# Improving Deep SORT Tracking Embedding

**Erick Platero (1812570), Keyon Amirpanahi (1958721)*, Nihil Patel (1901253)***

*Department of Computer Science, University of Houston*

**These authors contributed equally to this work and are ordered based on the first letter of their first names.*

**Abstract:** Multi-object Tracking (MOT) has undergone many advances due to its applications in security, passenger counting, and more. In this work, we seek to evaluate the impact of the optimization function in training Deep SORT's re-identification network. More specifically, we aim to evaluate the differences in training Deep SORT's re-identification network using triplet loss with cosine and euclidean distance, weighted triplet loss of both cosine and euclidean distance, and the quadruplet loss. The results of this study indicate that all training configurations reach close to 100% mean accuracy on the testing set.

## 1. Introduction

The task of Multi-object Tracking (MOT) has undergone many advances. SORT formulated MOT as a data association problem where we can associate the detections of a detection model with tracks [1]. The association was made by calculating the Intersection over Union (IoU) of detections and the predicted location of tracks using the kalman filter. Then, Deep SORT introduced a re-identification embedding to recover tracks that were temporarily occluded [2]. Since then, multiple formulations, each with their own unique scheme, have been formed by leveraging IoU overlap and the re-identification embedding to make associations [2–6]. While most tracking methods focus on the tracking-by-detection paradigm, recent efforts have focused in creating a tracking module that performs detection and tracking in a single step [3, 7]. The benefits of such tracking is that they are able to perform tracking in real-time at the expense of lower accuracy.

In this work, we will be focusing on evaluating what the impact of training a re-identification network with different optimization schemes are. More specifically, we will train a re-identification network with triplet loss using cosine/euclidean distance, a weighted combination of triplet loss with both cosine and euclidean distance, and the quadruplet loss with its learned distance metric. The triplet loss function uses a reference image (anchor), a positive image of the same identity as the anchor and a negative image of a different identity [8]. Optimization of this loss requires minimizing the distance between the anchor and positive images, and maximizing the distance between the anchor and negative images [9]. We chose triplet loss because this is a standard loss function used in many tracking works and is also a method that has been thoroughly studied [9–11]. Further, the reason we will implement a weighted combination of triplet loss with cosine and euclidean distance is because we hypothesize that both distances optimize different aspects of the vector (more on this explain on Supplemental section of this paper). Quadruplet loss functions introduce a new negative image with yet another different identity. Adding a new term referencing a negative-negative image pair and comparing it to the anchor-positive pair allows the loss function to distinguish whether a pair contains the same identity or not [12]. By introducing this new constraint, the loss function is cited to generalize better with testing data. This is important as there are works that suggest that triplet loss generalizes poorly from training to testing datasets [12]. As such, this work also seeks to provide evidence for or against the speculation that triplet loss generalizes poorly on the testing dataset.

We have decided to base our experiments on Deep SORT's re-identification model because it is a popular off-the-shelf tracking framework that has been reproduced and adapted to many projects. In general, Deep SORT uses a pre-trained model to generate detections (e.g., YOLOv3) and then leverages a cascade matching scheme that first associates tracks and detections with a re-identification network and then associates with an IOU overlap association. The re-identification model takes as input the image patches that were localized on the detection bounding boxes and outputs an embedding that attempts to extract the signal features of the person identified in the bounding box. As such, this re-identification is tasked with compressing the image from a high-dimensional space to a low one that captures the semantics of the interested object. In general, we want these embeddings to be as close to each other for people that belong to the same identity and be as far away as possible for people with different identities.

In short then, the contributions of this work is that:

- we seek to evaluate the performance of different optimization methods on Deep SORT's re-identification model

- we seek to find any differences in generalization performance between triplet and quadruplet loss
- we seek to improve Deep SORT by finding the optimal training optimization loss

## 2. Methods

### *2.1. Mining Triplets and Quadruplets*

To quickly mine triplets, we first select a random anchor image from the dataset. Then, we randomly select an image that contains the same identity as our anchor. Finally, one image of a different identity from the anchor and positive is randomly chosen to serve as the negative image. For quadruplet mining, we follow the same paradigm with an extra image sample (referenced as negative2) whose identities differs from the anchor, positive, and negative samples.

