available they can solve the track assignment problem as a global optimization problem.

Approaches that make use of appearance features [1, 28, 31] generally perform better than tracking-by-detection on its own at the cost of more computational complexity.

The rise of deep learning in the field of re-id lies at the foundation of this paper [6, 9, 10, 11, 15, 26, 27, 32, 33, 34]. From these works triplet loss [11, 26] in combination with hard mining [11, 16] still seems to be a top performer in terms of learning strategy.

The rise of deep learning has also had a big impact on object detectors. Hand crafted approaches like ACF [5] and DPM [7] have for a while now been overshadowed by deep learning pipelines. Region based approaches like R-FCN [4] and Faster R-CNN [25] in combination with Feature Pyramid Networks [17] currently get the best results in terms of accuracy, while Single-Shot networks like SSD [18] and YOLO [23, 24] offer an optimal trade-off in terms of accuracy and speed.

Recent work [29, 30] has also tried to combine detection and tracking into one network for the task of identifying a single person or multiple people in a larger video set. They modify the Faster R-CNN [25] object detector. The region proposal part of Faster R-CNN is kept as is, and they concatenate the region proposal net of Faster R-CNN with a re-id network, creating in essence a detector that can be used directly for re-identification. Compared to our approach by using Faster-R-CNN they still use a two stage pipeline, albeit a pipeline that was previously used for detection and classification. The disadvantage of using Faster R-CNN however is that it is not as fast as a single shot network. Computational cost also increases when more people are in the scene.

3. YOLO-REID

The idea of our person detection and re-identification network *YOLO-REID* is to create one network, based on YOLOv2 [23], that is able to accurately detect people in an image (same accuracy as standard YOLOv2) and calcu-

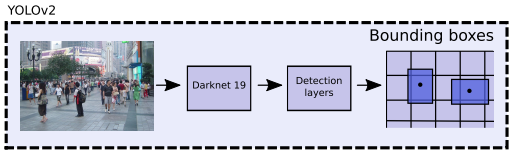

Figure 2. Simplified YOLOv2 network architecture

late re-identification vectors (embeddings) on these people at the same time by evaluating the network only once.

3.1. Network architecture

Figure 2 gives a (simplified) overview of the YOLOv2 network architecture. An input image is first passed to the Darknet19 fully convolutional network, which outputs a set of feature maps which are used as input to an again fully convolutional network that outputs class probabilities and detections in the final layer. The output layer of YOLOv2 is structured as a grid of 13x13 cells (in the case of the default input size of 416x416) each containing k anchor points representing different bounding box sizes. Every anchor point contains a bounding box description (x and y-offset, a scale in x and y direction w.r.t. the anchor and the probability that this bounding box contains an object) and also a classification output. In our modified architecture we propose to also output an embedding for each cell in the output grid. For more information about the YOLOv2 architecture we refer to [23].

One of the goals in our proposed architecture is to be able to share Darknet19 weights for detection and re-identification so we only need to pass through the network once. To do this we tried two architectures: "split end" (figure 3) and "mixed end" (figure 4). During training we update all weights starting from the pre-trained Darknet19 weights trained on ImageNet. In both approaches the classification outputs from the original YOLO are removed, and instead the network outputs a 128-value embedding at each cell in the output grid.

"Split end" uses two separate layers (having the same size as the YOLO detection layers) that are kept separate from the detection pipeline, with the idea that the network can use these to learn a better embedding without impacting detection. "Mixed end" does away with this separate pipeline and fits everything in the YOLO architecture, creating a network that is smaller than the original YOLO architecture with 80 classes (see section 5.3). Training for both architectures remains the same. Our total loss function is the following:

$$L = L_{\text{region}} + \alpha L_{\text{reid}}, \tag{1}$$

where α is the weight of the re-id loss, L_{region} is the standard YOLOv2 region loss with the softmax classification loss removed, and L_{reid} is the re-identification (triplet) loss (see section 3.2).

