

# Traffic Analysis in Original Video Data of Ayalon Road

Ido Greenberg

2019



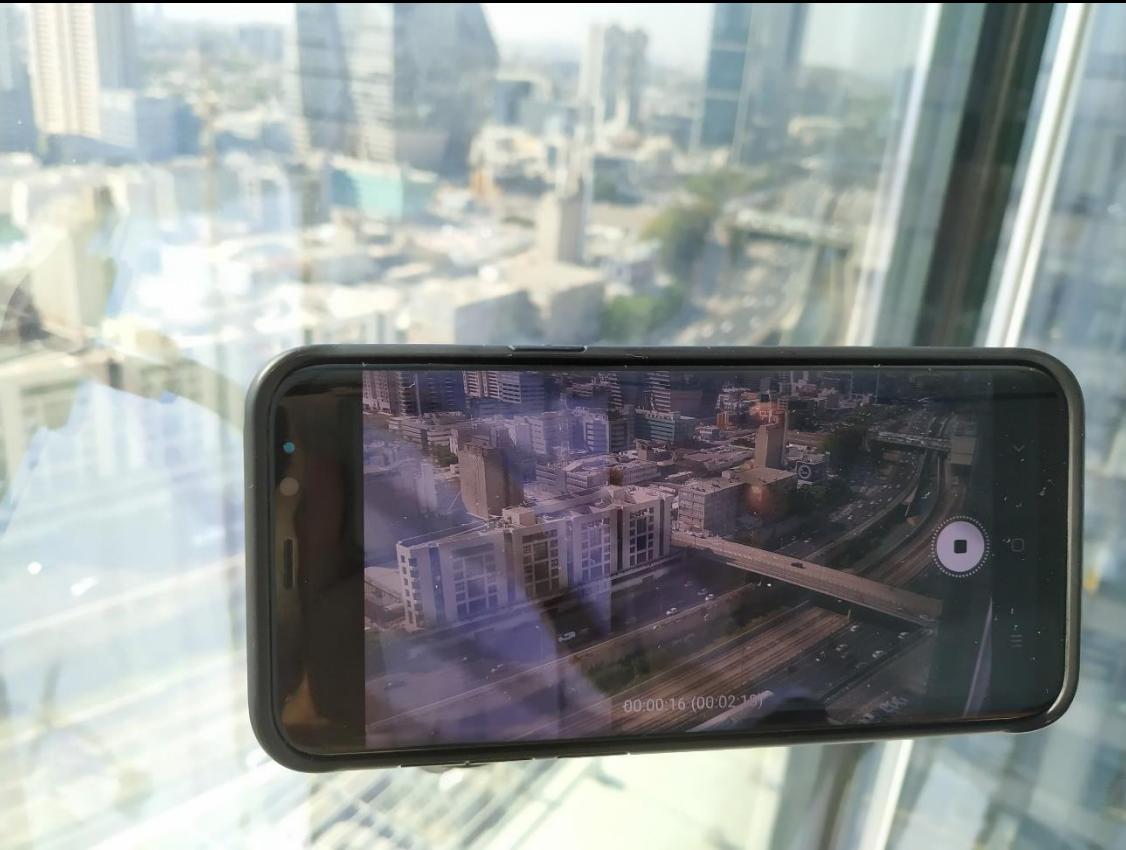
Background



# Looking down: data are everywhere

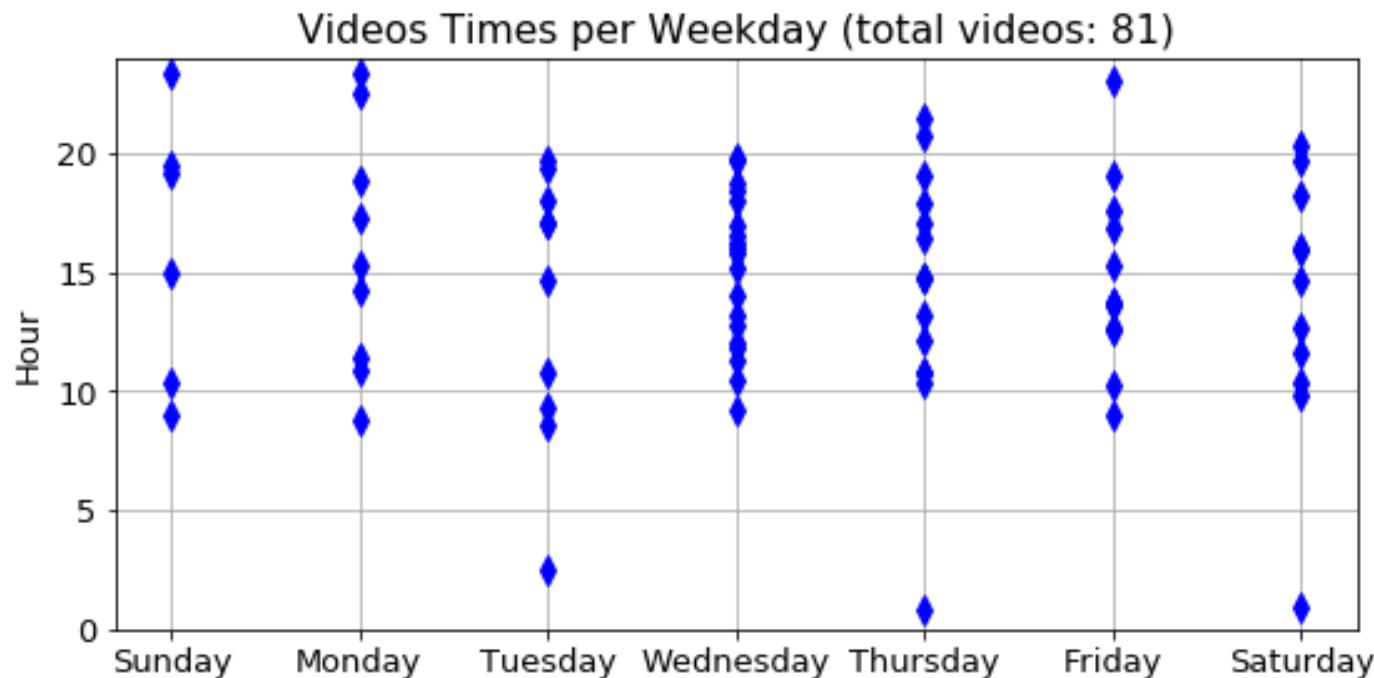
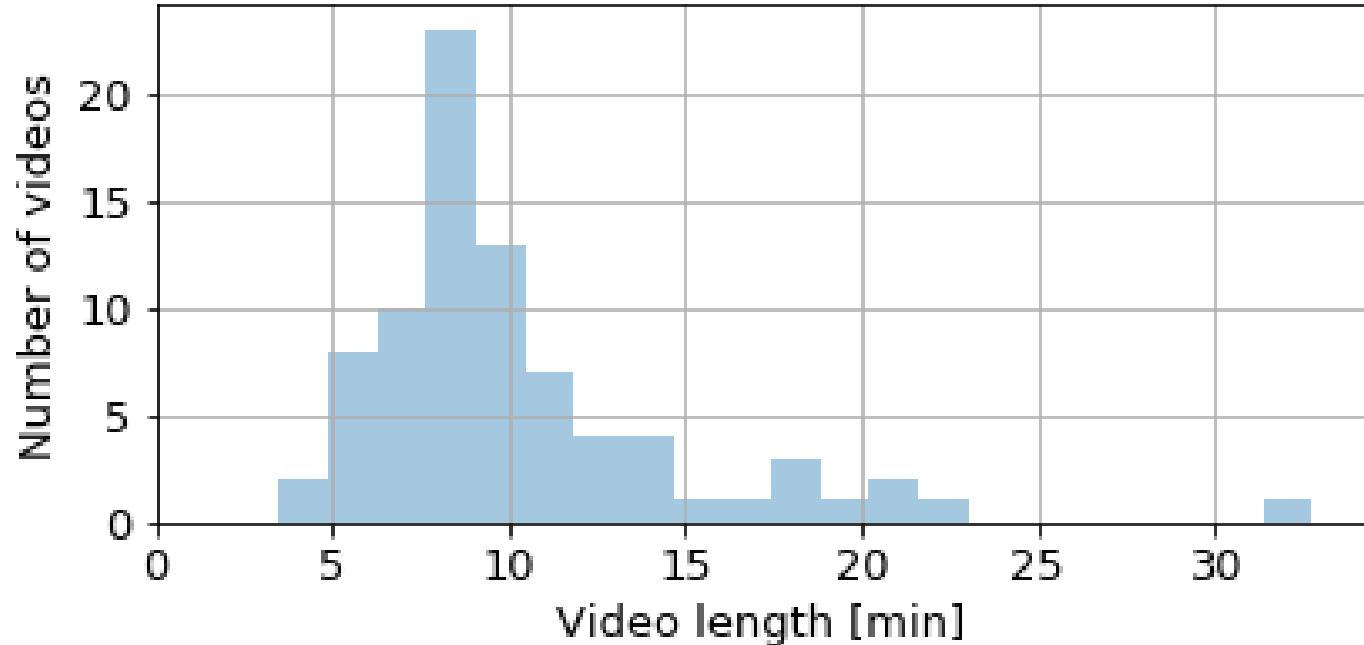