### *2.2. Training, Validating, and Testing the Model*

We train the model on the Market-1501 dataset's training partition [13]. This dataset is used to train the model because it provides photos of human subjects from different angles and under different environmental conditions. This is necessary to train a robust re-identification embedding model that is invariant to factors such as lighting, angle, and the subject's pose. The original Deep SORT model was also trained using this dataset, so training our model on it will allow us to better compare the two models. The Market-1501 testing partition is used for validation, and the model is tested using the MOT-17 dataset's training partition [14].

### *2.3. Metric*

The ultimate goal of the person re-identification embedding is to be able to correctly associate embeddings of the same identity. Since most tracking models are evaluated under the MOT benchmark, which are made up of crowded videos on a moving camera. As such, to assess performance of a tracking pipeline, training and validation of these models are done on video sequences. However, in this work, we are only interested in assessing the performance of the re-identification network alone, not the entire Deep SORT tracking pipeline. As such, we formulate a re-identification accuracy metric that assesses whether our model is able to correctly associate an anchor with its positive samples and not the negative samples. We do this by calculating the nearest-neighbor of both the anchor and positive samples. To do this, we need to compute the fully-connected distances between each pair. Given that we have three nodes, this creates a triangle. Knowing this allows us to understand that when computing nearest-neighbor of the anchor and positive node, there are ten total scenarios that may arise. In general, there are three-types of triangles that will be formed: (i) equilateral triangle, (ii) isosceles triangle, and (iii) scalene triangle. Because of the permutations that we can form with our anchor, positive, negative nodes, each type of triangle may result in multiple scenarios. More specifically, an equilateral triangle scenario creates one case since it is permutation-invariant, isosceles triangle creates three cases, and each permutation on a scalene triangle creates a unique case. Thus, there are ten scenarios that we may have when computing the nearest-neighbor between the anchor and positive node. Iterating these scenarios is important as it gives us insight to what the relationship between the nodes are in a dimensional space that we cannot visualize. Although counting the specific scenarios was not implemented in this scenario, we leave this as further work for future analysis. eq. (1) shows the implementation of our re-identification accuracy for triplet-loss where $(a, p, n)$ represent the anchor, positive and negative embedding respectively. eq. (2) is the nearest-neighbor w.r.t. the first argument, eq. (3) shows the distance function we use for the triplet accuracy, and eq. (4) is the defined Kronecker delta.

$$reidAcc(a,p,n,nn) = \frac{\delta_{nn(a,(p,n))} + \delta_{nn(p,(a,n))}}{2} \tag{1}$$

where

$$nn(x,y_i) = argmin((dist(x,y_i))) \tag{2}$$

$$dist(x,y) = \|x-y\|_2 \tag{3}$$

$$\delta_i = \begin{cases} 1 & \text{if } i = 0 \\ 0 & \text{if } i \neq 0 \end{cases} \tag{4}$$

For the quadruplet loss, the re-identification accuracy is shown in eq. (5) where the new term $n2$ refers to the second negative embedding.

$$reidAcc(a,p,n,n2,nn) = \frac{\delta_{nn(a,(p,n,n2))} + \delta_{nn(p,(a,n,n2))}}{2} \tag{5}$$

However, the distance function *dist* for the quadruplet loss is a learned metric. For the training and validation set, accuracy is averaged per batch and per epoch and for the testing set, accuracy is averaged per single epoch.

## 3. Ablation Study

In this section, we describe our experiments and results. We train each model on 50 epochs each with a batch size of 128. For each batch, random triplets and quadruplets are selected as described in section 2.1. To keep the randomness fixed across each training configurations, we set a fixed seed to generate the same random triplets per training of each model. For pre-processing, we resize the image to (128,64) resolution as the original Deep SORT implementation does. Further, we perform three random operations to our cropped images: (i) color jitter (ii) random horizontal flip and (iii) random rotation. Lastly, we blur the outer parts of each image in each batch to force the model to focus on the center content of the image (thereby acting as a kind of attention mechanism). Finally, we train and validate our model on the Market-1501 dataset and test on MOT17's training dataset as described in section 2.2.