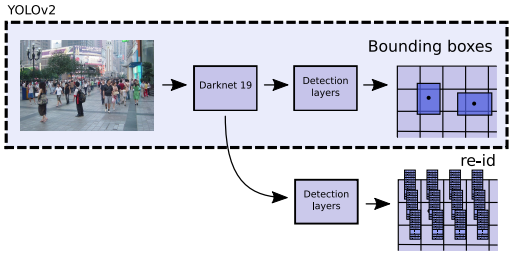

Figure 3. Overview of the "split end"-architecture: detection and re-identification are preformed separately.

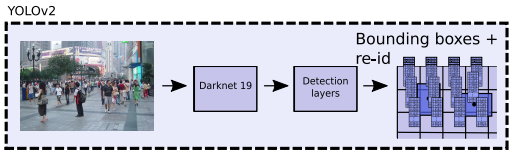

Figure 4. Overview of the "mixed end"-architecture: detection and re-identification layers are merged.

The scale factor α is determined empirically in part, but also varies depending on the input resolution of the images used to calculate region loss during training.

3.2. Triplet loss

We use triplet loss [11, 26] with batch hard mining as described by Hermans et al. [11], implemented in the open-reid framework[1]:

$$L_{\text{tri}}(\theta) = \sum_{\substack{a,p,n \\ y_a=y_p\neq y_n}} [D_{a,n} - D_{a,p} + \mathrm{m}]_+ \tag{2}$$

Where $D_{x,y}$ is the Euclidean distance between two embeddings, a is an anchor embedding, p is a positive embedding (same person as a), and n is a negative embedding (different person than a). This loss function essentially tries to maximize the distance between a and n, and tries to minimize the distance between a and p within some margin m. If $D_{p,a} < m$ the triplet is not moved closer to the anchor.

As proposed by Hermans et al. the loss function using batch hard mining is given as follows:

$$L_{\text{Re-ID}}(\theta, X) = \overbrace{\sum_{i=1}^{P}\sum_{a=1}^{K}}^{\text{all anchors}} [m + \overbrace{\max_{p=1\ldots K} D(f_\theta(x_a^i), f_\theta(x_p^i))}^{\text{hardest positive}} - \underbrace{\min_{\substack{j=1\ldots P \\ n=1\ldots K \\ j\neq k}} D(f_\theta(x_a^i), f_\theta(x_n^j))}_{\text{hardest negative}}]_+, \tag{3}$$

where f_θ is the neural net parametrized by θ that maps from image space to embedding space. For every batch of size P ($= 6$) the hardest positive sample (the one furthest away

[1] https://cysu.github.io/open-reid/

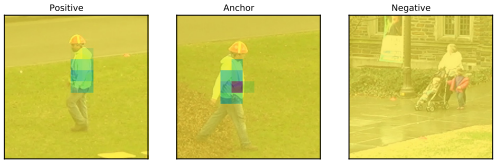

Figure 5. Example of the output of the network. The distance between anchor point and embedding is visualized, brighter means a large distance, darker means a small distance.

from the anchor) and the hardest negative sample (the one closest to the anchor) is chosen to calculate loss. For every identity two instances are sampled from the dataset.

3.3. Multi target training

During training we calculate both losses separately. Region loss is trained in the same manner as standard YOLOv2 complete with data augmentation. Our implementation is based on two free software PyTorch implementations: Lightnet[2] and pytorch-yolo2[3].

We use the CUHK-SYS dataset [29] as it seems to lend itself perfectly for this purpose. Instead of cropped patches it contains annotated persons (both id and bounding box) in uncropped images in a variety of scenes.

As explained in section 3.2 we use batch hard mining to train re-id. For every mini-batch a random set of ids and corresponding images is sampled from which the hardest positive and hardest negative is selected to calculate triplet loss. To know which images are the hardest, we make a forward pass of all images through the network. Since images within this mini-batch are selected randomly, we also use them to calculate region loss. For every image (not just the one selected to calculate triplet loss) we have bounding box annotations that we use to calculate region-loss in addition to triplet-loss.

We use triplet loss as explained in section 3.2 with that difference that we only learn the embedding-vector at the grid cell in the center of the target person, the cell where an anchor point would also generate a detection. Figure 5 illustrates the desired outcome, cells at the center of the positive patch (left) have a low Euclidean distance at the anchor point in the center of the person (darker) and other samples (right and elsewhere in the images) have a high distance (lighter).

