Discriminative Appearance Modeling with Multi-track Pooling for Real-time Multi-object Tracking

Chanho Kim [1] Li Fuxin [2] Mazen Alotaibi [2] James M. Rehg [1]
[1]Georgia Institute of Technology [2]Oregon State University

1. Network Architecture Details

In this section, we present the network architectures used in our experiments.

1.1. Track Proposal Classifier

Table 3 shows the network architecture of our joint appearance and motion model. We used this network to test the proposed approach on the MOT Challenge Benchmarks in Table 6, 7, and 8 of the main paper. Table 4 shows the appearance baseline model that we used in Table 2, 3, 4, and 5 of the main paper.

1.2. Track Proposal Classifier Input

For the appearance models, we first resize the input image to 64×128 (width, height) and then subtract the ImageNet mean from the image. For the motion model, the location and scale of the detection bounding box is used as input. The motion model input is represented by a 4-dimensional vector $(\frac{x^{\text{topleft}}}{\text{image width}}, \frac{y^{\text{topleft}}}{\text{image height}}, \frac{w}{\text{image width}}, \frac{h}{\text{image height}})$. We normalize this motion input by subtracting the mean and then dividing by the standard deviation before we feed it into the motion model. The mean and standard deviation are calculated using the ground truth bounding boxes in the training videos.

1.3. Bounding Box Corrector

Following the bounding box regressor presented in [2], we regress four scalars that correct the location and scale of the original bounding box from the input image. We also classify true and false positive detections using the same input image by utilizing an additional prediction head. The network architecture that we use for the box corrector is shown in Table 5. We utilize the same CNN as the one used in a track proposal classifier. Specifically, once the track classifier is trained, we freeze all the CNN weights and then train the linear layers in the two prediction heads from scratch. We use raw DPM detections provided by the MOT16 Challenge organizers as our training data to train this model. We use this model when we run our tracker without refining the input detections using Tracktor (i.e. Table 7 and 8 of the main paper).

Training Set
MOT17 - {02, 04, 05, 09, 10, 11, 13} MOT15 - {PETS09-S2L1, ETH - (Sunnyday, Bahnhof), TUD - (Campus, Stadtmitte), KITTI-(13, 17)}, ETH - (Jelmoli, Seq01), KITTI - (16, 19), PETS09-S2L2, TUD-Crossing, AVG-TownCentre
Validation Set
CVPR 19 (MOT Challenge) - {01, 02, 03, 05}
Test Set
MOT17 - {01, 03, 06, 07, 08, 12, 14}

Table 1. Split 1

Training Set
MOT15 - {PETS09-S2L1, ETH - (Sunnyday, Bahnhof), TUD - (Campus, Stadtmitte), KITTI-(13, 17)}, ETH - (Jelmoli, Seq01), KITTI - (16, 19), PETS09-S2L2, TUD-Crossing, AVG-TownCentre
Validation Set
MOT17 - {02, 04, 05, 09, 10, 11, 13}
Test Set
MOT17 - {01, 03, 06, 07, 08, 12, 14}

Table 2. Split 2

2. Additional Training Details

In this section, we explain our training settings for the experiments in the main paper.

2.1. Dataset

Table 1 and 2 show the training, validation, and test sets that we used in our experiments. For the final MOT Challenge Benchmark results in Table 6, 7, and 8 of the main paper, the training sequences in Table 1 were used as the training data.

2.2. Training Setting - Appearance Model

We used the SGD optimizer with the initial learning rate of 0.005 for Bilinear LSTM and the initial learning rate of 0.0005 for ResNet 50 (pre-trained on ImageNet). We

Soft-max	2								
FC	2								
FC-relu	24								
Concatenation	24								
2× FC-relu	16							2× FC-relu	8
Concatenation ($\mathbf{m}_t^{\text{all}}$)	16								
Matrix-vector Multiplication-relu ($\mathbf{m}_t^+$)	8			Max-pooling ($\mathbf{m}_t^-$)					
				Matrix-vector Multiplication-relu ($\mathbf{M}_t^-$)	$(M-1)\times 8$				
Reshape	8×256	Reshape	256×1	Reshape	$(M-1)\times 8\times 256$	Reshape	256×1	FC-relu	8
LSTM	2048			LSTM	$(M-1)\times 2048$			Group Normalization	64
FC-relu	256	FC-relu	256	FC-relu	$(M-1)\times 256$	FC-relu	256	LSTM	64
ResNet-50	2048	ResNet50	2048	ResNet-50	$(M-1)\times 2048$	ResNet50	2048	FC-relu	64
$\mathbf{x}_{t-1}^+$	$128\times 64\times 3$	$\mathbf{x}_t$	$128\times 64\times 3$	$\mathbf{x}_{t-1,i}^-$	$(M-1)\times 128\times 64\times 3$	$\mathbf{x}_t$	$128\times 64\times 3$	$\mathbf{x}_t^{\text{location, scale}}$	4

Table 3. The appearance + motion model used to generate the results in Table 6, 7, and 8 of the main paper. The number of non-target object tracks used in the multi-track pooling module is represented by $M-1$.