For this ablation study, we trained the same Deep SORT model among eight different optimization functions: triplet loss using cosine (triplet_cos) and euclidean distance (triplet_eucl), a weighted combination of both losses (triplet_comb_x_y where x and y represent numbers representing the corresponding weighted value), and the quadruplet loss. The order of the combined triplet loss is weighting the value of triplet using cosine distance first and triplet using euclidean distance second. The mean accuracy of each configuration on the training, validation, and testing set is shown in 1 with the configuration with best performance shown in bold. For each testing configuration, the last saved checkpoint of the model during training and evaluation is used.

From 1, we can immediately tell that all configurations reached a mean testing accuracy higher than 98%. This is surprising as we reasoned that mean testing accuracies would be significantly lower than mean validation accuracy since the Market-1501 dataset and the MOT17's training partition are different in many aspects. However, this is not the case. In fact, with the exception of the quadruplet loss, the mean accuracies across training, validation, and testing for each corresponding configuration are almost equivalent. To understand why the quadruplet loss is the exception, **??** shows us the mean accuracies per every 100th batch for all configurations. From this we can generally tell that the quadruplet loss needed to be trained on about 3000 batches before converging to an accuracy of about 100%. This is expected as the quadruplet loss is tasked with learning a re-identification representation and a distance representation simultaneously. What is not expected is how fast the triplet variants converged to an accuracy close to 100%. In fact, a similar case also arises for the mean batch accuracies of the validation set as shown in fig. 2. This may lead us to think that the model will be overfitting to our data and thus have less of a performance on the testing set, however, this was not the case. Further, If we look at the training and validation losses as shown in figs. 3 and 4, we see that by the time we start recording loss at the first 100th batch, the model has already converged as when the batches continue, all losses remain mostly stagnant. What's surprising about this is that while all models quickly converged, only the quadruplet loss took about 3000 batches to properly converge to a high accuracy in both training and validation despite the loss not changing much per every 100th batch.

## 4. Conclusion

Our results in the ablation study show us that all loss functions are able to generalize exceptionally well w.r.t. the re-identification accuracy metric. While this may suggest that all training configurations are able to generalize well to testing datasets, these results are premature as there are multiple improvements that can be made to test

Table 1. Training, Validation, and Testing Mean Accuracies.

| **Loss** | **Weight** | **Mean Train Acc.** | **Mean Val. Acc.** | **Mean Test Acc.** |
|---|---|---|---|---|
| triplet_cos | 1 | 99.98% | 99.98% | 99.95% |
| triplet_eucl | 1 | 99.75% | 99.74% | 98.07% |
| triplet_comb_1_1 | [1,1] | 99.88% | 99.84% | 99.86% |
| triplet_comb_.8_.2 | [.8,.2] | 99.84% | 99.86% | 99.86% |
| triplet_comb_.6_.4 | [.6,.4] | 99.92% | 99.91% | 99.89% |
| triplet_comb_.4_.6 | [.4,.6] | 99.88% | 99.86% | 99.84% |
| triplet_comb_.2_.8 | [.2,.8] | 99.85% | 99.86% | 99.81% |
| **quadruplet_learnedmetric** | **1** | **67.62%** | **66.92%** | **100%** |