4. Person Tracking

We demonstrate the usefulness of our approach by making a person tracker. Indeed, as the re-id vector computed by our 23-layer network will not be equally powerful as those

[2] https://gitlab.com/EAVISE/lightnet

[3] https://github.com/marvis/pytorch-yolo2

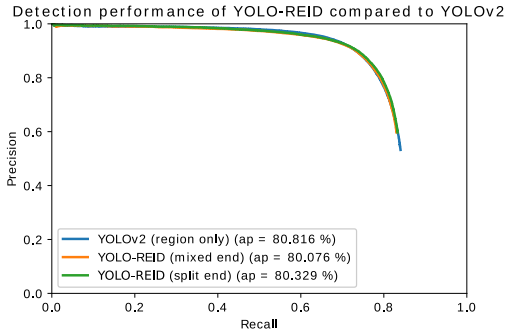

Figure 6. Comparison of our YOLO-REID network to standard YOLOv2.

computed by a full ResNet, we will test its usefulness in the reasonably less demanding application of tracking people within one camera view. We will now briefly discuss the functionality of this tracker. We base our tracker on the SORT tracking-by-detection tracker [2], which uses a simple Kalman tracker combined with the Hungarian algorithm and a simple IoU metric for track assignment. Similarly to [28] we extend this tracker using our learned re-id vector as an association metric. The tracker uses both detections and appearance embedding vectors coming from our network. In contrast to [28], in our approach the addition of an appearance feature comes at a negligible cost in terms of computational complexity. The computational complexity is also not influenced by the amount of people in the image. Appearance features are calculated by the network for every cell. A full evaluation of the tracking algorithm can be found in section 5.4.

5. Results

In this section we evaluate the performance of our network. We will first evaluate the detection performance, compared to our standard YOLOv2, trained and evaluated on the CUHK-SYS train/test-set. Next, we evaluate the network using the criteria of person search. We calculate average precision and top-1 accuracy on the test set of CUHK-SYS yielding a combined figure for detection and re-id. We also give an overview of the computational complexity of the different architectures. We evaluate the tracker using the MOT16 tracking challenge metrics.

5.1. Detection

Figure 6 compares the standard YOLO people detector to our YOLO-REID network with embedding vectors built in. Both detectors where trained on the same CUHK-SYS dataset [29], “Mixed end” uses shared weights up to the last output layer of the network (figure 4), “split end” uses a different approach where the two last output layers are kept separate (figure 3). We can see that detection accuracy does

	mAP (%)	Top 1 (%)
YOLO-REID (mixed end)	20.38	33.37
YOLO-REID (split end)	21.58	35.67
YOLOv2 + histogram	4.44	9.66
Xiao et al. [29]	55.7	62.7

Table 1. The re-id performance of our two architectures compared to colour histogram matching, which has a similar computational cost.

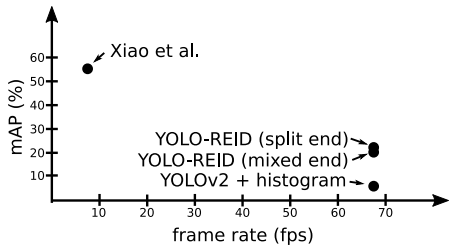

Figure 7. Comparison in speed and accuracy.

not seem to suffer much by replacing the multi class classification with a re-id vector. Compared to "mixed end", "split end" seems to perform better by a small margin. This can be expected considering the extra layers in "split end".

5.2. Person search

Person search is the combination of detection and re-id merged into one task. Given some query images together with the annotated query person, it is the task of the algorithm to find this person in the gallery. It has to detect the person and also correctly identify them as the query person.

Table 1 shows the "person search" accuracy in terms of both mean average precision (mAP) and top-1 score. We evaluate on the CUHK-SYS test set with a gallery size of 50. As a baseline we also compare to colour histogram matching (based on an implementation of a particle tracker[4]) with YOLOv2 region only detections, which is often used in tracking as another low cost method. It is clear that our method performs a lot better. We can also see that adding extra layers in "split end" clearly has increased accuracy over "mixed end". Xiao et al. [29] still seems to be more accurate than our method. However, if we also compare them in terms of speed (figure 7) our method makes a great trade-off in terms of speed and accuracy. This makes it usable in resource constraint applications like embedded platforms and large scale video processing.

