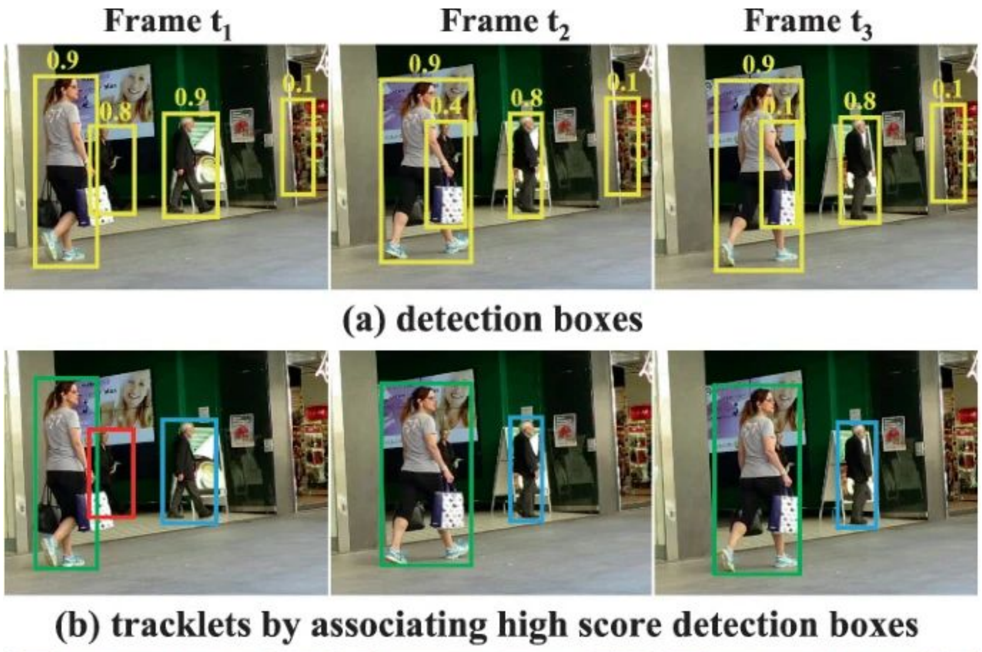
ByteTrack: Multi-object Tracking by Associating Every Detection Box

Presentation by: Sara Larson, Ethan Sims

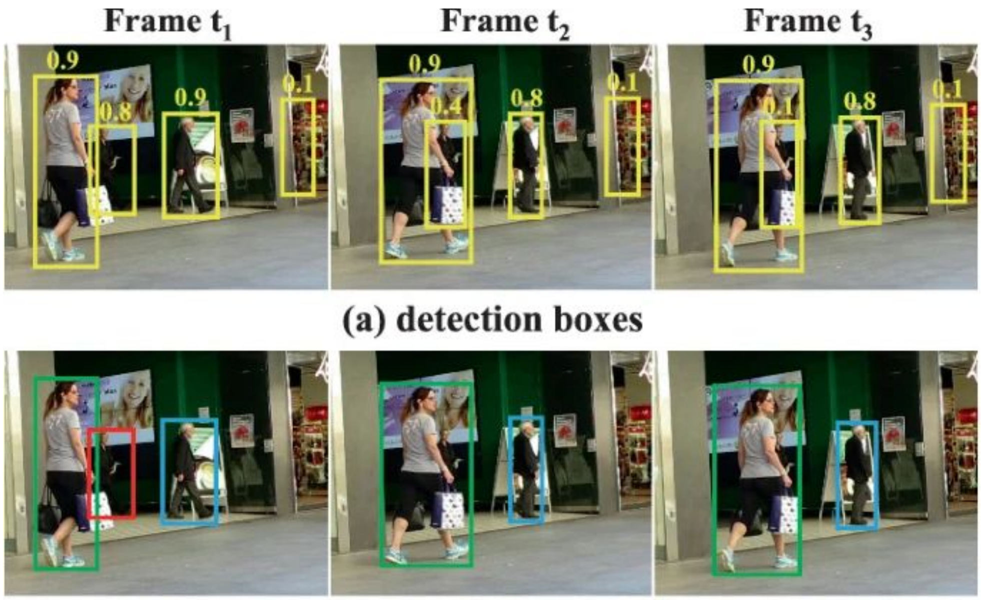
The Problem

- Multi-object tracking
  - Computer vision
  - Object detection in videos
  - Application areas include:
    - Autonomous driving
    - Sports analytics
    - Surveillance
    - etc.
- Most effective paradigm is tracking-by-detection
  - Object detection on each frame
  - Use detections to guide tracking



