Balancing Multiple Object Tracking Objectives based on Learned Weighting Factors

Xuan-Thuy Vo, Tien-Dat Tran, Duy-Linh Nguyen and Kang-Hyun Jo
Department of Electrical, Electronic and Computer Engineering,
University of Ulsan
Ulsan (44610), South Korea
Email: {xthuy, tdat}@islab.ulsan.ac.kr; ndlinh301@mail.ulsan.ac.kr; acejo@ulsan.ac.kr

***Abstract*—Advanced multiple object tracking requires multi-task learning in order to solve object detection and data association tasks simultaneously. One fundamental characteristic of multi-task learning is that there is correlated information among tasks, and leveraging this property in training the model can result in better generalization performance. However, in multiple object tracking, most existing methods learn such property by treating multiple task losses equally and independently. In this paper, we takes the weighting of multiple object tracking losses into consideration, relying on the related information among object detection and data association tasks. Firstly, this paper introduces a simple but effective Learned Weighting Factors (LWF) method where the weighting factors are learned through shallow neural networks. These learned factors are used to balance multi-task losses during training dynamically. Thus, our LWF method avoids manually tuning these weighting factors because this process is difficult and expensive caused by the high dimension of the search space. To the best of our knowledge, the proposed LWF method is a new and different perspective in the literature. Secondly, we conduct extensive experiments on two benchmark datasets, i.e., MOT16 and MOT20, surpassing state-of-the-art trackers without extra training samples. Video surveillance demos are available at https://bit.ly/3hkgBxo.**

***Index Terms*—Multiple object tracking, object detection, data association, multi-task learning**

I. Introduction

Multi-task learning is a learning paradigm [1], which learns the related information across multiple tasks to improve the generalization performance of all tasks. In the deep learning generation, multi-task learning encodes the task relatedness information in two components: (i) multi-task architectures with shared network parameters train multiple tasks simultaneously, (ii) task weighting is to balance the joint learning of multiple tasks to prevent an objective imbalance that one or more tasks can overwhelm training. Being multi-task learning issue, multiple object tracking (MOT) can be potentially improved from multi-task learning perspectives. Based on such ability, this paper leverages the benefits of multi-task learning into learning MOT research.

MOT is a fundamental yet challenging task in the computer vision field, which has been widely used in many real-world applications such as video surveillance systems, autonomous systems, and human computer interaction. The MOT requires multi-task learning that learns the shared information about: (i) object detection task classifies and localizes the presence of all objects over all frames, (ii) data association task associates detection results over the time-domain based on object identities. Conventionally, object detection task includes classification and regression (localization) sub-tasks, and data association is treated by the classification task. Accordingly, multi-task learning in MOT consists of one regression task and two classification tasks. If these tasks are related, combining all tasks into a single tracking model is to learn the complementary information across tasks by using a shared layer mechanism. This strategy reduces the computation cost and boosts the generalization performance. Otherwise, if these tasks are unrelated, learning all tasks together without prior knowledge can degrade the performance [2]. However, in the existing MOT methods [3]–[16], when jointly learning multiple tasks, they treat all tasks equally without investigating which tasks are related.

For the task balancing problem, in the optimization view, the joint learning of MOT tasks minimizes multiple loss functions during training. However, optimizing all task losses equally can lead to objective imbalance due to the opposite characteristics of the various tasks as follows: (1) The range of each task loss is inharmonious because the regression task takes input offsets in the logarithmic range with $[0, \infty)$, while the input to the classification objectives is normalized in the range $[0, 1]$; (2) The contribution of each task to the total loss is altered since the gradient norm of each task is different; and (3) The difficulties of the tasks are heterogeneous since the regression task only makes predictions on positive samples, while the classification objective computes predictions for all samples (e.g., negative samples and positive samples). Therefore, these reasons perturb the gradients when updating network weights and cause more challenging to balance loss values. Recent MOT methods [3]–[12], [14]–[16] used a weighted sum of objectives to the single total loss where hard weighting factors are employed to balance the ranges of each task. They state that the performance of the models is sensitive to the weighting factors. Usually, optimal factors are manually tuned by many experiments, which is expensive and difficult due to the high dimensions of the search space. Moreover, these weighting factors are fixed during training and do not reflect the relationship of the tasks since classification and box regression tasks have a positive correlation.

To address this problem, this paper presents a new Learned Weighting Factors (LWF) method that predicts weighting

factors via lightweight neural networks, dynamically balancing multiple loss functions. This strategy relies on a data-driven process, and thus it ignores manual tuning weighting factors. The correlation and difficulty of the tasks are revealed to guide these weighting factors at each iteration during training.

II. Related Works

Multi-task learning. In recent years, many methods have been proposed to improve the generalization performance of multi-task learning in task weighting [18]–[21].