- Hyperlapse, x8 speed, Full-HD
- 120MB for 8 minutes



# Collected data

- 81 videos X 8 minutes over 2 months



# What we need

- **Detection:** find vehicles in a frame
- **Tracking:** connect the same vehicle in different frames
- **Analysis of traffic**



Detection

# Out-of-the-box YOLO





Out-of-the-box SSD (better with small, crowded objects)

car: 55.08%

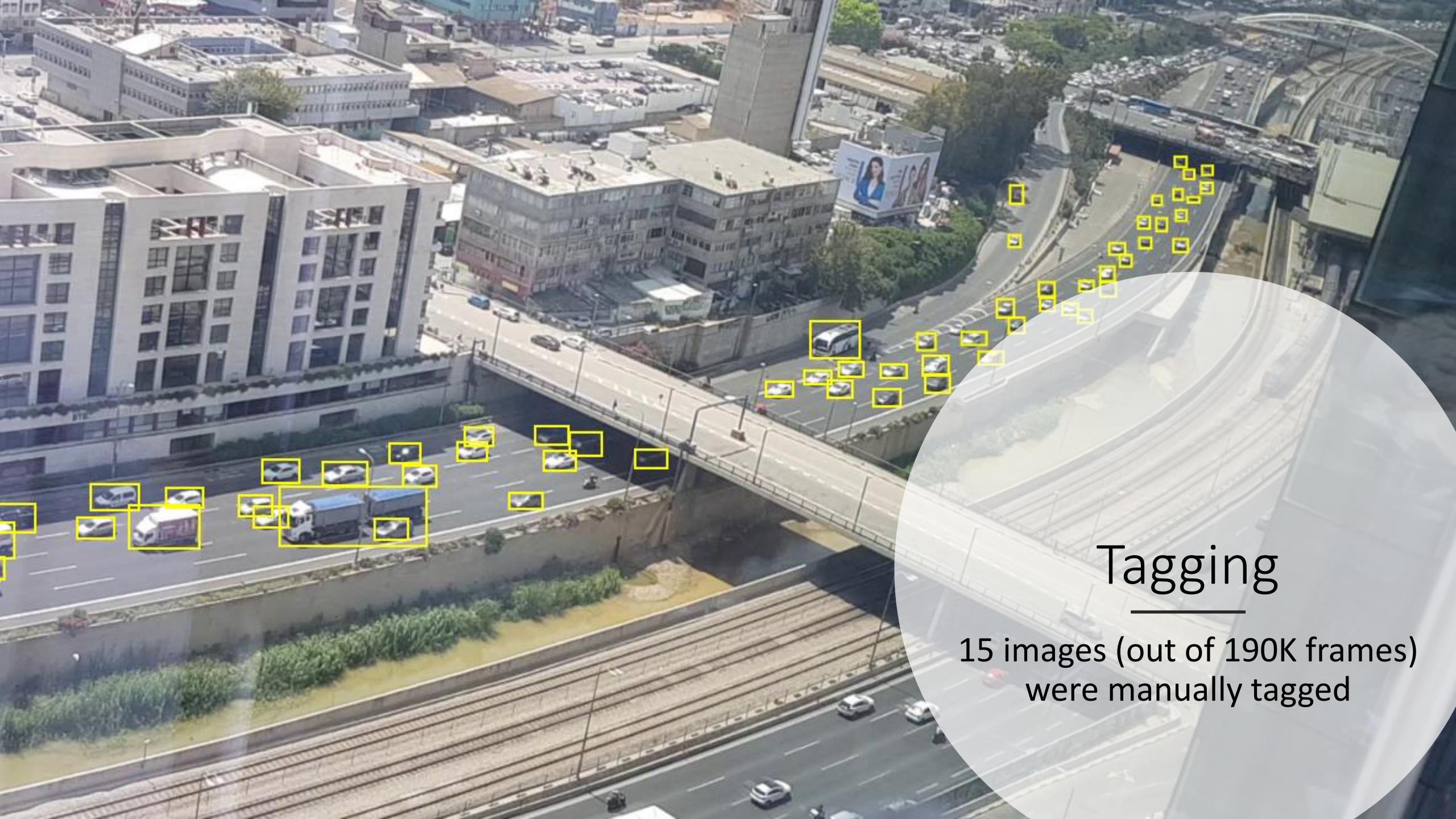
Zoom in?