Tracking by Detection

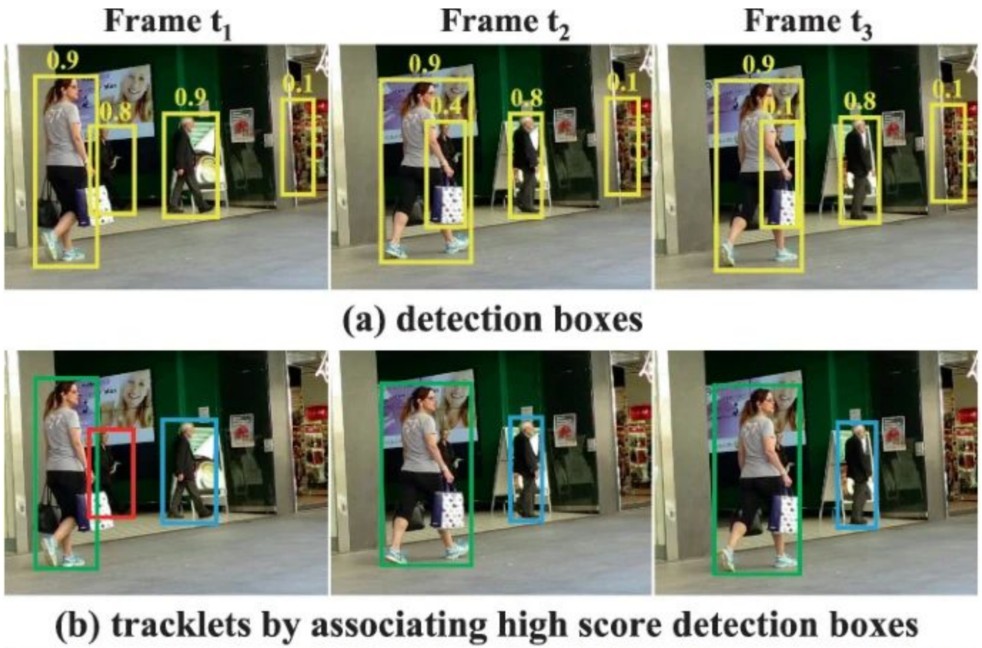
- True positive / false positive trade off
- Eliminate low confidence boxes
    - Based on some threshold value (0.5 for image)
    - Can lead to missed objects and / or tracking inconsistencies
- Issues arise with motion blur and occlusion
    - Use previous frames to address problem