In the existing literature, there are two ways that balance the multi-task losses for updating network weights in an optimal manner, balancing gradient magnitude [18] and assigning task-specific weighting factors [19]–[21]. SPMTL [19] presents a self-paced regularizer where parameters in this regularizer are generated to adjust the hardness level of tasks and task samples. GradNorm [18] dynamically balances the task-specific gradient magnitude based on $L1$ gradient normalization to guide the task-specific weighting factors. Motivated by the task priorities of SPMTL, Dynamic Task Prioritization [20] extends this idea to more various tasks, which focuses on the learning of difficult tasks. The method in [21] exploits homoscedastic uncertainty theory to balance multi-task losses, and the final weighting factors are derived by optimizing likelihood function. Unlike existing methods, this paper learns weighting factors via shallow neural network self-supervised by multi-task losses while other methods generate weighting factors based on task priorities [19], [20], gradient normalization [18], and task uncertainty [21].

MOT. In the existing literature, MOT methods are grouped into two main types: tracking-by-detection and joint-detection-and-tracking. In the tracking-by-detection methods [3]–[10], detection and data association tasks are treated in isolation, i.e., each separate network is learned for each task. Specifically, these methods firstly locate objects in each frame by detectors and then associate these detections over time based on Kalman motion prediction, Re-ID appearance features, and IoU similarity. DeepSORT [8] matches detected boxes by Re-ID features and associates to the the the next frame by Hungarian matching. POI [10] combines the re-implemented detector on extra datasets and deep learning-based Re-ID to improve the performance. In joint-detection-and-tracking methods [11]–[16], detection and data association tasks are jointly learned through a single network. Tracktor [11] uses the box results at the current frame as region proposals for the next frame and then refines these proposals via the regression head of the detector. CenterTrack [12] takes two adjacent frames and prior heatmaps as input and predicts center offsets for the current frame. JDE [13] and FairMOT [16] add one Re-ID branch to the head of the detector to obtain higher performance. CTracker [15] adds one ID verification branch for learning the IoU similarity between two frames to the object detector and proposes Joint Attention Module (JAM) to model the related task. However, in the multi-task learning view, most MOT methods [3]–[16] treat multi-task branches independently or balance multi-task losses equally. Although joint-detection-and-tracking methods inherit the benefit from multi-task learning, these methods weakly consider the importance of related tasks in both network and optimization.

III. The Proposed Method

In this section, we analyze objective imbalance problem originating from jointly learning multi-task losses of MOT in subsection III-A and propose a solution for this problem based on this analysis in subsection III-B.

Fig. 1 shows the overall architecture of the joint-detection-and-tracking network. The used backbone network is ResNet-50 pre-trained on ImageNet for feature extraction. Following common methods, FPN [17] is used for constructing multi-level feature maps. The proposed LWF method performs a weighted sum of three objectives ($\mathcal{L}_{cls}$, $\mathcal{L}_{reg}$, and $\mathcal{L}_{reid}$) where weighting factors are learned, shown in Fig. 2.

A. Objective Imbalance in MOT

To investigate the objective imbalance because of multi-task learning, we revisit the definition of each task loss. Based on this analysis, a novel learned weighting factors (LWF) method is proposed to solve the objective imbalance problem in object detection and data association tasks.

Conventionally, the classification objective $\mathcal{L}_{cls}$ is defined as:

$$\mathcal{L}_{cls} = \frac{1}{N_{pos}+N_{neg}} \sum_{i=1}^{N_{pos}+N_{neg}} FL(p_i, \hat{p}_i), \tag{1}$$

$$FL(p_i, \hat{p}_i) = -a(1-p_i)^b \log(\hat{p}_i), \tag{2}$$

where N_{pos}, and N_{neg} are the number of positive samples, and number of negative samples. Following by common methods, if the IoU (Intersection of Union) between the anchor box and the ground truth bounding box is greater than a threshold, this sample is considered as the positive sample, and otherwise. $FL(p_i, \hat{p}_i)$ indicates the Focal loss in which a, and b are balanced coefficients to control the contribution of hard samples. p_i, and $\hat{p}_i$ are the predicted classification score and class label. The range of the classification probability scores are limited by intervals $[0, 1]$ because of that:

$$p_i = \delta(x) = \frac{1}{1+e^{-x}}, \tag{3}$$

where δ is the sigmoid activation function normalizing the digit score x to output the confident scores for each class.