# No one's gonna train for me

- **Train:** tune the detection according to the data
- **Design:** adjust models to fit typical sizes & density
  - Also simplify – no need for multiple object classes



Having to train by myself - illustration

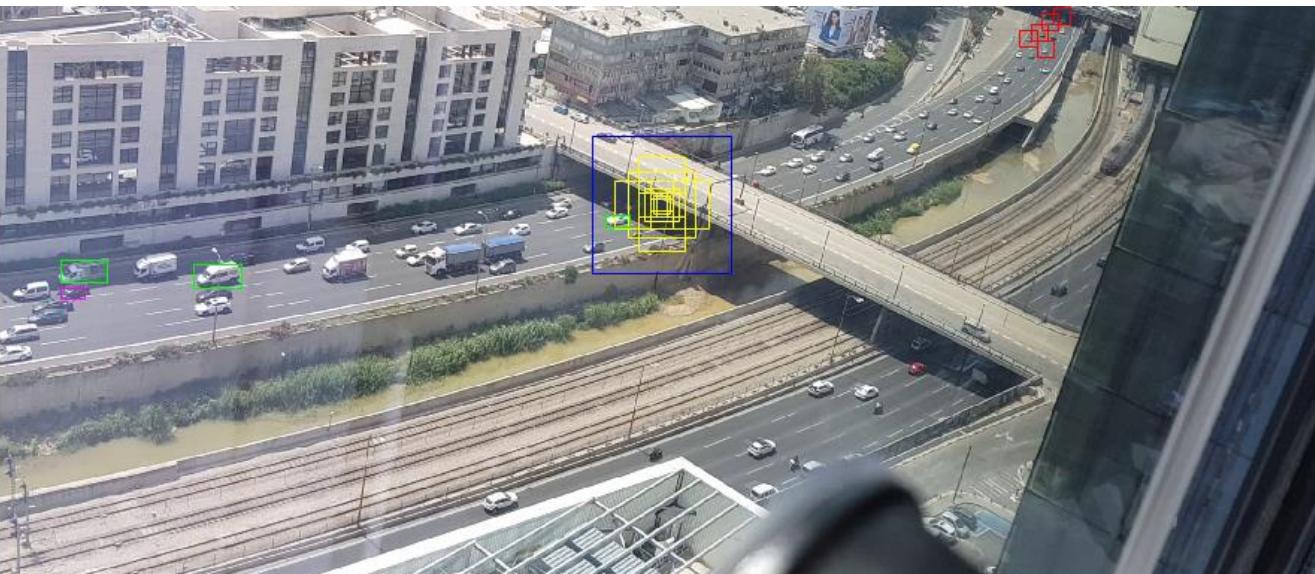


## Tagging

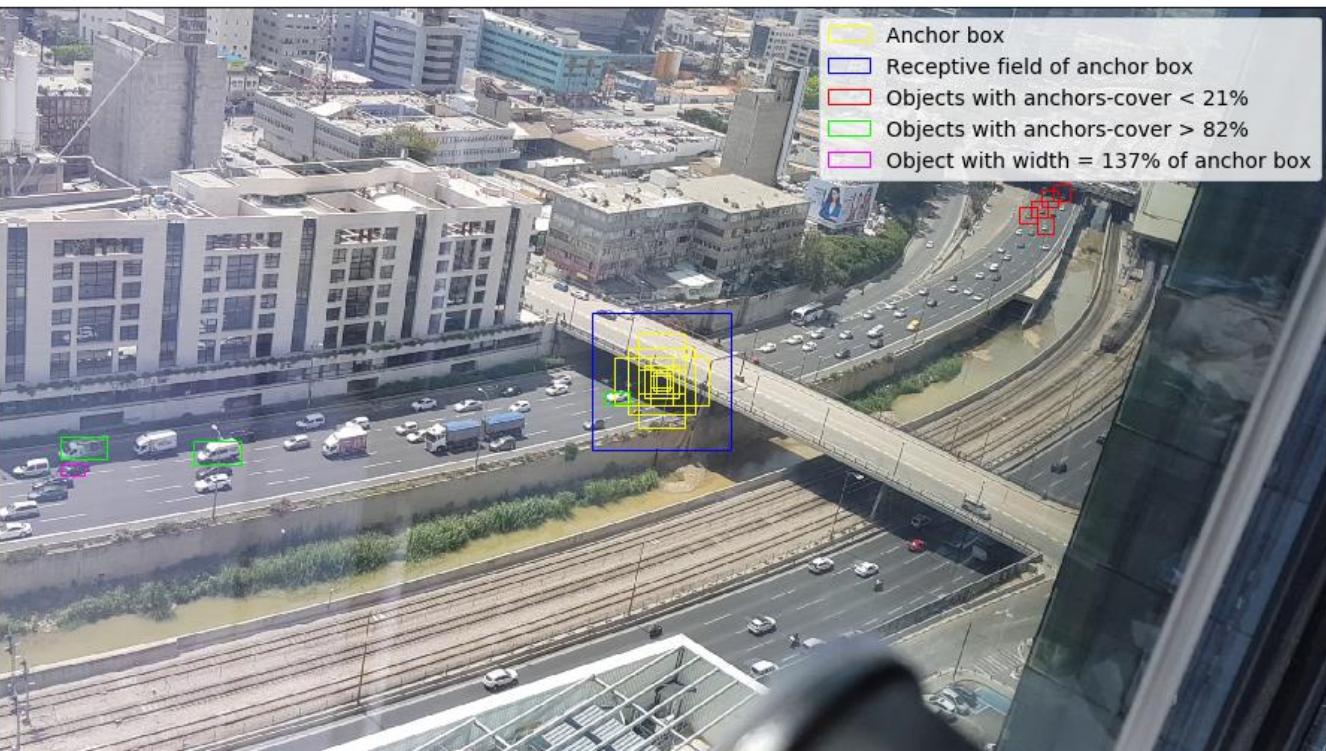
15 images (out of 190K frames)  
were manually tagged