(b) tracklets by associating high score detection boxes

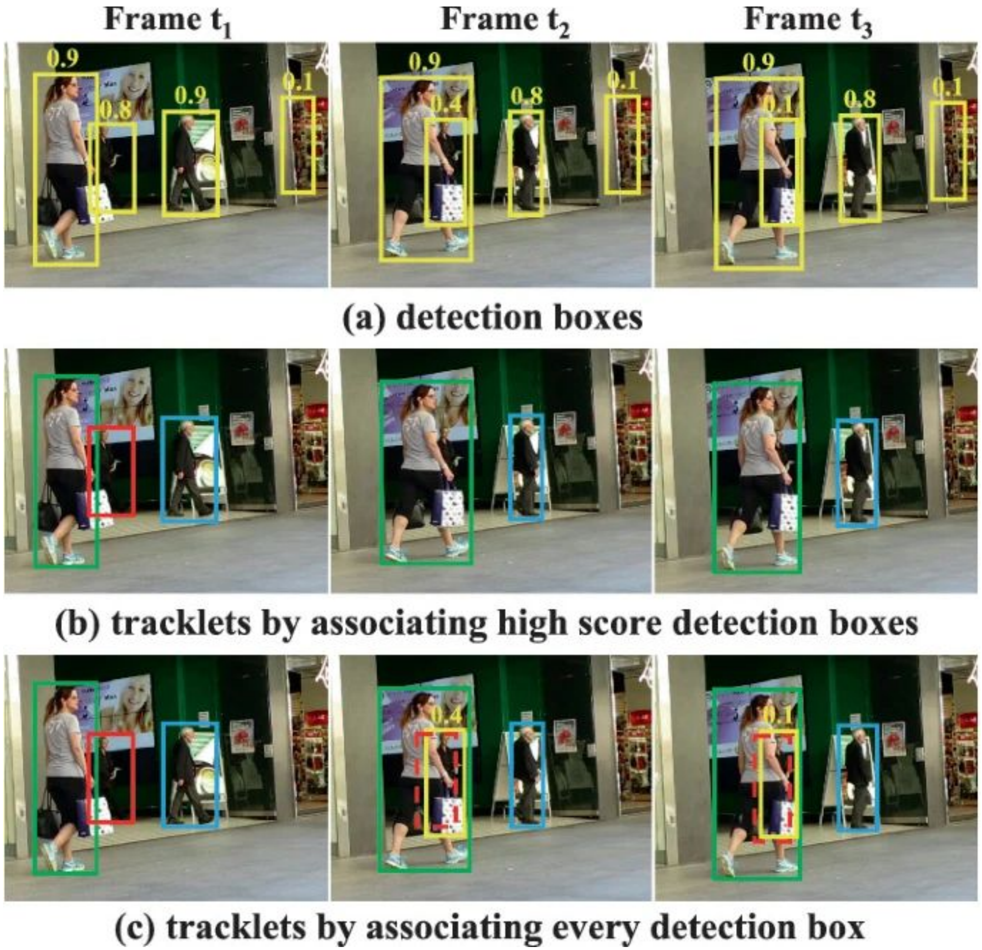
Detection by Tracking

- Use tracking to help define detection boxes
- Predict tracklet locations in next frame, merge prediction with detection
- Propagate boxes between frames



Key Idea

- Most approaches only keep detection boxes above some threshold
- This loses some objects that are properly tracked but with low confidence
- **Keep all detection boxes and associate across all of them**
    - **Increase recall while maintaining precision**



Data Association - BYTE

- Data association is the process of matching objects between frames
- BYTE is this paper's solution to this problem
    - First, associates high scoring detection boxes with tracklets
        - Some tracklets might not match
    - Then, associates low scoring detection boxes with unmatched tracklets
        - Minimizes number of unmatched tracklets
        - Removes detection boxes which are not actually objects
    - Any unmatched tracks at this point are deleted
    - Any unmatched, high scoring boxes are turned into tracks

Algorithm 1: Pseudo-code of BYTE.

Input: A video sequence `V`; object detector `Det`; detection score threshold τ
Output: Tracks $\mathcal{T}$ of the video

Initialization: $\mathcal{T} \leftarrow \emptyset$
for *frame* f_k *in* `V` **do**
    `/* Figure 2(a) */`
    `/* predict detection boxes & scores */`
    $\mathcal{D}_k \leftarrow \texttt{Det}(f_k)$
    $\mathcal{D}_{high} \leftarrow \emptyset$
    $\mathcal{D}_{low} \leftarrow \emptyset$
    for d *in* $\mathcal{D}_k$ **do**
        if $d.score > \tau$ **then**
            $\mathcal{D}_{high} \leftarrow \mathcal{D}_{high} \cup \{d\}$
        end
        else
            $\mathcal{D}_{low} \leftarrow \mathcal{D}_{low} \cup \{d\}$
        end
    end

    `/* predict new locations of tracks */`
    for t *in* $\mathcal{T}$ **do**
        $t \leftarrow \texttt{KalmanFilter}(t)$
    end

    `/* Figure 2(b) */`
    `/* first association */`
    Associate $\mathcal{T}$ and $\mathcal{D}_{high}$ using `Similarity#1`
    $\mathcal{D}_{remain} \leftarrow$ remaining object boxes from $\mathcal{D}_{high}$
    $\mathcal{T}_{remain} \leftarrow$ remaining tracks from $\mathcal{T}$

    `/* Figure 2(c) */`
    `/* second association */`
    Associate $\mathcal{T}_{remain}$ and $\mathcal{D}_{low}$ using `similarity#2`
    $\mathcal{T}_{re-remain} \leftarrow$ remaining tracks from $\mathcal{T}_{remain}$

