Deep Affinity Networks for Multiple Object Tracking

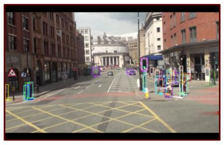
Darren Mei, Duncan MacWilliams, Aditya Khandelwal ***CS 230 Deep Learning***

Problem Definition

Multiple object tracking builds off of object detection and assigns identities that are consistent throughout a video stream. Using the Multiple Object Tracking (MOT) dataset, this project solved this problem with a Deep Affinity Network. With tweaks to the original implementation, our model performs slightly better and generalizes well to the test set. Additionally, it resolves many of the identity switching issues brought up in the baseline.

Data and Features

- The MOT17 training set consists of 21 videos 30-60 seconds long and taken at 14-30 fps.
- Since only the training set contains ground truth annotations, we split the 21 videos into 17 for the training set, 2 for evaluation, and 2 for our final test results.



- The inputs are the color image frames as well as the precomputed bounding box centers
- These bounding boxes are given by the MOT17 dataset, and were found with standard object detection algorithms like FRCNN, SDP, and DPM

Evaluation Metrics

$$\text{MOTA} = 1 - \frac{\sum_t (m_t + f_{p_t} + mme_t)}{\sum_t g_t}$$

$$\text{MOTP} = \frac{\sum_{i,t} d_{i,t}}{\sum_t c_t}$$

IDF1: The ratio of correctly identified detections over the average number of ground truth and computed detections

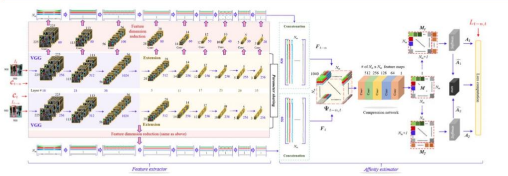
MT: Mostly tracked targets. The number of ground truth trajectories that are covered by a track prediction for at least 80% of their spans

Models

SORT Baseline

- Simple Online and Realtime Tracking assigns detections to targets without deep learning
- Assignments are calculated using IOU distance between each previous frame detection and all current frame bounding boxes

Deep Affinity Network



Loss Functions

$$L_f(\mathbf{L}_1, A_1) = \frac{\sum(\mathbf{L}_1 \odot (-\log A_1))}{\sum(\mathbf{L}_1)}$$

$$L_b(\mathbf{L}_2, A_2) = \frac{\sum(\mathbf{L}_2 \odot (-\log A_2))}{\sum(\mathbf{L}_2)}$$

$$L_c(A_1, A_2) = ||A_1 - A_2||_1$$

$$L_a(\mathbf{L}_3, A_1, A_2) = \frac{\sum(\mathbf{L}_3 \odot (-\log(max(A_1, A_2))))}{\sum(\mathbf{L}_3)}$$

$$L = \frac{L_f + L_b + a_c + L_c}{4}$$

Results and Conclusions

Experiments

Modified DAN Network:

-We noticed that the architecture did not involve dropout in any of the convolutional layers and thought it would be beneficial to add a dropout with low drop probability to the network at these convolutional stages (p=0.1)

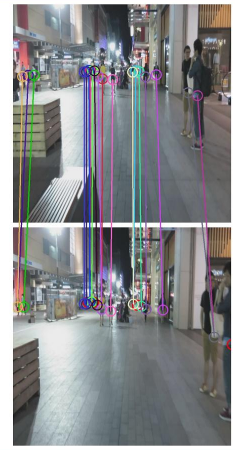
-We also switched the order of the activation and batch normalization layers so that we do not normalize over negative values that we will throw out with subsequent activation

Results

Dataset	MOTA	MOTP	IDF1	MT
SORT Baseline				
Train Set	41.75%	0.173	43.51%	13.765
Val Set	38.65%	0.133	5.65%	10
Test Set	45.70%	0.189	8.25%	13.5
Original DAN				
Train Set	39.82%	0.212	45.78%	13.706
Val Set	39.10%	0.1605	46.45%	18
Test Set	50.85%	0.217	50.30%	15
Modified DAN				
Train Set	39.88%	0.213	46.24%	13.765
Val Set	39.30%	0.1605	47.00%	17.5
Test Set	51.05%	0.2175	51.20%	15

Model Output

Error Example



Conclusion and Future Work

- Best Model: Modified DAN
- Results of Test Set: MOTA: 51.05% MOTP: 0.218IDF1: 51.20% MT: 15
- Future work: Modifications to the network architecture, potentially substituting ResNet for the VGG16-like subnetwork, replacing convolutional layers with a pyramid structure, and further fine tuning of the hyperparameters

Acknowledgments

We would like to thank the teaching staff for hosting a great course and Ashwin for discussing our project in office hours. We would also like to thank the MOT challenge for providing a great dataset and Sun et al. for providing an interesting model architecture to study and work with.