# Tagged vehicles → network-compatible output

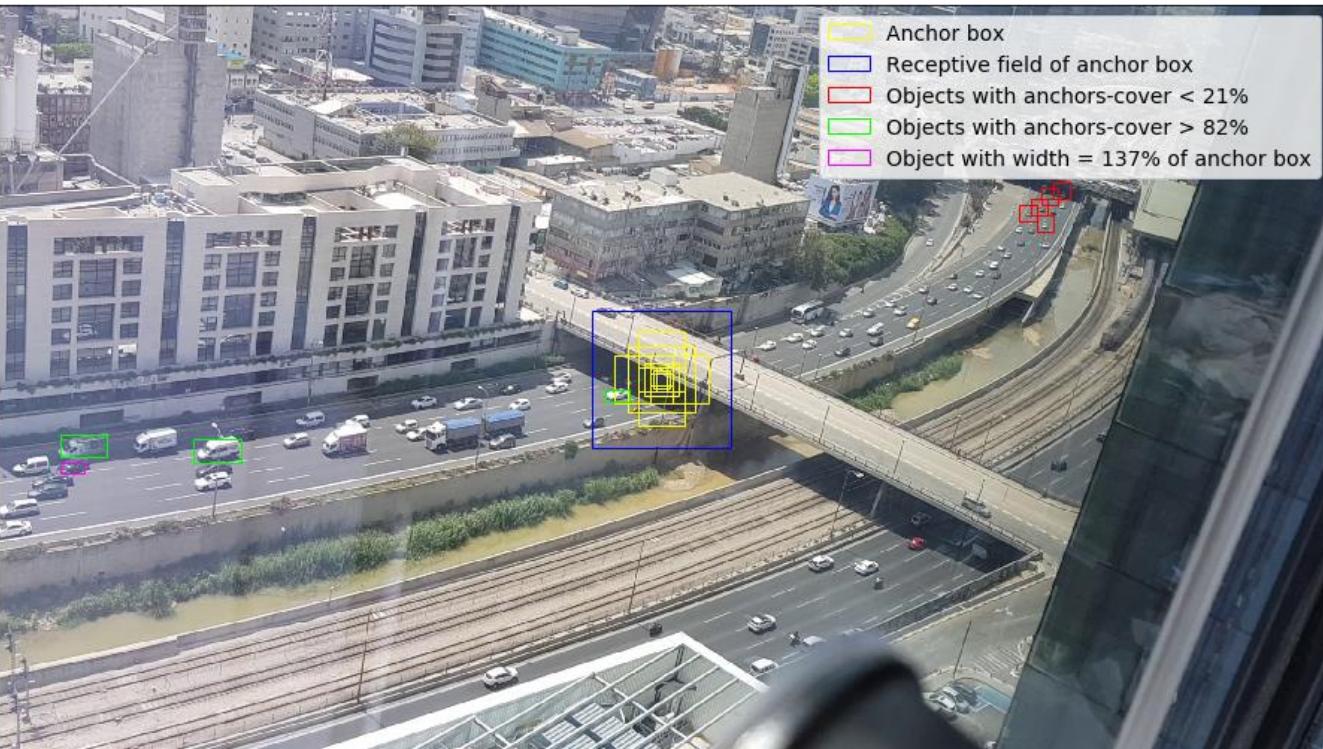
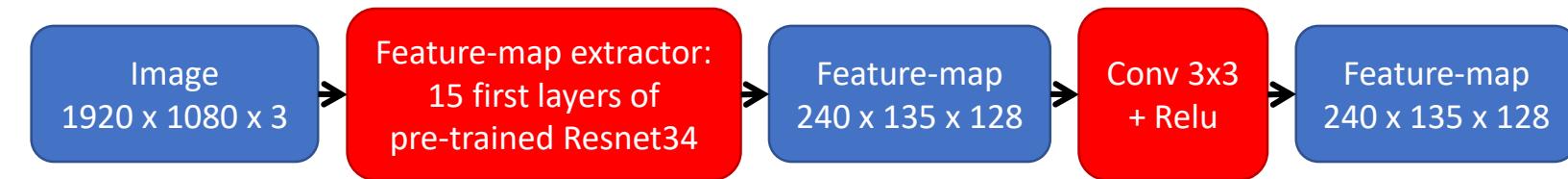
- Split image to 290K (overlapping) rectangles (***anchor boxes***)
  - 9 boxes per location – various scales & shapes
  - A box receives info from pixels around it (***receptive field***)
- Output per box:
  - Highly-overlapping object → (1, object location)
  - Little/no overlap → (0, ...arbitrary...)
  - Somewhere in the middle → drop from training data



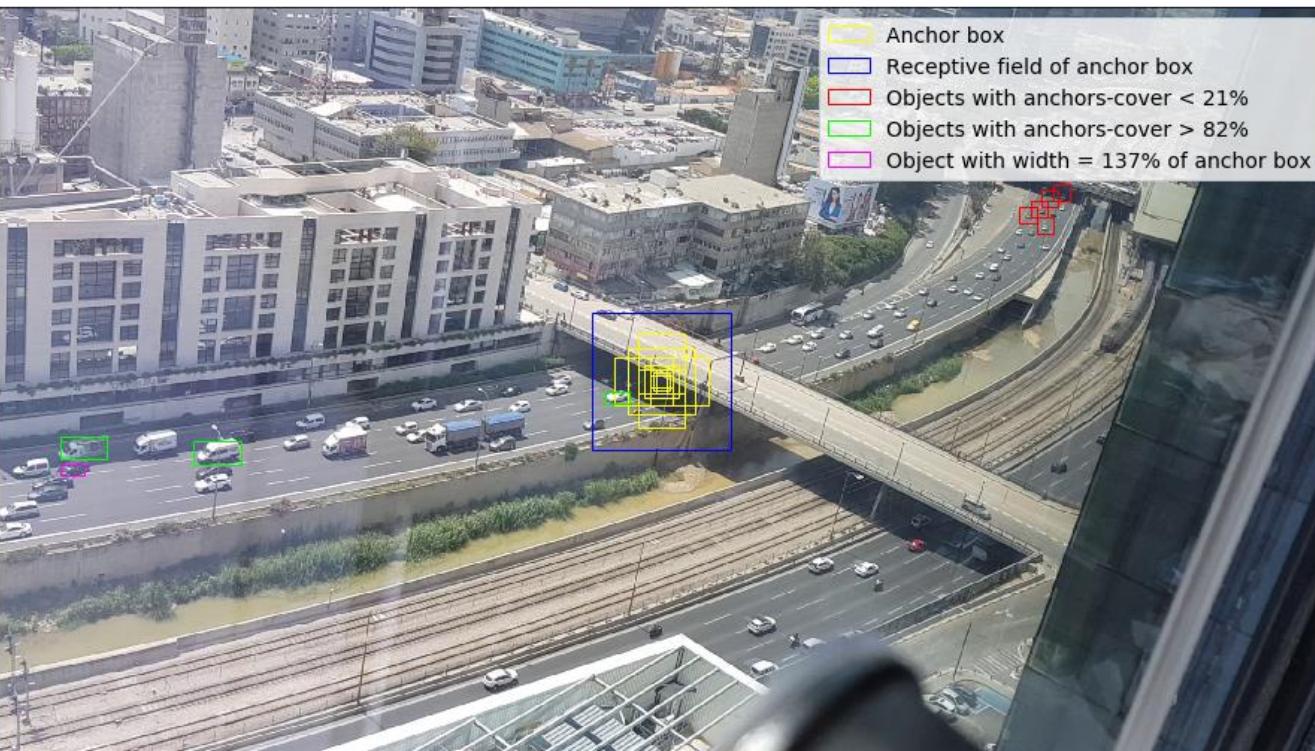
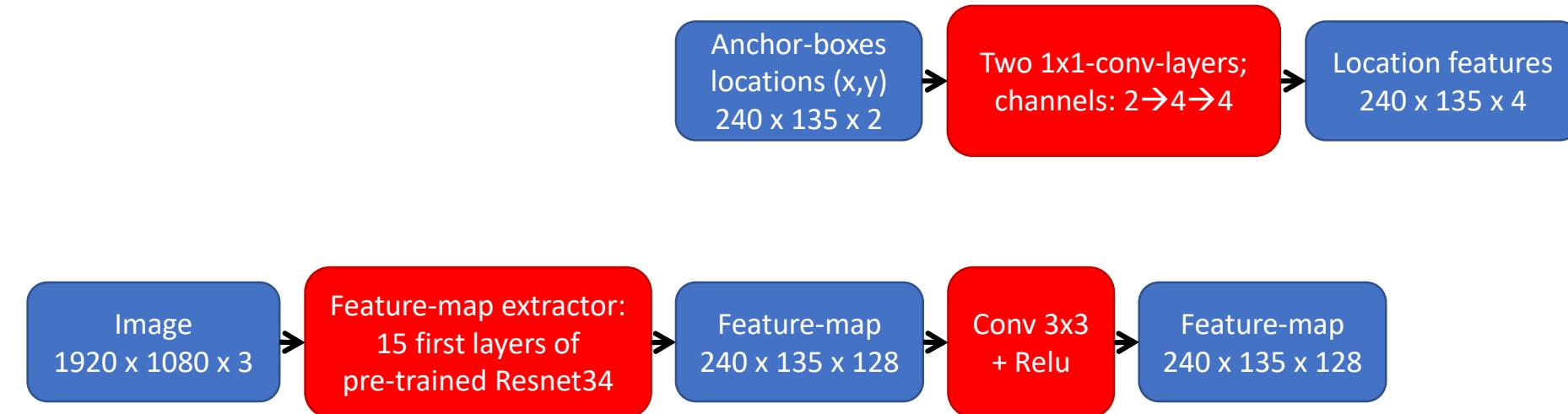
# Network architecture