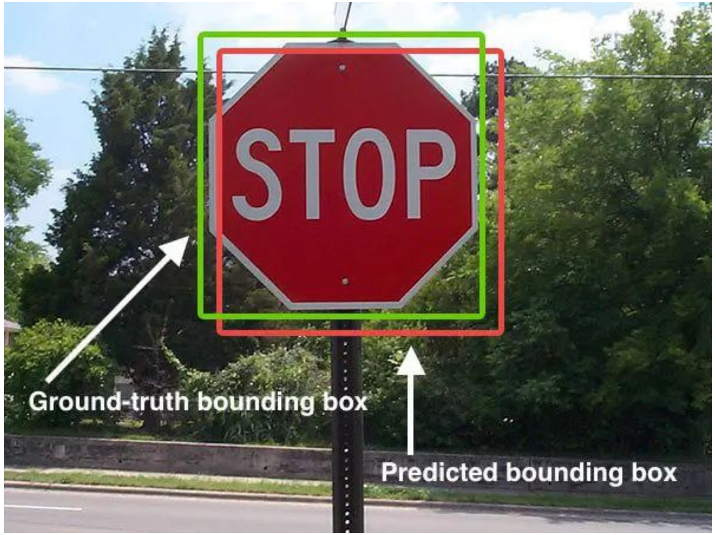
    `/* delete unmatched tracks */`
    $\mathcal{T} \leftarrow \mathcal{T} \setminus \mathcal{T}_{re-remain}$

    `/* initialize new tracks */`
    for d *in* $\mathcal{D}_{remain}$ **do**
        $\mathcal{T} \leftarrow \mathcal{T} \cup \{d\}$
    end
end
Return: $\mathcal{T}$

Track rebirth [70, 89] is not shown in the algorithm for simplicity. In green is the key of our method.

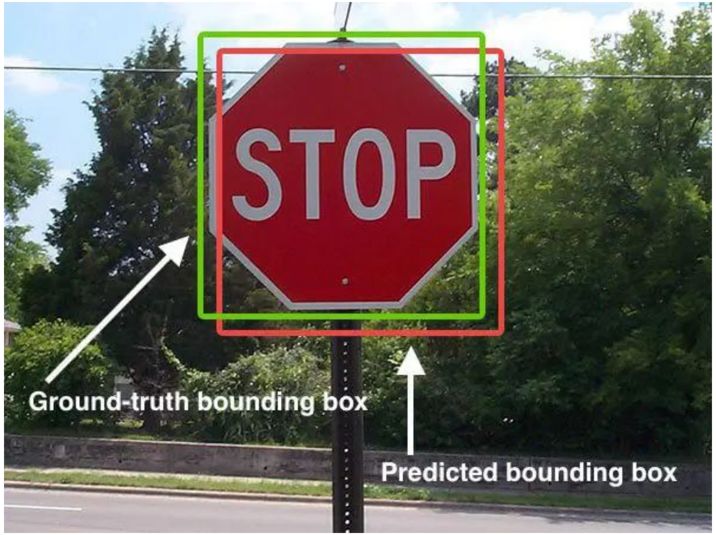
Similarity Between Tracklets and Detection Boxes

- New tracklet positions estimated with Kalman Filter
- First: Similarity with high scoring boxes
  - IoU Matching: Intersection over Union
    - Measures overlap of predicted tracklets and the detection boxes
  - Re-ID: neural network for re-identifying people from an image
  - Hungarian Algorithm
    - Algorithm for finding optimal matches based on IoU or Re-ID
    - $O(n^3)$



Similarity Between Tracklets and Detection Boxes

- Second: Similarity with low scoring boxes
  - Typically associated with motion blur or occlusion, so Re-ID doesn't work well
  - IoU alone is used for Hungarian Algorithm



The Full Tracker

- Each frame is given to YOLOX, which returns detection boxes
- BYTE uses these detection boxes and existing tracklets to determine which detection boxes are accurate enough to keep
- Accurate detection boxes are then used to update or create tracklets

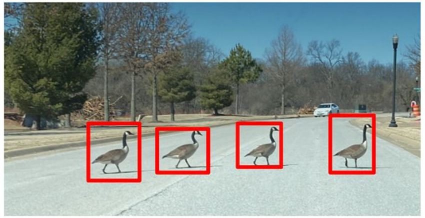
Experiments - Datasets

- Training
  - MOT17
  - MOT18
  - CrowdHuman
  - Cityperson
  - ETHZ
- Testing
  - MOT17
  - HiEve
  - BDD100K