The regression objective $\mathcal{L}_{reg}$ is $smooth_{L1}$ loss:

$$L_{reg} = \frac{1}{N_{pos}} \sum_{i=1}^{N_{pos}} \sum_{j \in \{x,y,w,h\}} smooth_{L1}(o_{i,j} - \hat{o}_{i,j}), \tag{4}$$

where $o_{i,j}$, and $\hat{o}_{i,j}$ are the regressed offsets and transformed targets. Specifically, $o_{i,j} = \{o_{i,x}, o_{i,y}, o_{i,w}, o_{i,h}\}$ and $\hat{o}_{i,j} = \{\hat{o}_{i,x}, \hat{o}_{i,y}, \hat{o}_{i,w}, \hat{o}_{i,h}\}$ are the transformed coordinates (e.g.,

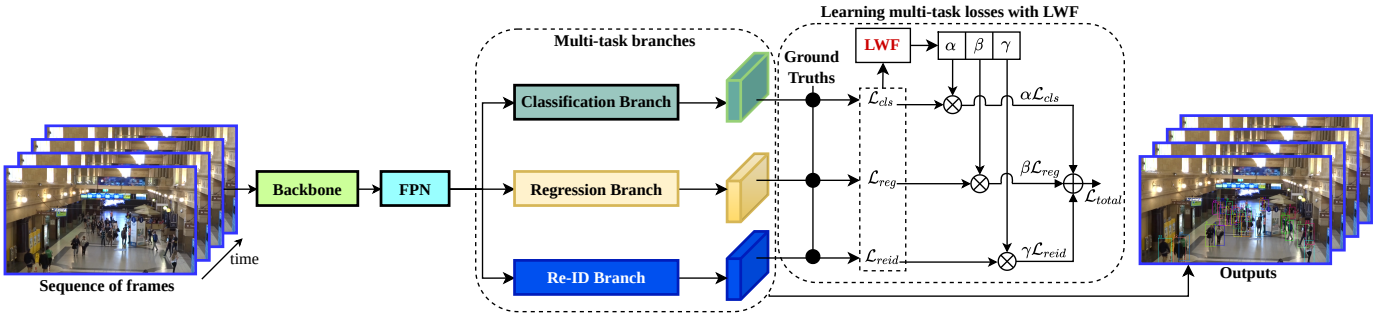


Fig. 1. The overall architecture of the multiple object tracking network includes four components: backbone network, feature pyramid network (FPN), multi-task branches, and multi-task losses. The input of this system is a sequence of frames generated videos with 30 frames per second. The backbone network extracts informative features from the images. The feature pyramid FPN [17] indicates multi-level feature maps with different scales. $\mathcal{L}_{cls}$, $\mathcal{L}_{reg}$, and $\mathcal{L}_{reid}$ are classification, regression, Re-ID objectives, which serve as the input of the Learned Weighting Factors (LMF) module. $\mathcal{L}_{total}$ is the total loss. α, β, and γ are coefficients learned by the LWF module. The output of the system is the localization of objects and identity numbers.

box center (x, y), spatial dimensions of the bounding box) generated by *log* function, defined by the recent detectors as:

$$\begin{aligned} o_{i,x} &= (x_i - x_i^a)/w_i^a, & o_{i,y} &= (y_i - y_i^a)/h_i^a, \\ o_{i,w} &= \log(w_i/w_i^a), & o_{i,h} &= \log(h_i/h_i^a), \\ \hat{o}_{i,x} &= (\hat{x}_i - x_i^a)/w_i^a, & \hat{o}_{i,y} &= (\hat{y}_i - y_i^a)/h_i^a, \\ \hat{o}_{i,w} &= \log(\hat{w}_i/w_i^a), & \hat{o}_{i,h} &= \log(\hat{h}_i/h_i^a), \end{aligned} \tag{5}$$

where $\{x_i, y_i, w_i, h_i\}$ denotes the offset prediction (box's center, width, and height) for positive sample i. $\{x_i^a, y_i^a, w_i^a, h_i^a\}$ is the coordinates of the anchor box. $\{\hat{x}_i, \hat{y}_i, \hat{w}_i, \hat{h}_i\}$ indicates the coordinates of the ground truth bounding box.

As shown in Equation 5, the box's normalized center $(o_{i,x}, o_{i,y})$ is still in real numbers and $(o_{i,w}, o_{i,h})$ is transformed by the log function. It is obvious that the input to the regression loss is converted to range $[0, \infty)$ while the input to the classification loss is normalized to intervals $[0, 1]$. This makes the range of each objective inconsistent. According to Equations 1 and 4, we observe that the classification task focuses on classifying all samples (negative samples and positive samples), while the regression task only calculates loss values for positive samples. Hence, learning the classification task is more difficult than the regression task, since negative samples can contain hard samples (IoU scores around the predefined threshold) that have higher loss values.

Following common methods [12], [13], [15], [16], the Focal loss is applied for the Re-ID objective computed as:

$$\mathcal{L}_{reid} = \frac{1}{N_{pos}} \sum_{i=1}^{N_{pos}} FL(id_i, \hat{id}_i), \tag{6}$$

where id_i, $\hat{id}_i$ are the identity confident scores, ID truth label based on IoU matching of same targets. Because Re-ID loss is the classification loss, the input to the loss is limited to the range $[0, 1]$.