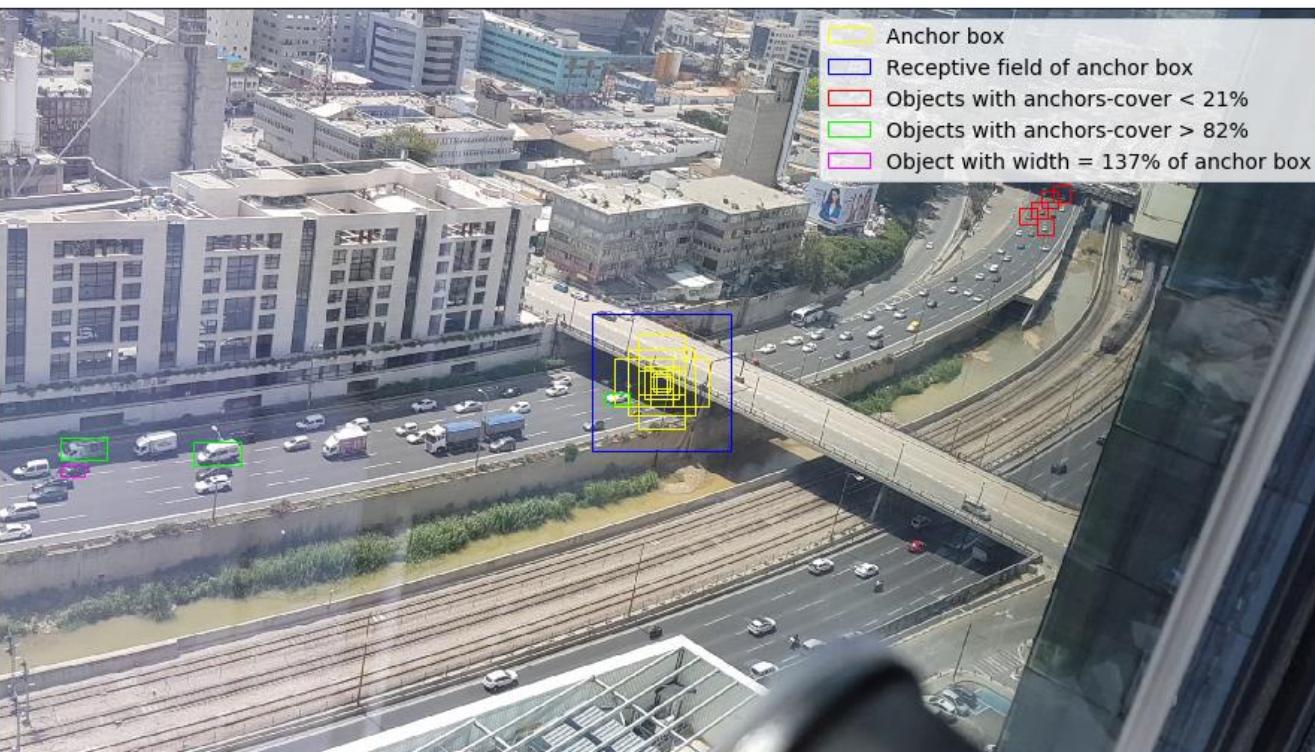
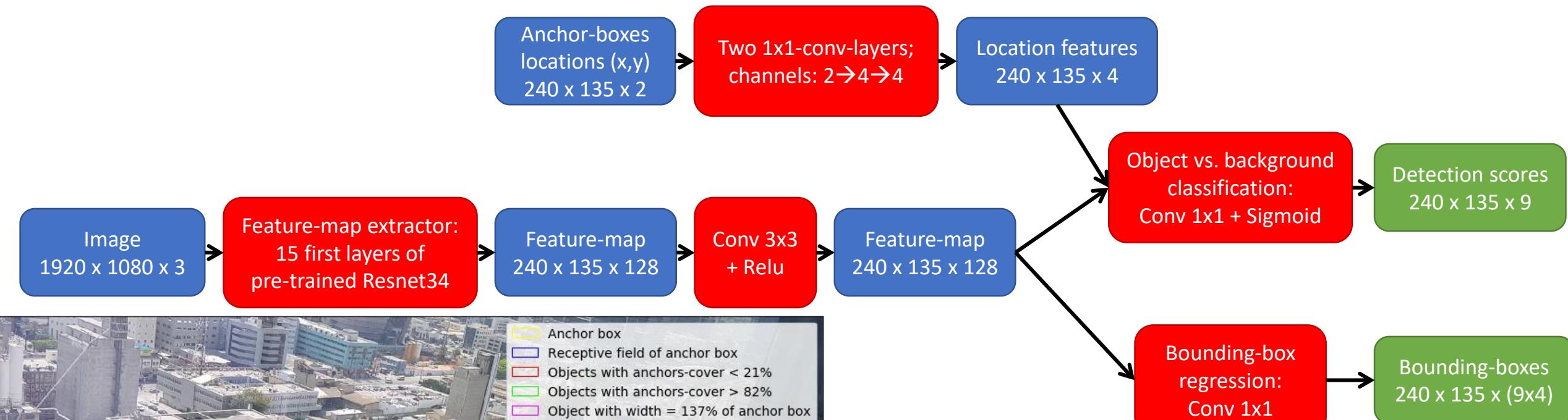
# Network architecture