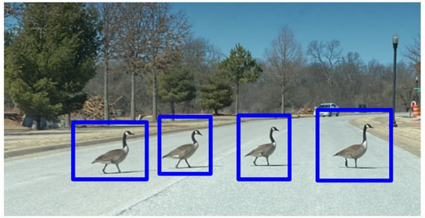
- All datasets similar
  - Crowded places
  - Humans are objects of interest
  - Mix of indoor and outdoor
  - BDD100K includes more than just humans as objects of interest

Experiments - Metrics

- MOTA (Multiple Object Tracking Accuracy)
  - Measures overall accuracy considering:
    - Missed detections
    - False positives
    - ID mismatches
- IDF1 (Identification F1 Score)
  - Measures how well a tracker maintains identity of objects over time
    - True positives
    - False positives
    - False negatives
      - Missed objects
- HOTA (Higher Order Tracking Accuracy)
  - Balances:
    - Detection accuracy (finding objects)
    - Association accuracy (tracking objects)



Ground Truth



Model Prediction

Experiments - Hardware and Details

- Training
  - Done on 8 NVIDIA Tesla V100 GPUs
  - Batch size: 48
  - ~12 hours training time
  - SGD (Stochastic Gradient Descent) optimization algorithm
- Evaluation
  - Single NVIDIA Tesla V100 GPU
  - FPS measured with FP16-precision
    - 16-bit floating point
- Image size: 1440x800

NVIDIA Tesla V100

Experiments - Similarity Metrics

- Comparison of different similarity metrics in BYTE algorithm
  - Similarity#1 used for high-scoring detection boxes
  - Similarity#2 used for low-scoring detection boxes

		MOT17			BDD100K		
Similarity#1	Similarity#2	MOTA↑	IDF1↑	IDs↓	mMOTA↑	mIDF1↑	IDs↓
IoU	Re-ID	75.8	77.5	231	39.2	48.3	29172
IoU	IoU	**76.6**	79.3	**159**	39.4	48.9	27902
Re-ID	Re-ID	75.2	78.7	276	45.0	53.4	10425
Re-ID	IoU	76.3	**80.5**	216	**45.5**	**54.8**	**9140**

Tracking Accuracy

Identity Preservation

ID Switches

BDD100K

BDD100K

BDD100K

Again:
- IoU = Intersection over Union, between existing tracklet and detection box
- Re-ID = Compares appearance of detected object to tracklet's previous frames

Experiments - Other Approaches

- Comparison of performance to other approaches

Method	w/ Re-ID	MOT17 MOTA↑	MOT17 IDF1↑	MOT17 IDs↓	BDD100K mMOTA↑	BDD100K mIDF1↑	BDD100K IDs↓	FPS
SORT		74.6	76.9	291	30.9	41.3	10067	**30.1**
DeepSORT	✓	75.4	77.2	239	24.5	38.2	10720	13.5
MOTDT	✓	75.8	77.6	273	26.7	39.8	14520	11.1
BYTE (ours)		**76.6**	79.3	**159**	39.4	48.9	27902	29.6
BYTE (ours)	✓	76.3	**80.5**	216	**45.5**	**54.8**	**9140**	11.8