Soft-max	2		
FC	2		
Matrix-vector Multiplication-relu ($\mathbf{m}_t$)	8		
Reshape	8×256	Reshape	256×1
LSTM	2048		
FC-relu	256	FC-relu	256
ResNet-50	2048	ResNet50	2048
$\mathbf{x}_{t-1}$	$128\times 64\times 3$	$\mathbf{x}_t$	$128\times 64\times 3$

Table 4. The baseline appearance model that we compared with our proposed appearance model

FC	4	Soft-max	2
		FC	2
FC-relu	512	FC-relu	512
Reshape			16384
ResNet-50 (Block4)			$4\times 2\times 2048$
$\mathbf{x}_{t-1}^+$			$128\times 64\times 3$

Table 5. Bounding Box Corrector. ResNet-50 is shared with the one in Table 3.

Soft-max	2
FC	2
FC-relu	8
Group Normalization [5]	64
LSTM	64
FC-relu	64
$\mathbf{x}_t^{\text{location, scale}}$	4

Table 6. Motion model. We first train this motion model from scratch and use the pre-trained weights when we train the joint model in Table 3.

trained the model with the initial learning rate for the first 4 epochs ($\sim$ 120k iterations), and then reduced the learning rate with the decay factor of 0.1 for the next 4 epochs and reduced it one more time for the last 4 epochs ($\sim$ 360k iterations in total).

For the actual tracking episodes, we used truncated BPTT with a temporal window size of 10. We used all the ground truth tracks in the current frame as our training data so the mini-batch size in this case was equal to the number of ground truth tracks in the current frame.

For the random tracking episodes, we used 40 frames as the maximum frame gap between the randomly selected start and end frame. Thus, each mini-batch contains a track whose length is up to 40. Due to the limited GPU memory, the number of tracks for random tracking episodes was limited by N_{max}. We used $N_{\text{max}} = 8$ in our experiments.

2.3. Training Setting - Motion Model

We used the Adam optimizer [3] with the initial learning rate of 0.001 for training the motion model in Table 6. We trained the motion model with the initial learning rate for the first 4 epochs ($\sim$ 120k iterations) and then reduced the learning rate with a decay factor of 0.1 for the next 2 epochs and reduced it one more time for the last 6 epochs ($\sim$ 360k iterations in total).

2.4. Training Setting - Appearance and Motion Model

We first trained the appearance and motion models separately as described above before jointly training the model presented in Table 3. When the joint model was trained, new layers (i.e. top five rows in Table 3) were trained from scratch, and the rest of the layers (except for ResNet 50 in which all the weights were frozen) were fine-tuned from the pre-trained models. We used the SGD optimizer with the initial learning rate of 0.005 for the new layers and 0.0005 for the pre-trained layers. We trained the model with the initial learning rate for the first 2 epochs ($\sim$ 60k iterations), and then reduced the learning rate with the decay factor of 0.1 for the next 2 epochs ($\sim$ 120k iterations in total).

When we trained the joint model, we realized that the appearance features can be stronger than the motion features in the beginning of the training. As a result, our resulting model heavily relied on the appearance features, often ignoring the motion features. In order to make our model balance between these two types of features, we placed a dropout layer on the appearance features right before we concatenated the appearance and motion features (i.e. after the 2× FC-relu layer on the appearance side in Table 3). In the beginning of the training, we randomly dropped appearance features and then gradually decreased the drop rate as the training proceeded. This trick prevented the joint model from relying too much on the appearance cues in the early

training stage. In our experiments, we used 0.9 as the drop rate for the first ∼19k iterations, 0.6 for the next ∼10k iterations, 0.3 for the next ∼9k iterations, and 0.0 for the rest of the training.

2.5. Online Hard Example Mining

We found that online hard example mining can improve the model performance. We trained all the model with all the training examples for the first two epochs (∼60k iterations). We trained the models for the remaining epochs with top k hard examples (i.e. k examples that incurred high loss values) in the mini-batch. We used $k = 30$ in our experiments.

2.6. Missing Detection Augmentation

We randomly drop the bounding boxes in the tracks for missing detection augmentation. For each track in each mini-batch, we randomly choose the missing detection rate from a probability between 0.05 and 0.95. After selecting the missing detection rate, we randomly drop the bounding boxes in the selected track according to the selected rate.

2.7. Noisy Track Augmentation

In addition to using the ground truth tracks as the training data, we also generate tracks from noisy object detections. Given ground truth tracks and noisy detections, one can assign correct track IDs to noisy detections by finding an assignment that maximizes the Intersection over Union score (IoU) between ground truth tracks and object detections. The MOT Challenge Benchmark provides public object detections from three detectors (DPM [1], FRCNN [4], SDP [6]). Thus, we generate three additional sets of noisy tracks constructed from these public detections. Note that the track-detection assignments are optimal although the resulting tracks are noisier than the original ground truth tracks. Localization and missing detection errors caused by the object detector are embedded naturally in such tracks.

References

[1] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010. 3

[2] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 1

[3] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 2

[4] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 3

[5] Yuxin Wu and Kaiming He. Group normalization. In *ECCV*, 2018. 2

[6] Fan Yang, Wongun Choi, and Yuanqing Lin. Exploit all the layers: Fast and accurate cnn object detector with scale dependent pooling and cascaded rejection classifiers. In *CVPR*, 2016. 3