Finally, the total loss $\mathcal{L}_{total}$ is the weighted sum of objectives, defined as:

$$\mathcal{L}_{total} = \alpha\mathcal{L}_{cls} + \beta\mathcal{L}_{reg} + \gamma\mathcal{L}_{reid}, \tag{7}$$

where α, β, and γ are the weighting factors fixed during training, i.e., hard weighting factors (HWF).

During optimization, the parameters are updated using Gradient Descent, defined as:

$$\begin{aligned} \theta_t &= \theta_{t-1} - lr \frac{\partial \mathcal{L}_{total}}{\partial \theta_{t-1}} \\ &= \theta_{t-1} - lr \frac{\partial(\alpha\mathcal{L}_{cls} + \beta\mathcal{L}_{reg} + \gamma\mathcal{L}_{reid})}{\partial \theta_{t-1}}, \end{aligned} \tag{8}$$

where θ is the network weights. lr is the learning rate. As shown in Equation 8, the performance of the network is sensitive to the weighting factors. If we do not balance the range of each objective, the model will heavily pay attention to a certain task. As a result, the regression loss values are always much larger than the classification loss and Re-ID loss values. Thus, the regression loss takes up the largest proportion of the total loss. It is proved in Equations 1, 4, and 6 since the number of input variables in each objective is different. Specifically, the regression objective takes four variables (box's center, width, and height) as input, while the classification objective only takes one variable as input. Hence, the tracking model focuses too much on the regression task, unnerving the advantage of multi-task learning.

From the above comprehensive analyses, it reveals the objective imbalance issue in jointly learning multi-task losses.

TABLE I
SEVERAL EXPERIMENTS WITH HWF ON THE MOT16 VALIDATION SET

α	β	γ	Training time (h)	MOTA↑	IDF1↑	MOTP↑
0.5	1.0	1.5	20	74.8	65.8	85.2
0.5	2.0	1.0	20	74.3	65.2	85.5
0.7	0.4	0.9	20	74.9	65.2	84.9
1.0	1.0	1.0	20	73.1	63.3	85.3
1.0	0.5	1.5	20	74.9	65.3	84.9
2.0	0.9	1.7	20	75.2	65.2	84.9

B. Learned Weighting Factors (LWF)

In conventional approaches [3]–[12], [14]–[16], the weighting factors are selected based on many experiments to get optimal values. It takes many hours to complete each trial. For

example, Table I describes the different values of weighting factors. The total training time takes 120h (5 days) for all experiments, which are measured by a Tesla V100 GPU. Because the search space with three tasks is large, it is difficult to find optimal weighting factors (the model achieves better performance at these values). Moreover, the weighting factors are fixed during training, and the network can not reflect the correlation of each task. Therefore, using the HWF strategy is a straightforward way to balance multi-task losses.

To avoid sub-optimal selection and manual tuning of weighting factors, this paper proposes learned weighting factors (LWF) operation, which leverages the relationship of detection and Re-ID tasks to predict weighting factors. Our strategy empowers the weighting factors process to be dynamic and learnable through lightweight neural networks. The weighting factors are formulated as shown in Fig. 2.

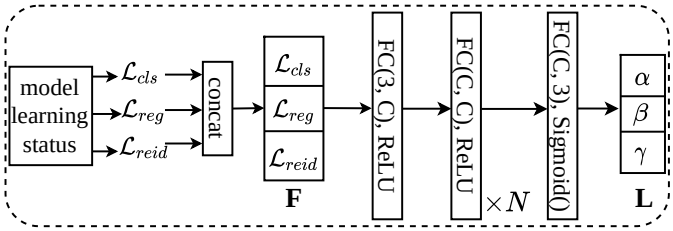


Fig. 2. The Learned Weighting Factors (LWF) sub-network is proposed to learn weighing factors self-supervised by three losses of MOT from the learning status. concat stands for a concatenation operation. FC(3, C) is the fully connected layer with input dimension 3, and output dimension C. N denotes the number of hidden layers.

Our key solution is to accommodate α, β, γ dynamically during training. As described in Fig. 1, the $\mathcal{L}_{cls}$, $\mathcal{L}_{reg}$, and $\mathcal{L}_{reid}$ are computed at every iteration, taken from the model learning status and these factors become free hyperparameters. In another aspect, the weighting factors are generated by considering the difficulty of each task. Thus, using multi-task losses from the learning status to generate weighting factors is an intuitive way and can adapt the network weights according to task-dependent. However, in this way, the gradient of weighting factors is ignored during backward pass. It is worth noting that weighting factors in this way are not supervised by the model during training, omitting the information of class labels, ground truth boxes, and ID labels.