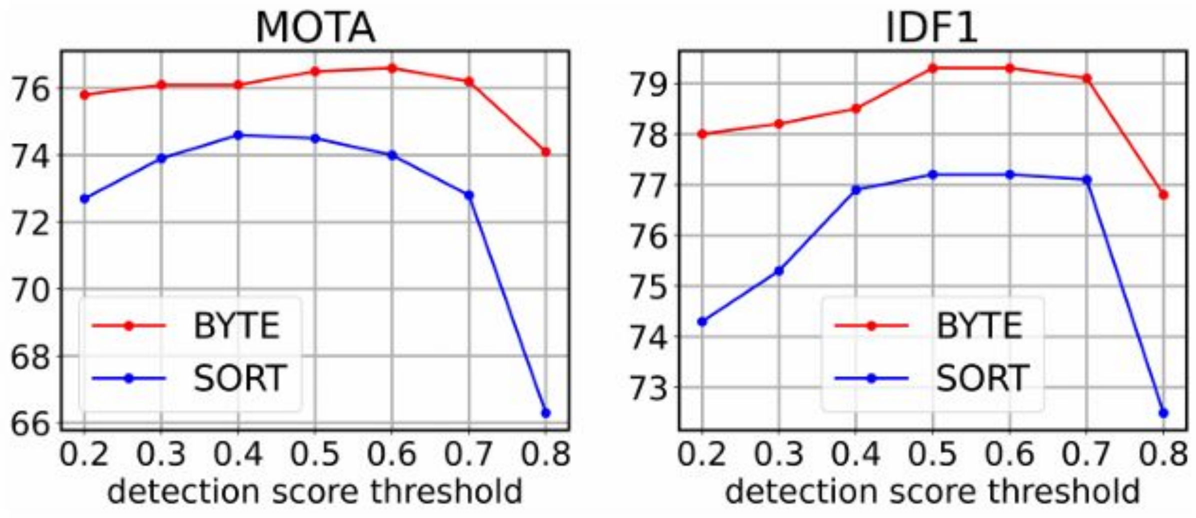
Uses re-identification

Sort: No deep learning, prone to identity switches
DeepSORT: Uses deep learning, appearance cues
MOTDT: Uses appearance and motion cues

Re-identification: continue identifying an object even if it temporarily disappears across frames

Experiments - Detection Score Threshold

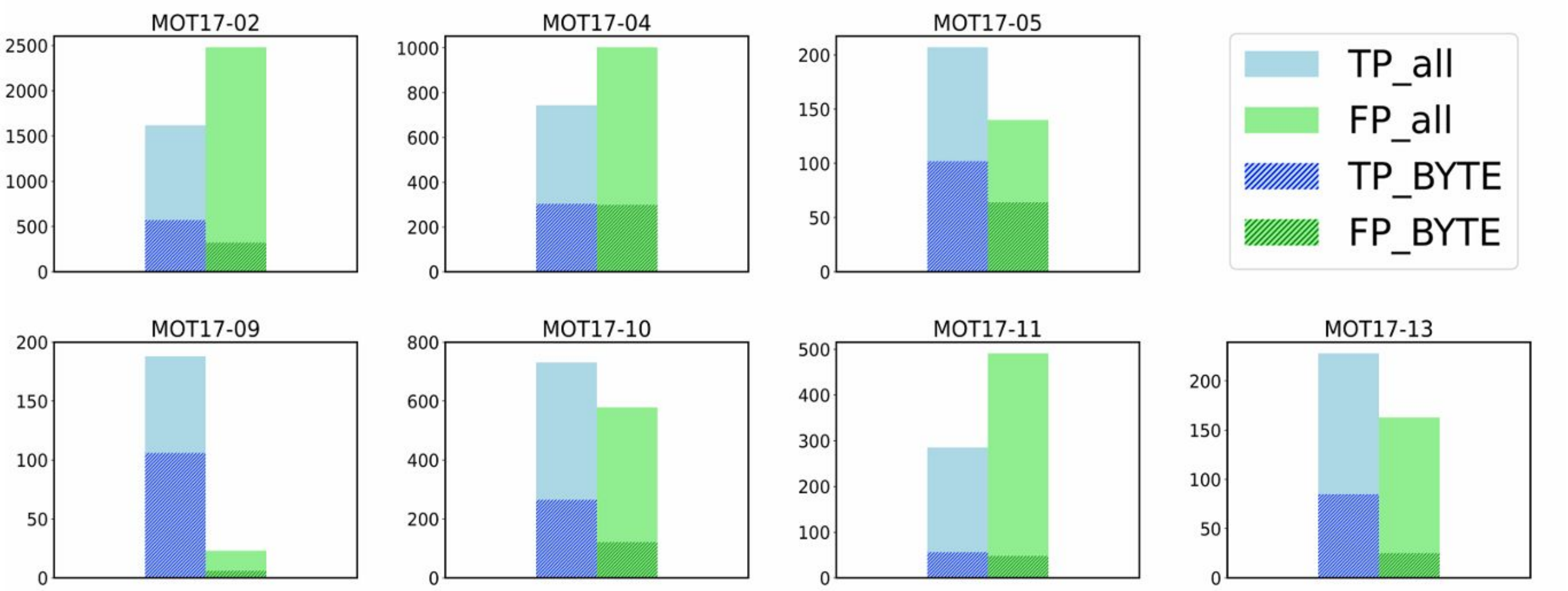
- Comparison of performance with different detection score thresholds
    - Shows BYTE is more robust and consistent