5.3. Computational complexity

We mentioned frequently that our network does not come with much extra cost compared to standard YOLO. Table 2 shows the computational complexity in terms of network parameters and floating point operations (FLOPs) per image of both architectures compared to standard YOLOv2.

[4] https://bitbucket.org/kschluff/particle_tracker

	Parameters	MFLOPs
YOLO-REID (mixed end)	50 704 761	29 370.03
YOLO-REID (split end)	62 505 337	33 357.24
YOLOv2 (80 class)	50 983 561	29 464.17

Table 2. Computational cost of different networks.

The complexity of "mixed end" is lower than the standard YOLOv2 because classification output is removed. Adding extra layers comes at the cost of 12% extra computation and 19% extra parameters.

5.4. Person tracking

We evaluate our tracker using re-id appearance features against a standard one class YOLO tracker trained ourselves without the use of appearance features (tracking by detection, which only uses a Kalman tracker combined with Hungarian algorithm and IoU) using the MOT16 evaluation framework. The result is shown in table 3. Both trackers are run at the same working point on the PR curve. Overall we get great improvements in tracking accuracy using our method. We get significantly more mostly tracked (MT) and partly tracked (PT) trajectories suggesting that our re-id vector is successful in reconstructing tracks. Overall MOTA and IDF scores are also increased. More information about these measurements can be found in [19].

6. Conclusion

We presented a single shot neural network architecture based on YOLOv2 that is able to perform both detection and re-identification, while only introducing a negligible increase in complexity compared to the normal YOLOv2. Our experiments indicate that our architecture is able to yield a usable re-id vector, with minimal loss in accuracy ($\pm 0.5\%$ average precision), compared to other low cost approaches. We evaluated two architectures and found that adding extra layers in the more complex "split end" architecture benefits detection and re-id accuracy only a small amount.

We demonstrated the effectiveness of our approach in the application of tracking. Using our approach tracking accuracy increased greatly compared to only using detection for tracking.

Acknowledgements

This work is supported by the agency Flanders Innovation & Entrepreneurship (VLAIO), Research Foundation - Flanders (FWO) and the company Robovision.

References

[1] S.-H. Bae and K.-J. Yoon. Robust online multi-object tracking based on tracklet confidence and online discriminative appearance learning. In *CVPR*, pages 1218–1225, 2014. 2

	IDF1	IDP	IDR	Rcll	Prcn	FAR	GT	MT	PT	ML	FP	FN	IDs	FM	MOTA	MOTP	MOTAL
YOLOv2 Tracking-by-Detection	27.0	**59.1**	17.5	27.8	**93.6**	**0.40**	517	13	187	317	**2102**	79759	774	**1951**	25.2	**77.5**	25.9
YOLO + histogram	31.8	57.3	22.0	34.2	89.1	0.87	517	23	220	274	4630	72646	1135	2336	29.0	75.8	30.0
YOLO-REID (split end)	**33.9**	56.7	24.2	**36.1**	84.5	1.37	517	**26**	227	264	7306	**70580**	882	2327	28.7	75.5	29.5
YOLO-REID (mixed end)	33.2	**59.1**	23.1	35.2	89.8	0.83	517	**26**	212	279	4393	71569	**723**	2001	**30.5**	76.5	**31.2**

Table 3. Comparison of tracking performance against our own trained one class YOLO without re-id and with re-id enabled.