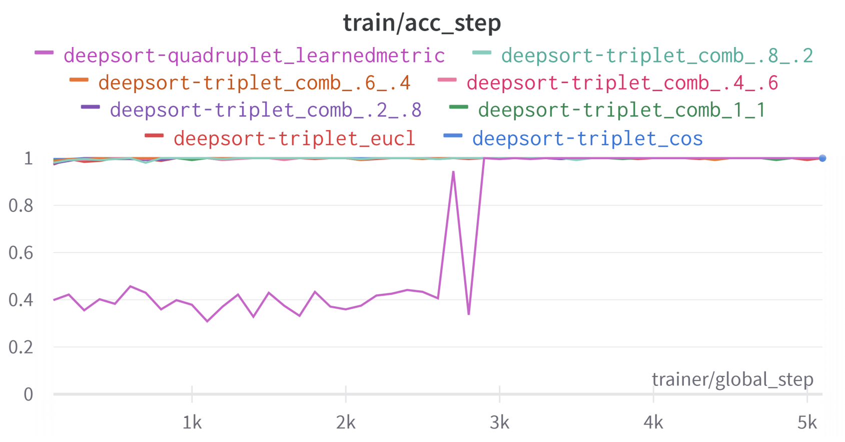

Fig. 1. Mean training accuracy per 100th batch.

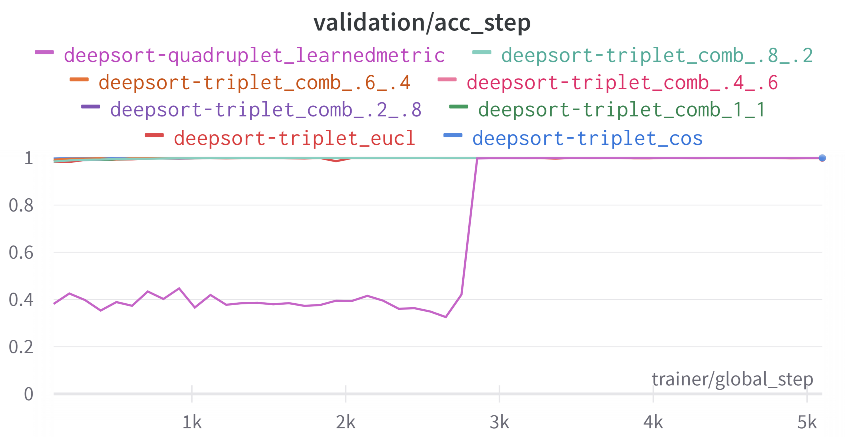

Fig. 2. Mean validation accuracy per 100th batch.

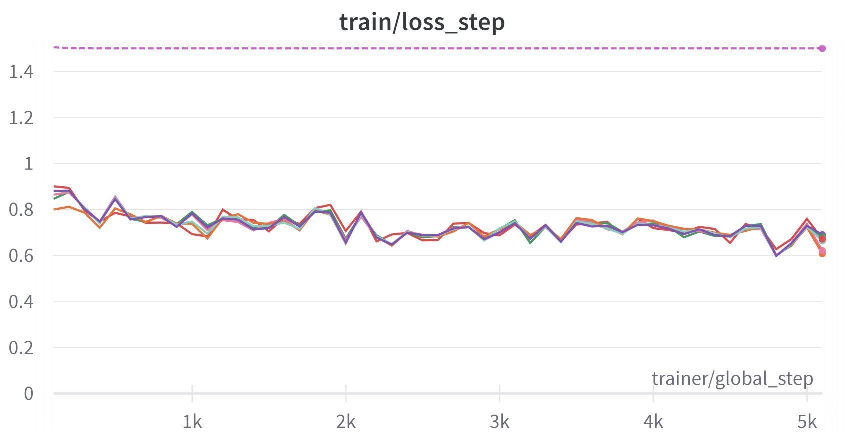

Fig. 3. Mean training loss per 100th batch (quadruplet shown as dashed line).

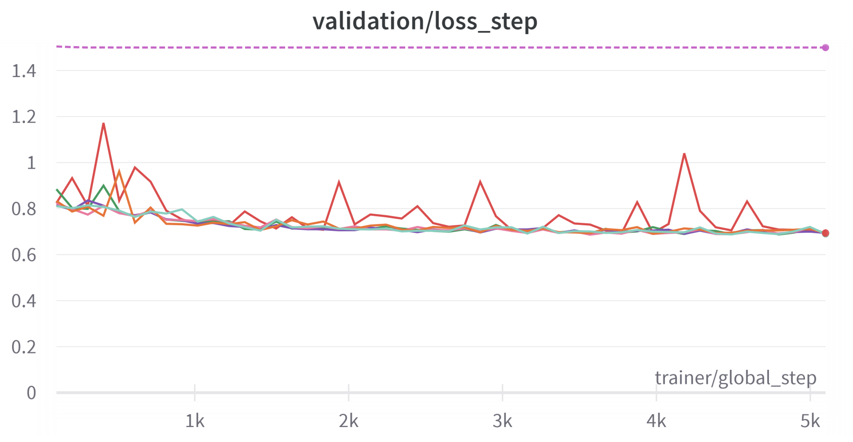

Fig. 4. Mean validation loss per 100th batch (quadruplet shown as dashed line).