"Detection score threshold, τhigh, is a sensitive hyper-parameter and needs to be carefully tuned in the task of multi-object tracking"

Experiments - Analysis On Low Score Detection Boxes

- TP_all (True Positive) and FP_all (False Positive) cover all low-score detection boxes
- TP_BYTE and FP_BYTE cover those that don't get eliminated in BYTE



Experiments - Benchmarks (MOT17)

- Best in all accuracy metrics
- Best in framerate (fastest)
- Not the best in false positives
  - Likely due to considering more of the low-scoring detections (which traditional methods reject)

Tracker	MOTA↑	IDF1↑	HOTA↑	FP↓	FN↓	IDs↓	FPS↑
DAN [61]	52.4	49.5	39.3	25423	234592	8431	<3.9
Tube_TK [46]	63.0	58.6	48.0	27060	177483	4137	3.0
MOTR [80]	65.1	66.4	-	45486	149307	2049	-
CTracker [48]	66.6	57.4	49.0	22284	160491	5529	6.8
CenterTrack [89]	67.8	64.7	52.2	**18498**	160332	3039	17.5
QuasiDense [47]	68.7	66.3	53.9	26589	146643	3378	20.3
TraDes [71]	69.1	63.9	52.7	20892	150060	3555	17.5
MAT [25]	69.5	63.1	53.8	30660	138741	2844	9.0
SOTMOT [87]	71.0	71.9	-	39537	118983	5184	16.0
TransCenter [75]	73.2	62.2	54.5	23112	123738	4614	1.0
GSDT [67]	73.2	66.5	55.2	26397	120666	3891	4.9
Semi-TCL [32]	73.3	73.2	59.8	22944	124980	2790	-
FairMOT [85]	73.7	72.3	59.3	27507	117477	3303	25.9
RelationTrack [78]	73.8	74.7	61.0	27999	118623	**1374**	8.5
PermaTrackPr [63]	73.8	68.9	55.5	28998	115104	3699	11.9
CSTrack [33]	74.9	72.6	59.3	23847	114303	3567	15.8
TransTrack [59]	75.2	63.5	54.1	50157	86442	3603	10.0
FUFET [54]	76.2	68.0	57.9	32796	98475	3237	6.8
SiamMOT [34]	76.3	72.3	-	-	-	-	12.8
CorrTracker [65]	76.5	73.6	60.7	29808	99510	3369	15.6
TransMOT [15]	76.7	75.1	61.7	36231	93150	2346	9.6
ReMOT [76]	77.0	72.0	59.7	33204	93612	2853	1.8
ByteTrack (ours)	**80.3**	**77.3**	**63.1**	25491	**83721**	2196	**29.6**

Experiments - Benchmarks (MOT20)

- Similar performance to MOT17
- MOT20 presents much more crowded areas
    - More opportunities for occlusion
    - Average number of pedestrians in an image is 170

Tracker	MOTA↑	IDF1↑	HOTA↑	FP↓	FN↓	IDs↓	FPS↑
MLT [83]	48.9	54.6	43.2	45660	216803	2187	3.7
FairMOT [85]	61.8	67.3	54.6	103440	88901	5243	13.2
TransCenter [75]	61.9	50.4	-	45895	146347	4653	1.0
TransTrack [59]	65.0	59.4	48.5	27197	150197	3608	7.2
CorrTracker [65]	65.2	69.1	-	79429	95855	5183	8.5
Semi-TCL [32]	65.2	70.1	55.3	61209	114709	4139	-
CSTrack [33]	66.6	68.6	54.0	**25404**	144358	3196	4.5
GSDT [67]	67.1	67.5	53.6	31913	135409	3131	0.9
SiamMOT [34]	67.1	69.1	-	-	-	-	4.3
RelationTrack [78]	67.2	70.5	56.5	61134	104597	4243	2.7
SOTMOT [87]	68.6	71.4	-	57064	101154	4209	8.5
ByteTrack (ours)	**77.8**	**75.2**	**61.3**	26249	**87594**	**1223**	**17.5**