To create weighting factors in an optimal way, our LWF method is proposed, which is described in Fig. 2. The LWF module is the sub-network self-supervised by multi-task losses, which takes $\mathcal{L}_{cls}$, $\mathcal{L}_{reg}$, and $\mathcal{L}_{reid}$ features as inputs to predict weighting factors for each task. The LWF network is simple, which includes several fully connected (FC) layers mapping concatenated features into a higher dimensional vector. These layers perform the global interaction of the prediction and objectives, which reflects the positive correlation between detection and Re-ID tasks. Since α, β, and γ are always positive values, the ReLU activation function is employed to avoid potential risks. The sigmoid function fully captures task dependencies, i.e., which objective is enabled to be emphasized at each iteration. Accordingly, the vector $\mathbf{L} = [\alpha, \beta, \gamma]^T$ for weighting factors can be calculated as:

$$\mathbf{F} = concat(\mathcal{L}_{cls}, \mathcal{L}_{reg}, \mathcal{L}_{reid}), \tag{9}$$

$$\mathbf{L} = \sigma(\mathbf{W}_3\delta_2(\mathbf{W}_2\delta_1(\mathbf{W}_1\mathbf{F}))), \tag{10}$$

where σ and δ indicate ReLU and sigmoid functions. $\mathbf{W}_1 \in \mathbb{R}^{3\times C}$, $\mathbf{W}_2 \in \mathbb{R}^{C\times C}$ and $\mathbf{W}_3 \in \mathbb{R}^{C\times 3}$ are linear transforms implemented by FC layers. In this way, the gradient of the LWF network is propagated to the overall network, and our LWF network is very lightweight, which only affects the training time of the network. The Pytorch code of the LWF sub-network is provided in Appendix.

IV. Experiments and Results

A. Datasets, Evaluation Metrics, and Implementation Details

The performance of the proposed method is evaluated on two benchmark datasets: MOT16 [22] and MOT20 [23]. Further information about these datasets is described in Table II, where #training and #testing indicate the number of training and testing videos, respectively.

TABLE II
Some descriptions of two benchmarks

Dataset	#training	#testing	Camera	Condition
MOT16	7	7	moving, static	outdoor
MOT20	4	4	static	indoor, outdoor

More importantly, in this paper, we only train the model on the training set of the MOT16 or MOT20 while CenterTrack [12], JDE [13], and FairMOT [16] use combinations of other large-scale datasets for training. Thus, we do not include some methods in this paper for fair comparisons.

All tracking performances are measured by three standard metrics: Multiple Object Tracking Accuracy (MOTA), ID F1 score (IDF1) defined by CLEAR MOT, and Higher Order Tracking Accuracy (HOTA).

All experiments are conducted by the deep learning Pytorch framework. The backbone ResNet-50 is pre-trained on the dataset ImageNet. The weight initialization of the newly added convolutional layers in the FPN, multi-task branches, and LWF module is fulfilled from the Gaussian distribution. The GPU Tesla V100 device with Cuda 10.2, and CuDNN 7.6.5 is used to train the model for 100 epochs with a batch size of 8. The Adam optimizer is applied for minimizing the detection and Re-ID objectives. The learning rate is set to $5\times e^{-5}$, and the number of anchor boxes tiled per one feature location is set to $A = 1$ for all implementations.

B. Results

This subsection analyzes the main tracking results computed on the testing set of two benchmarks in subsection IV-B1, as well as the ablation study carried out on the MOT16 validation set in subsection IV-B2. Qualitative results in surveillance systems, and additional experiments of the proposed method are provided in the Appendix.

TABLE III
COMPARISON WITH STATE-OF-THE-ART METHODS ON THE TESTING SETS OF THE MOTCHALLENGE BENCHMARKS

Dataset	Method	MOTA↑	IDF1↑	MOTP↑	MT↑	ML↓	FP↓	FN↓	IDS↓
MOT16 [22]	DMAN [3]	46.1	54.8	73.8	17.4	42.7	7909	89874	532
	MOTDT [4]	47.6	50.9	74.8	15.2	38.3	9253	85431	792
	BLSTM-MTP [5]	48.3	53.5	-	17.0	38.7	9792	83707	735
	Tracktor [11]	54.4	52.5	78.2	19.0	36.9	3280	79149	682
	MPNTrack [6]	58.6	61.7	78.9	27.3	34.0	4949	70252	**354**
	TADAM [7]	59.1	59.5	-	-	-	**2540**	71542	529
	DeepSORT [8]	61.4	62.2	79.1	32.8	**18.2**	12852	56668	781
	ArTIST [9]	63.0	61.9	-	29.1	33.2	7420	59376	635
	JDE [13]	64.4	55.8	-	**35.4**	20.0	-	-	1544
	POI [10]	66.1	**65.1**	**79.5**	34.0	20.8	5061	55914	805
	CTracker [15]	67.6	57.2	78.4	32.9	23.1	8934	**48305**	1897
	Ours	**69.2**	58.4	78.9	32.3	24.1	6036	48579	1628
MOT20 [23]	SORT20 [24]	42.7	45.1	-	16.7	26.2	27521	264694	4470
	Tracktor++ [11]	51.3	**47.6**	-	24.9	26.0	**16263**	253680	**2584**
	Ours	**53.5**	45.6	**76.9**	**35.0**	**19.8**	42702	**191973**	6156

1) Comparison with State-of-the-art Methods: In this subsection, we describe the main tracking results of the proposed network on testing sets of the MOTChallenge benchmark, listed in Table III. The bold font denotes the best result across all state-of-the-art methods. Since the test sets of MOT benchmarks are not provided for evaluation, all the tracking results are uploaded to the official MOT evaluation protocols.