# Network architecture

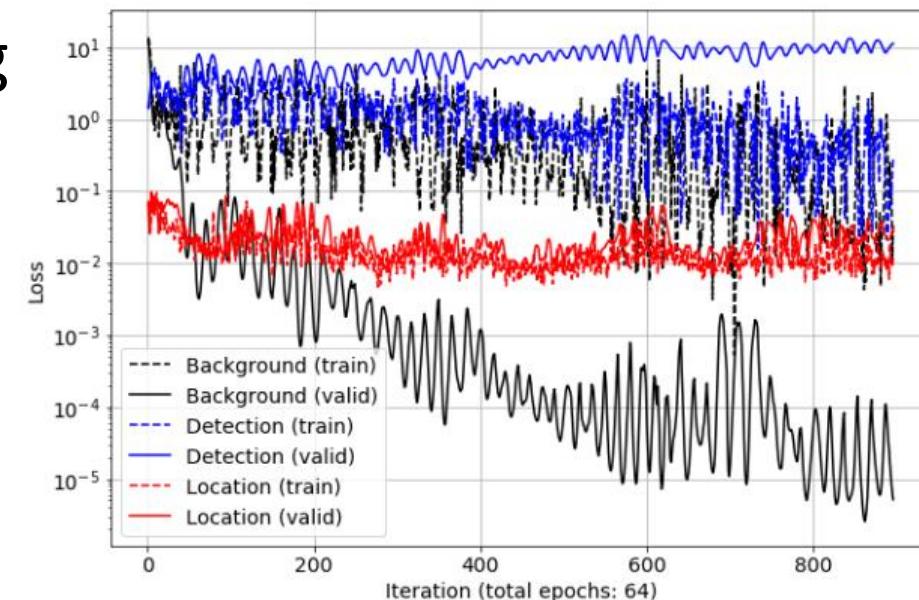


# Network architecture



# Training

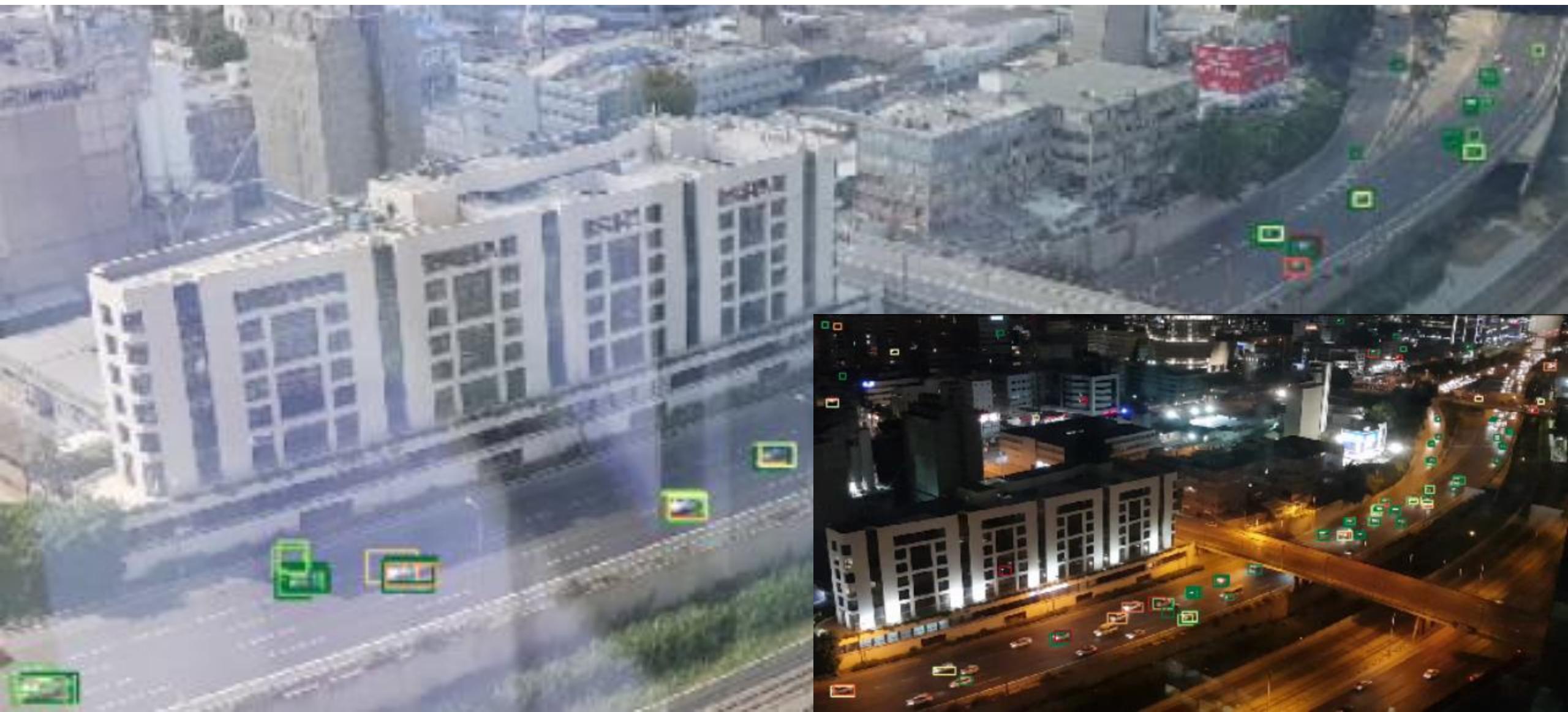
- **Optimizer:** Adam
- **Transfer learning:** begin with freezed pre-trained Resnet layers
- **Loss:** Binary cross-entropy (detection) + L1 (location)
  - Manual tuning of the FP/FN tradeoff
- **Hard negative mining:** up-sample boxes with large loss
- Architecture & hyper-params empirical tuning
- Running time on my laptop:
  - Freezed Resnet-layers: 5 minutes
  - Unfreezed Resnet-layers: +7 minutes



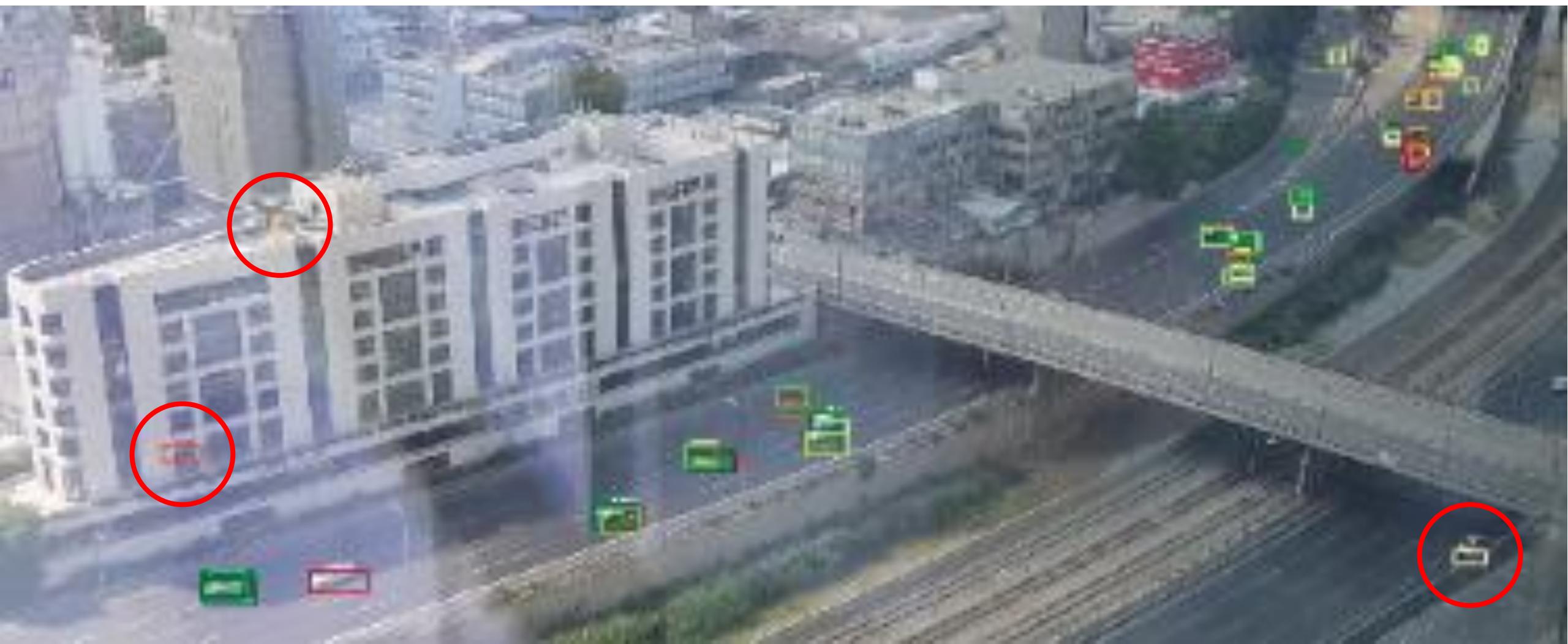
# *Without Hard Negative Mining*



# 3 training images



Middle of training (15 training images)