performance more rigorously as described in the Future Improvements section. Regardless, with the data we have now, the only difference we are able to see between training methods is that quadruplet loss takes much more time to converge and it also may perform the best out of all configurations since it achieved a testing accuracy of 100%. Further, [12] gives evidence that triplet losses generalize poorly on testing sets but this study so far gives counter-evidence to this argument since all models reached testing accuracies close to 100%. Finally, we reach the conclusion that training Deep SORT's re-identification network using quadruplet loss may be the best alternative despite needing more time to converge.

## 5. Supplemental

In this section, we delve a closer look at the performance of triplet using cosine and euclidean distance. figs. 5 and 6 shows the mean accuracies per 100th batch of triplet using cosine and euclidean for the training and validation sets. These visualizations make it clear that the triplet loss that is the most stable would is triplet cosine, while also converging much faster than its euclidean version. To some degree, this makes sense as the range of the cosine distance is [0,1] while the range of euclidean is from $[0, \text{inf}]$. Because the cosine is more restricted, the model may be better able to find a faster optimal configuration. Because of this restriction, and because cosine distance is invariant to the magnitude of a vector, we reasoned that performing a wighted triplet loss with cosine and euclidean would give us a better performance since we infer that both of these losses optimize different aspects of a vector (cosine optimizes the angles between vectors while euclidean optimizes the angles and magnitudal difference). However, as shown in fig. 7 and **??**, we see that the weighted combination of triplets using cosine and euclidean reduces the amount of training variance in the euclidean while it inherits a faster convergence rate. Given that all training configurations performed similarly on the testing set, we are unable to draw any conclusions as to whether training with a weighted combination of triplet using cosine and euclidean has any added improvements as to just training with either distance alone.

## 6. Future Improvements

This section describes future improvements that could be implemented to make more rigorous comparisons among different training configurations.

First, we created the re-identification accuracy metric to quantify triplet loss performance without applying the tracking module directly to the MOT17 benchmark. This means that we only evaluated triplet accuracy, but ignore the overall tracking performance of Deep SORT on the MOT17 benchmark. We reason that if we are able to improve the performance of the re-identification model over the original implementation of Deep SORT, then we should also see overall tracking improvements on the MOT17 testing dataset. Further, if we wanted to possibly improve testing accuracy, we could introduce more randomness to the pre-processing operations such as mosaic mixture.

Second, the re-identification accuracy we formulated is a simple heuristic to associate a vector based on its nearest-neighbor. However, we reason that it we can also associate on a distance based on a maximum distance threshold. This would require that if the distance between two vectors is below $\tau$, they are assumed to be the same identity. Else, different. The highlight of this approach is that if we use a gallery of identities, we could cluster

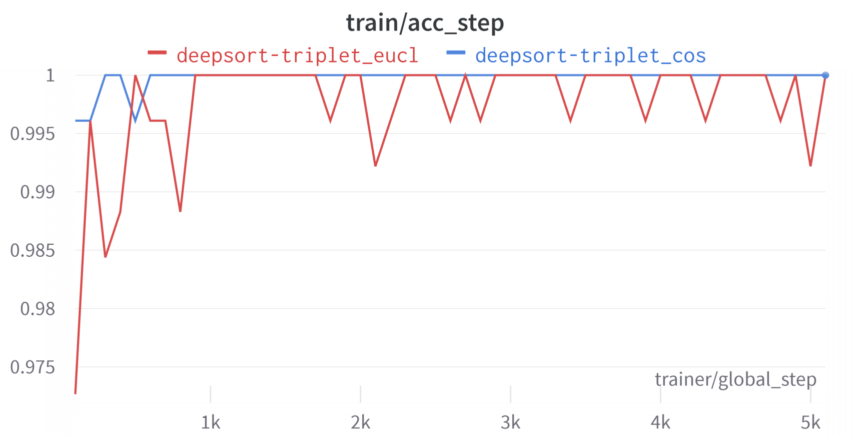

Fig. 5. Mean training accuracy per 100th batch of triplet cosine and euclidean.