Our proposed network achieves state-of-the-art performances on two datasets in terms of the MOTA score and IDF1. In MOT16, the proposed network achieves an MOTA score of 69.2%, which outperforms all other trackers by a clear margin. More specifically, our method outperforms DeepSORT [8] by 7.8%, JDE [13] by 4.8%, POI [10] by 3.1%, and strong method CTracker [15] by 1.6%, respectively. In recent MOT20 dataset, we compare our method with SORT20 [24], and Tracktor++ [11], which achieves better performance among them.

2) Ablation Study:

a) Hyperparameters in LWF: The experiments are conducted to investigate how the model is affected by the number of hidden channels C and hidden layers N in the LWF module. These results are shown in Table IV and Table V.

TABLE IV
THE EFFECTS OF C ON THE PERFORMANCE

C	MOTA↑	IDF1↑	MOTP↑	MT↑	FP↓	#params
4	74.9	66.0	85.3	279	1926	0.04k
8	75.2	66.2	85.2	272	1881	0.11k
16	75.3	66.5	85.7	279	1653	0.35k
32	**76.2**	67.0	85.9	286	1709	1.22k
64	75.9	67.1	85.9	284	1763	4.48k
128	75.5	67.0	85.8	285	1965	17.15k

Table IV shows that the performance is best when the number of hidden channels $C = 32$. The results are saturated when employing too high C. The explanation for this phenomenon is that setting $C = 32$ is sufficient to contain the informative features extracted from objective values of all images.

We report the effects of the number of hidden channels N on the tracking performance by fixing $C = 32$ and changing N. Table V shows that stacking more FC layers in the LWF module degrades performance, which is due to the fact that the network is overfitting to the input objective $[\mathcal{L}_{cls}, \mathcal{L}_{reg}, \mathcal{L}_{reid}]^T$. The LWF sub-network only brings 1.22k (thousand) of parameters, which is tiny compared to the tracking model.

TABLE V
THE EFFECTS OF N ON THE PERFORMANCE.

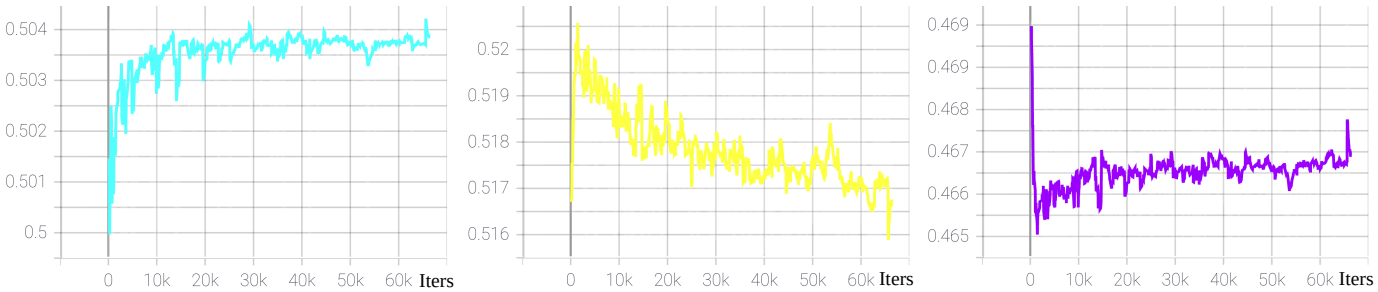
N	MOTA↑	IDF1↑	MOTP↑	MT↑	FP↓	#params
1	**76.2**	67.0	85.9	286	1709	1.22k
2	76.0	66.6	85.8	280	1863	2.24k
3	74.8	65.9	85.2	259	1691	3.26k

b) Behaviors of learned weighting factors: As shown in Fig. 3, the learning curves of weighting factors are reciprocal constraints toward balancing detection and Re-ID objectives. The curves of factor α and γ tend to increase weighting values while the curve of factor β shows the opposite influence. It is easy to understand that the regression loss is always larger than classification and Re-ID losses (discussed in Section III-A of the manuscript). Hence, the weighting factor β is lower to control the regression objective consistent with the other objectives, reducing the contribution of the regression task to the overall gradient. From the above analyses, we confirm that the proposed LWF can leverage multi-task learning in practical applications.