Middle of training

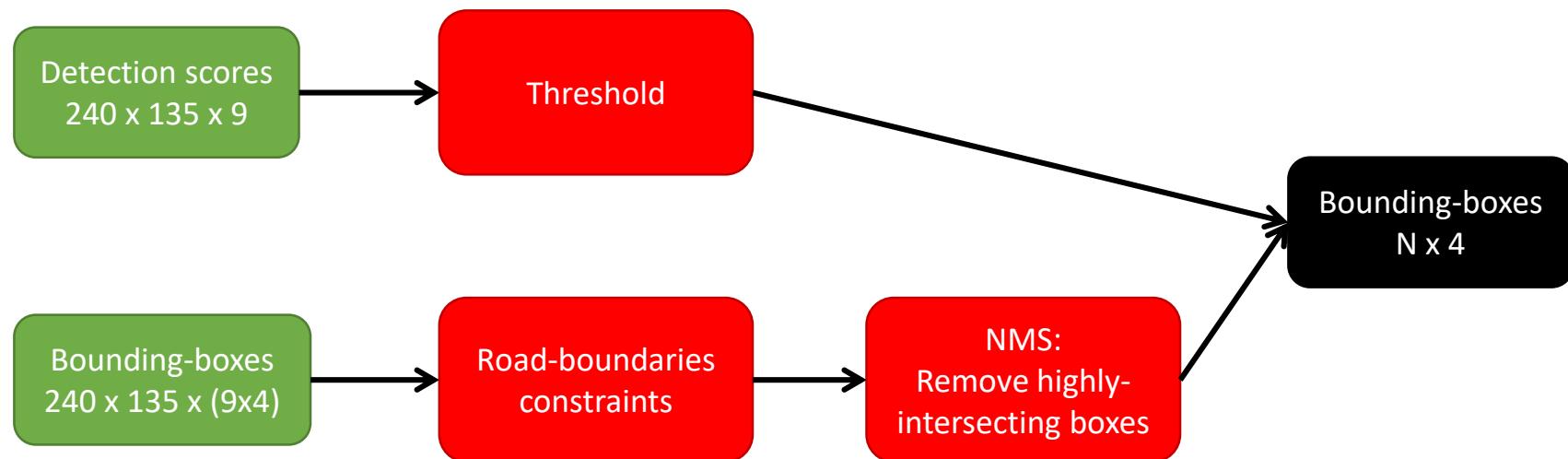


End of training

(15 training images)



# Network output → predicted boxes



122 objects (0.315 = red <= certainty <= green)

Road  
boundaries

\*All results are out-of-sample

42 objects (0.360 = red <= certainty <= green)

Night +  
reflections

\*All results are out-of-sample



22 objects (0.379 = red <= certainty <= green)

Shavuot

\*All results are out-of-sample



Tracking

# SORT: Simple, Online and Realtime Tracking

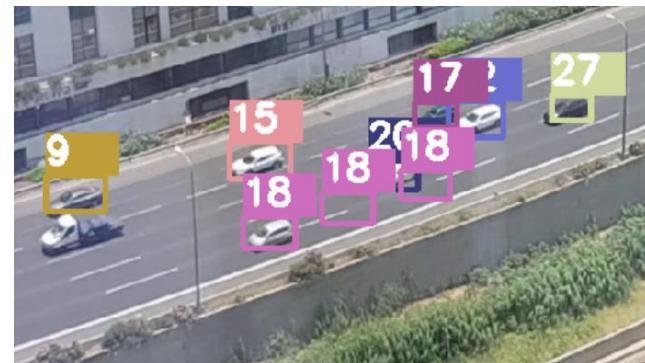
- Active track – represented by ***Kalman Filter***:
  - Iteratively merges info from motion extrapolation & new observations
  - At any time, holds the track location and its uncertainty

# SORT: Simple, Online and Realtime Tracking

- Active track – represented by ***Kalman Filter***:
  - Iteratively merges info from motion extrapolation & new observations
  - At any time, holds the track location and its uncertainty
  - [Detections  $\leftrightarrow$  tracks] assignment: bounding boxes intersection (**IOU**)
    - KF's location uncertainty is ignored
    - Fast vehicles + low frame-rate  $\rightarrow$  no intersection