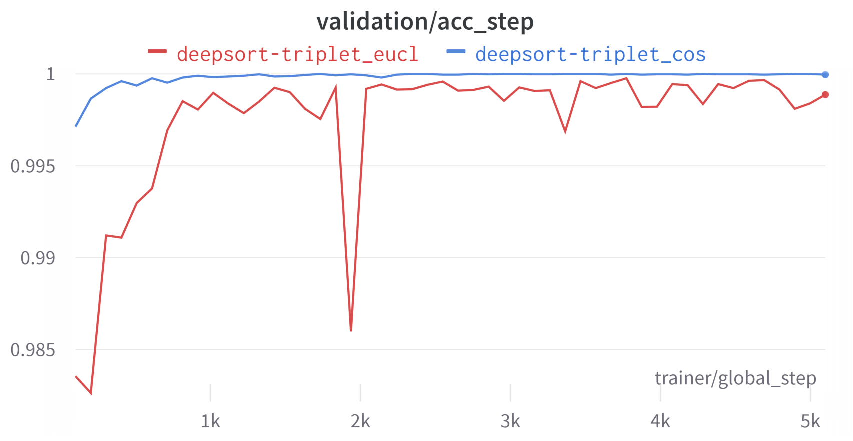

Fig. 6. Mean validation accuracy per 100th batch of triplet cosine and euclidean.

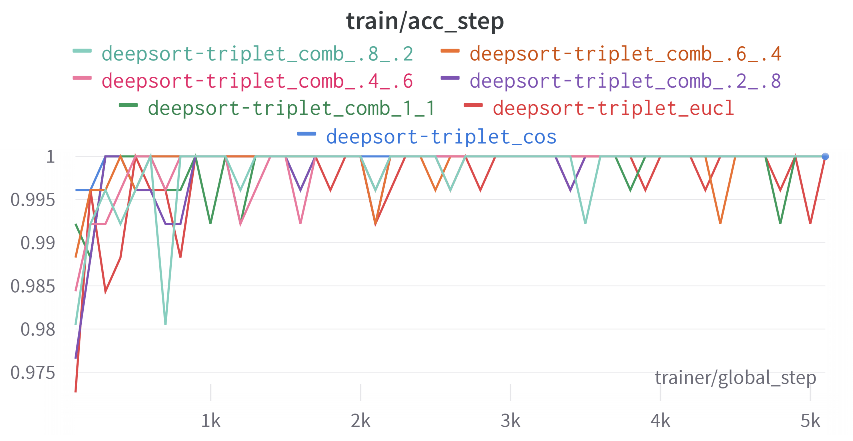

Fig. 7. Mean train accuracy per 100th batch of weighted triplets.

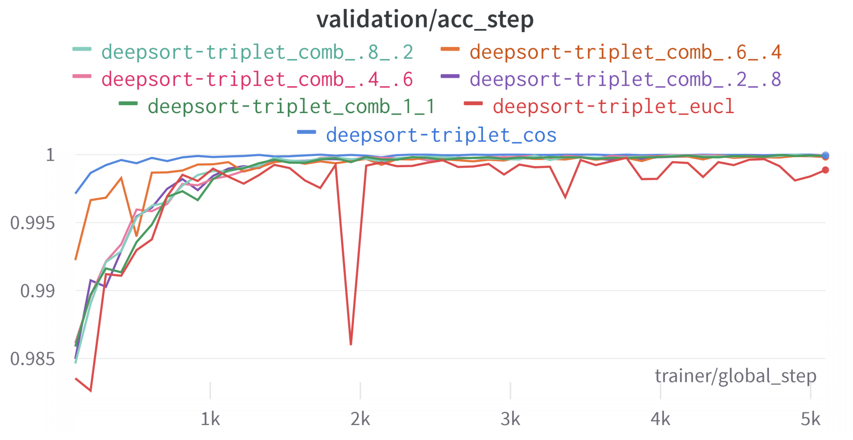

Fig. 8. Mean validation accuracy per 100th batch of weighted triplets.

identities based on a k-means algorithm where we can apply the $\tau$ threshold to classify a new detection.