TABLE VI
TIME COMPLEXITY ANALYSIS

Method	MOTA	GFLOPs	#params	FPS
Baseline	74.9	64.73	38.62	15.64
Ours	76.6	51.85	36.25	16.79

c) Time cost analysis: Table VI shows the inference speed (FPS - frames per seconds) of the baseline and our proposed network. These FPS values are measured on the same computer device with a single Tesla V100 GPU. As expected, our method outperforms the baseline in terms of tracking accuracy and inference speed. Specifically, our network achieves



(a) weighting factor α (b) weighting factor β (c) weighting factor γ

Fig. 3. The curves illustrate learning behaviors of weighting factors under the supervision of the LWF sub-network. The horizontal axis indicates the number of training iterations (iters). The vertical axis describes the weighting factor values.

a MOTA score of 76.6% and 16.79 FPS, while the baseline gets a MOTA score of 74.9% and 15.64 FPS.

V. Conclusion

This paper introduced a novel Learned Weighting Factors (LWF) module, which dynamically balances multi-task losses in the MOT during training. The lightweight LWF module attached to the MOT network learns weighting factors self-supervised by multiple objectives. It is a new and different perspective in solving multi-task learning and specific MOT task. The proposed method is evaluated on the MOT16, and MOT20 benchmarks, achieving state-of-the-art performance. We hope that our method can serve as the simple baseline for multi-task learning research. In the future, the proposed method will be applied to multiple high-level tasks such as abnormal action detection, human pose tracking, and human behavior detection in video surveillance systems.

References

[1] R. Caruana, "Multitask learning," *Machine learning*, vol. 28, no. 1, pp. 41–75, 1997.

[2] S. Vandenhende, S. Georgoulis, W. Van Gansbeke, M. Proesmans, D. Dai, and L. Van Gool, "Multi-task learning for dense prediction tasks: A survey," *IEEE transactions on pattern analysis and machine intelligence*, 2021.

[3] J. Zhu, H. Yang, N. Liu, M. Kim, W. Zhang, and M.-H. Yang, "Online multi-object tracking with dual matching attention networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 366–382.

[4] L. Chen, H. Ai, Z. Zhuang, and C. Shang, "Real-time multiple people tracking with deeply learned candidate selection and person re-identification," in *2018 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2018, pp. 1–6.

[5] C. Kim, L. Fuxin, M. Alotaibi, and J. M. Rehg, "Discriminative appearance modeling with multi-track pooling for real-time multi-object tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 9553–9562.

[6] G. Brasó and L. Leal-Taixé, "Learning a neural solver for multiple object tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 6247–6257.

[7] S. Guo, J. Wang, X. Wang, and D. Tao, "Online multiple object tracking with cross-task synergy," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 8136–8145.

[8] 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.

[9] F. Saleh, S. Aliakbarian, H. Rezatofighi, M. Salzmann, and S. Gould, "Probabilistic tracklet scoring and inpainting for multiple object tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021, pp. 14 329–14 339.

[10] 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 *European Conference on Computer Vision*. Springer, 2016, pp. 36–42.

[11] P. Bergmann, T. Meinhardt, and L. Leal-Taixe, "Tracking without bells and whistles," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 941–951.

[12] X. Zhou, V. Koltun, and P. Krähenbühl, "Tracking objects as points," in *European Conference on Computer Vision*. Springer, 2020, pp. 474–490.

[13] Z. Wang, L. Zheng, Y. Liu, Y. Li, and S. Wang, "Towards real-time multi-object tracking," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16*. Springer, 2020, pp. 107–122.

[14] B. Shuai, A. Berneshawi, X. Li, D. Modolo, and J. Tighe, "Siammot: Siamese multi-object tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 12 372–12 382.

[15] J. Peng, C. Wang, F. Wan, Y. Wu, Y. Wang, Y. Tai, C. Wang, J. Li, F. Huang, and Y. Fu, "Chained-tracker: Chaining paired attentive regression results for end-to-end joint multiple-object detection and tracking," in *European Conference on Computer Vision*. Springer, 2020, pp. 145–161.

[16] Y. Zhang, C. Wang, X. Wang, W. Zeng, and W. Liu, "Fairmot: On the fairness of detection and re-identification in multiple object tracking," *International Journal of Computer Vision*, vol. 129, no. 11, pp. 3069–3087, 2021.

[17] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

[18] Z. Chen, V. Badrinarayanan, C.-Y. Lee, and A. Rabinovich, "Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks," in *International Conference on Machine Learning*. PMLR, 2018, pp. 794–803.

[19] C. Li, J. Yan, F. Wei, W. Dong, Q. Liu, and H. Zha, "Self-paced multi-task learning," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[20] M. Guo, A. Haque, D.-A. Huang, S. Yeung, and L. Fei-Fei, "Dynamic task prioritization for multitask learning," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 270–287.

[21] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7482–7491.

[22] 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.

[23] P. Dendorfer, H. Rezatofighi, A. Milan, J. Shi, D. Cremers, I. Reid, S. Roth, K. Schindler, and L. Leal-Taixé, "Mot20: A benchmark for multi object tracking in crowded scenes," *arXiv preprint arXiv:2003.09003*, 2020.