Experiments - Benchmarks (HiEve = Human in Events)

- More complex events and more diverse cameras than MOT17 and 20
- Similar performance

Tracker	MOTA↑	IDF1↑	MT↑	ML↓	FP↓	FN↓	IDs↓
DeepSORT [70]	27.1	28.6	8.5%	41.5%	5894	42668	2220
MOTDT [12]	26.1	32.9	8.7%	54.6%	6318	43577	1599
IOUtracker [7]	38.6	38.6	28.3%	27.6%	9640	28993	4153
JDE [69]	33.1	36.0	15.1%	24.1%	9526	33327	3747
FairMOT [85]	35.0	46.7	16.3%	44.2%	6523	37750	**995**
CenterTrack [89]	40.9	45.1	10.8%	32.2%	3208	36414	1568
ByteTrack (Ours)	**61.7**	**63.1**	**38.3%**	**21.6%**	**2822**	**22852**	1031

MT = Mostly Tracked, higher MT score indicates that the tracker successfully follows objects for most of their existence
ML = Mostly Lost, lower ML score means fewer objects are lost early, which indicates better tracking performance

Experiments - Benchmarks (BDD100K)

- Driving Video Dataset (Autonomous Vehicles)
- Multiclass object tracking
- Similar performance
- Worse for IDF1 than ODTrack

Tracker	split	mMOTA↑	mIDF1↑	MOTA↑	IDF1↑	FN↓	FP↓	IDs↓	MT↑	ML↓
Yu *et al.* [79]	val	25.9	44.5	56.9	66.8	122406	52372	8315	8396	3795
QDTrack [47]	val	36.6	50.8	63.5	**71.5**	108614	46621	**6262**	9481	3034
ByteTrack(Ours)	val	**45.5**	**54.8**	**69.1**	70.4	**92805**	**34998**	9140	**9626**	**3005**
Yu *et al.* [79]	test	26.3	44.7	58.3	68.2	213220	100230	14674	16299	6017
DeepBlueAI	test	31.6	38.7	56.9	56.0	292063	**35401**	25186	10296	12266
madamada	test	33.6	43.0	59.8	55.7	209339	76612	42901	16774	**5004**
QDTrack [47]	test	35.5	52.3	64.3	**72.3**	201041	80054	**10790**	17353	5167
ByteTrack(Ours)	test	**40.1**	**55.8**	**69.6**	71.3	**169073**	63869	15466	**18057**	5107

Strengths

- Handles occlusion and motion blur very well
    - Through inclusion of lower confidence components
    - Common in video tracking
- Reduces false negatives substantially
    - Matching low confidence with existing tracklets
- Highly accurate
    - Outperforms previous methods on examined benchmarks
- Simple and fast
    - Can be integrated into existing pipelines without major overhead

Weaknesses

- More false positives
    - Double-edged sword of including lower confidence detections
    - Noisy and occluded areas inherently tricky, especially in extreme conditions
- More prone to ID switching / mismatching
    - Incorrect associations in complex or cluttered scenes
- Not as strong in multi-class tracking scenarios
    - ByteTrack primarily designed for single-class tracking, where all objects belong to one class (tracking pedestrians, cars, etc.)

Suggestions for Improvement

- Look into different similarity metrics for Similarity#1 and Similarity#2 in the BYTE algorithm
  - Current metrics (IoU, Re-ID) could be suboptimal for types of objects being tracked
- Experiment with other detectors
  - ByteTrack performance heavily dependent on detection algorithm
  - This paper used YOLOX
- Explore more optimizations in multi-class scenarios
  - One example: class-specific tracking algorithms

Our Project

- NVIDIA Xavier NX
  - Xavier: Computing platform designed for AI, robotics, embedded systems
    - Combines ARM-based CPU and NVIDIA GPU
  - Compare Symmetric CPU and GPU
    - CPU → Sequential processing
    - GPU → Parallelize detection and tracking tasks
- Measurements for varying input sizes
  - Framerate
    - FPS on CPU and GPU
  - Power usage
  - MOTA, IDF1, HOTA
    - Accuracy, Consistency, Accuracy and Consistency over time