[2] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft. Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3464–3468, 2016. 2, 4

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

[4] J. Dai, Y. Li, K. He, and J. Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems*, pages 379–387, 2016. 2

[5] P. Dollár, R. Appel, S. Belongie, and P. Perona. Fast feature pyramids for object detection. *IEEE TPAMI*, 36(8):1532–1545, 2014. 2

[6] A. Dosovitskiy, J. T. Springenberg, M. Riedmiller, and T. Brox. Discriminative unsupervised feature learning with convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 766–774, 2014. 2

[7] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *CVPR*, pages 1–8. IEEE, 2008. 2

[8] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *CVPR*, volume 2, pages 1735–1742. IEEE, 2006. 1

[9] X. Han, T. Leung, Y. Jia, R. Sukthankar, and A. C. Berg. Matchnet: Unifying feature and metric learning for patch-based matching. In *CVPR*, pages 3279–3286, 2015. 2

[10] B. Harwood, G. Carneiro, I. Reid, T. Drummond, et al. Smart mining for deep metric learning. *arXiv preprint arXiv:1704.01285*, 2017. 2

[11] A. Hermans, L. Beyer, and B. Leibe. In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737*, 2017. 1, 2, 3

[12] A. Hermans, L. Beyer, and B. Leibe. In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737*, 2017. 2

[13] C. Huang, B. Wu, and R. Nevatia. Robust object tracking by hierarchical association of detection responses. In *ECCV*, pages 788–801. Springer, 2008. 2

[14] H. Izadinia, I. Saleemi, W. Li, and M. Shah. 2t: Multiple people multiple parts tracker. In *ECCV*, pages 100–114. Springer, 2012. 2

[15] B. Kumar, G. Carneiro, I. Reid, et al. Learning local image descriptors with deep siamese and triplet convolutional networks by minimising global loss functions. In *CVPR*, pages 5385–5394, 2016. 2

[16] V. B. Kumar, B. Harwood, G. Carneiro, I. Reid, and T. Drummond. Smart mining for deep metric learning. *arXiv preprint arXiv:1704.01285*, 2017. 2

[17] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *CVPR*, volume 1, page 4, 2017. 2

[18] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *ECCV*, pages 21–37. Springer, 2016. 2

[19] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler. Mot16: A benchmark for multi-object tracking. *arXiv preprint arXiv:1603.00831*, 2016. 5

[20] 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, 2014. 2

[21] S. Oh, S. Russell, and S. Sastry. Markov chain monte carlo data association for multi-target tracking. *IEEE Transactions on Automatic Control*, 54(3):481–497, 2009. 2

[22] H. Pirsiavash, D. Ramanan, and C. C. Fowlkes. Globally-optimal greedy algorithms for tracking a variable number of objects. In *CVPR*, pages 1201–1208. IEEE, 2011. 2

[23] J. Redmon and A. Farhadi. Yolo9000: better, faster, stronger. *arXiv preprint arXiv:1612.08242*, 2016. 2, 3

[24] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. 2

[25] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 2

[26] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, pages 815–823, 2015. 1, 2, 3

[27] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer. Discriminative learning of deep convolutional feature point descriptors. In *ICCV*, pages 118–126. IEEE, 2015. 2

[28] N. Wojke, A. Bewley, and D. Paulus. Simple online and realtime tracking with a deep association metric. In *IEEE International Conference on Image Processing (ICIP)*, pages 3645–3649, 2017. 2, 4

[29] T. Xiao, S. Li, B. Wang, L. Lin, and X. Wang. End-to-end deep learning for person search. *arXiv preprint*. 2, 4, 5

[30] T. Xiao, S. Li, B. Wang, L. Lin, and X. Wang. Joint detection and identification feature learning for person search. In *CVPR 2017*, pages 3376–3385. IEEE, 2017. 1, 2

[31] F. Yu, W. Li, Q. Li, Y. Liu, X. Shi, and J. Yan. Poi: multiple object tracking with high performance detection and appearance feature. In *ECCV*, pages 36–42. Springer, 2016. 2

[32] Y. Yuan, K. Yang, and C. Zhang. Hard-aware deeply cascaded embedding. *CoRR, abs/1611.05720*, 1, 2016. 2

[33] S. Zagoruyko and N. Komodakis. Learning to compare image patches via convolutional neural networks. In *CVPR*, pages 4353–4361. IEEE, 2015. 2

[34] X. Zhang, H. Luo, X. Fan, W. Xiang, Y. Sun, Q. Xiao, W. Jiang, C. Zhang, and J. Sun. Alignedreid: Surpassing human-level performance in person re-identification. *arXiv preprint arXiv:1711.08184*, 2017. 2