[24] 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)*. IEEE, 2016, pp. 3464–3468.

APPENDIX

A. Pytorch code of the LWF sub-network

The pseudo-code of the Learned Weighting Factors (LWF) module is shown in the Algorithm 1. Here, *nn.Linear* indicates Fully-Connected (FC) layer.

Algorithm 1 Pytorch code of the LWF sub-network

```
import torch
import torch.nn as nn

# C is the number of hidden channels

###initial_layers####
LWF_net = nn.Sequential(
    nn.Linear(3, C),
    nn.ReLU(),
    nn.Linear(C, C),
    nn.ReLU(),
    nn.Linear(C, 3),
    nn.Sigmoid())
def LWF(cls_loss, reg_loss, reid_loss):
    # cls_loss (tensor): classification objective
    # reg_loss (tensor): regression objective
    # reid_loss (tensor): Re-ID objective

    # convert tensor to float type
    cls_loss  = cls_loss.detach().data
    reg_loss  = reg_loss.detach().data
    reid_loss = reid_loss.detach().data

    # concatenate three variables to row vector
    cat_loss = torch.cat((cls_loss.unsqueeze(0),
                          reg_loss.unsqueeze(0),
                          reid_loss.unsqueeze(0)),
                          dim=0).unsqueeze(1)

    # transpose row vector to column vector
    cat_loss = cat_loss.transpose(0, 1)

    # LWF sub-network
    output = LWF_net(cat_loss)

    # learned weighting factors
    alpha = output[:, 0]
    beta  = output[:, 1]
    gamma = output[:, 2]
    return alpha, beta, gamma
```

B. Additional experimental results

The detailed performances of our method on testing sets of MOTChallenge benchmarks are listed in Table VII. In the following, we provide some experiments for comparisons, measurements of inference time, and visualization.

a) LWF vs. other weighting factor methods: The comparative performance between the LWF method and other weighting factor strategies is described in Table VIII. To make practicable comparisons, we replace the LWF module with the Uncertainty Weighting method [21] for balancing detection and Re-ID losses. As a result, our proposed LWF method achieves the best performance among them. It demonstrates that our method can serve as a baseline for multi-task learning.

TABLE VII
THE PERFORMANCE ON EACH VIDEO OF THE MOT16 TESTING SET

Video	MOTA↑	IDF1↑	MOTP↑	MT↑	FP↓	IDs↓
MOT16-01	47.8	39.9	76.9	7	169	63
MOT16-03	86.9	66.8	78.7	124	3785	493
MOT16-06	55.9	56.3	78.4	64	517	207
MOT16-07	51.7	41.6	77.8	12	538	229
MOT16-08	37.0	35.1	83.1	11	352	172
MOT16-12	44.5	53.2	81.0	14	198	67
MOT16-14	40.1	43.1	78.4	73	477	397
Overall	69.2	58.4	78.9	245	6036	1628

TABLE VIII
COMPARISON WITH OTHER WEIGHTING FACTOR METHODS

Method	MOTA↑	IDF1↑	MOTP↑	MT↑	FP↓
HWF	74.9	66.6	85.3	260	1794
Uncertainty [21]	75.7	66.6	85.8	278	1878
LWF (Ours)	76.2	67.0	85.9	286	1709

b) Error analysis: The tracking error includes detection errors, localization errors, and association errors, which are shown in Fig. 4. The proposed method achieves an average

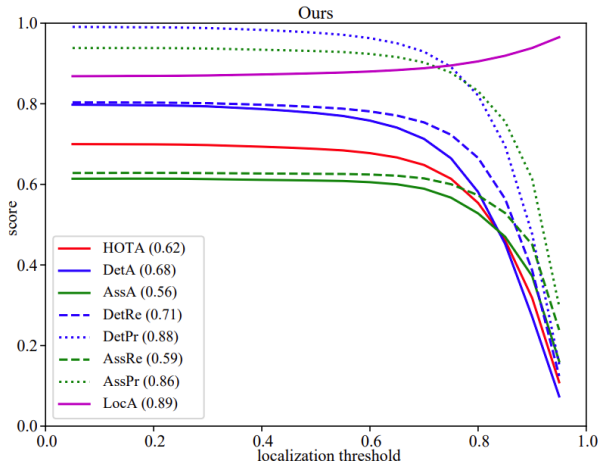


Fig. 4. HOTA score and error components at different localization thresholds.

HOTA score of 0.62 from threshold=0.05 to threshold=0.95 (localization threshold) with a step size of 0.05. DetA got a score of 0.68, which indicates detection accuracy. This is decomposed into DetRe (detection recall) and DetPr (detection precision). The association accuracy (AssA) measures the overlap between the predicted trajectories and ground truth, which consists of AssRe (association recall) and AssPr (association precision). Finally, LocA (localization accuracy) achieves a score of 0.89.