Third, the re-identification accuracy calculates the nearest-neighbor w.r.t. the positive sample. This means that we expect our model to create embeddings such that the nearest-neighbor of the anchor is positive and vice-versa. We notice that as it stands, the triplet loss does not directly optimize the relationship between the positive and negative sample to be further apart. As such, we could add this constraint to the triplet loss to possibly improve its performance.

Finally, improvements could also be made to our triplet and quadruplet mining methods. As simple as randomly selecting triplets and quadruplets is, this method leads to a mixture of triplets that are both easy and hard to classify on, which can skew the model's training process. When training FaceNet, Schroff et. al manually selected triplets using identities that would be very difficult to classify [9]. We chose to not use this "batch hard" strategy in the interest of time, but it would be useful to implement in the future.

## References

1. 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),* (2016), pp. 3464–3468.
2. N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *2017 IEEE International Conference on Image Processing (ICIP),* (IEEE, 2017), pp. 3645–3649.
3. Y. Zhang, C. Wang, X. Wang, W. Zeng, and W. Liu, "Fairmot: On the fairness of detection and re-identification in multiple object tracking," Int. J. Comput. Vis. **129**, 3069–3087 (2021).
4. Z. Wang, L. Zheng, Y. Liu, and S. Wang, "Towards real-time multi-object tracking," The Eur. Conf. on Comput. Vis. (ECCV) (2020).
5. C. Long, A. Haizhou, Z. Zijie, and S. Chong, "Real-time multiple people tracking with deeply learned candidate selection and person re-identification," in *ICME,* (2018).
6. Y. Zhang, P. Sun, Y. Jiang, D. Yu, Z. Yuan, P. Luo, W. Liu, and X. Wang, "Bytetrack: Multi-object tracking by associating every detection box," arXiv preprint arXiv:2110.06864 (2021).
7. M. Chaabane, P. Zhang, R. Beveridge, and S. O'Hara, "Deft: Detection embeddings for tracking," arXiv preprint arXiv:2102.02267 (2021).
8. L. Zheng, Z. Bie, Y. Sun, J. Wang, C. Su, S. Wang, and Q. Tian, "Mars: A video benchmark for large-scale person re-identification," in *Computer Vision – ECCV 2016,* B. Leibe, J. Matas, N. Sebe, and M. Welling, eds. (Springer International Publishing, Cham, 2016), pp. 868–884.
9. F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," CoRR **abs/1503.03832** (2015).
10. D. P. Vassileios Balntas, Edgar Riba and K. Mikolajczyk, "Learning local feature descriptors with triplets and shallow convolutional neural networks," in *Proceedings of the British Machine Vision Conference (BMVC),* E. R. H. Richard C. Wilson and W. A. P. Smith, eds. (BMVA Press, 2016), pp. 119.1–119.11.
11. H. Xuan, A. Stylianou, and R. Pless, "Improved embeddings with easy positive triplet mining," in *The IEEE Winter Conference on Applications of Computer Vision (WACV),* (2020).
12. W. Chen, X. Chen, J. Zhang, and K. Huang, "Beyond triplet loss: a deep quadruplet network for person re-identification," CoRR **abs/1704.01719** (2017).

13. L. Zheng, L. Shen, L. Tian, S. Wang, J. Wang, and Q. Tian, “Scalable person re-identification: A benchmark,” in *Computer Vision, IEEE International Conference on,* (2015).
14. A. Milan, L. Leal-Taixe, I. Reid, S. Roth, and K. Schindler, “Mot16: A benchmark for multi-object tracking,” (2016).