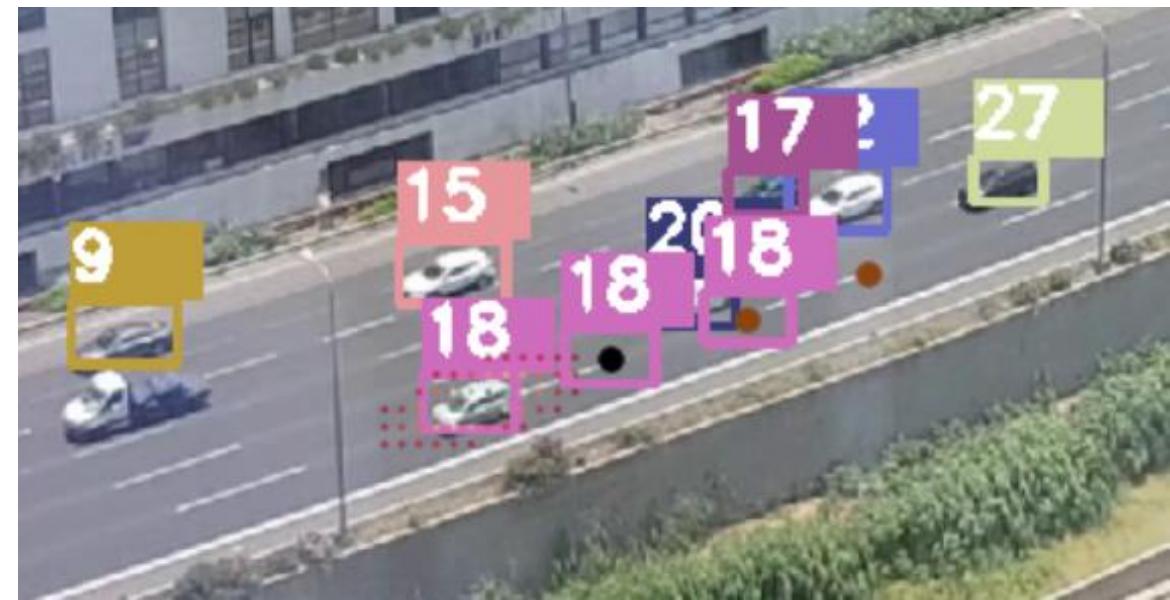
What the engineer had in mind



What the customer had in mind

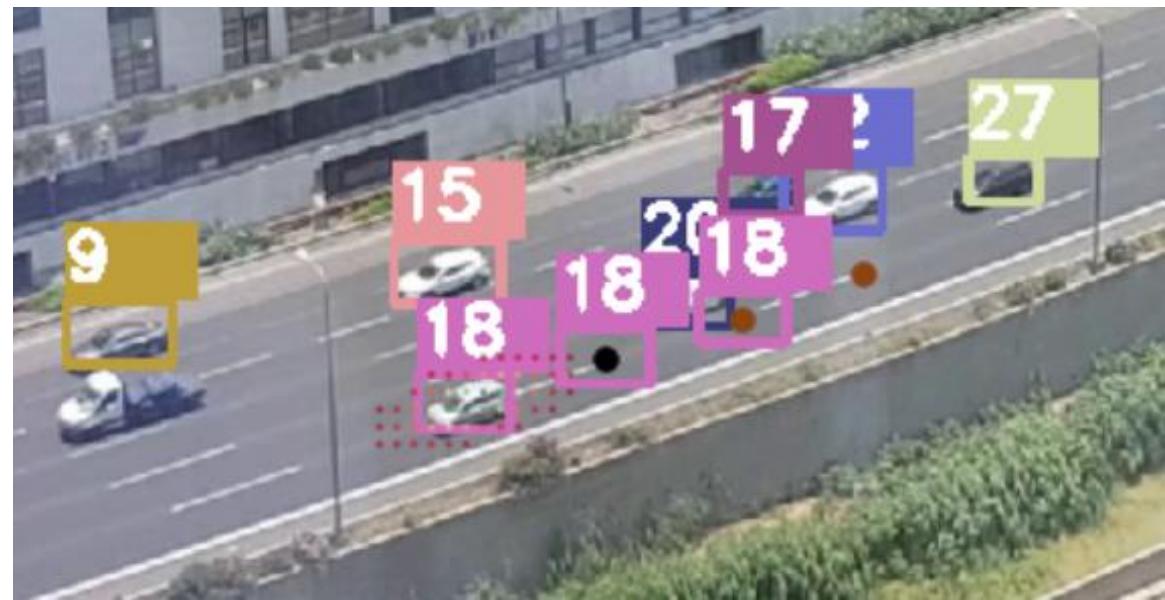
# KF-based objects assignment

- Predicted location = 2D Gaussian defined by KF's location & uncertainty
- Likelihood(new detection corresponds to track) =  $P(\text{detection's location})$



# KF-based objects assignment

- **Predicted location = 2D Gaussian** defined by KF's location & uncertainty
- **Likelihood(new detection corresponds to track)** =  $P(\text{detection's location})$
- Predicted-location error: 10 pixels in motion direction ( $u_1$ ), 2 in perpendicular ( $u_2$ )
  - Prior motion direction: globally-estimated once in a while
  - **Covariance** =  $U^T \cdot \begin{pmatrix} 10^2 & 0 \\ 0 & 2^2 \end{pmatrix} \cdot U$   $(U := (u_1, u_2))$



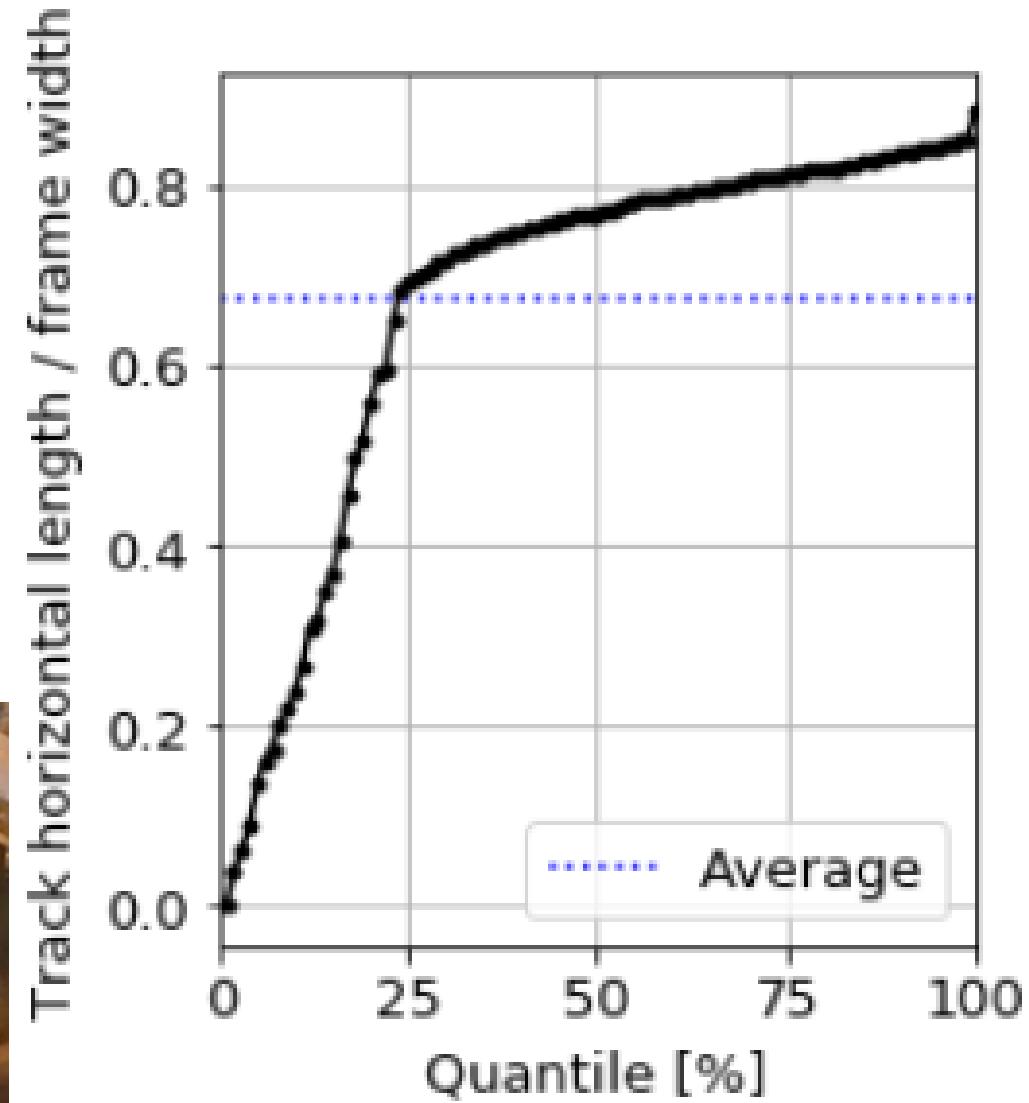
# Does it work?

- Vehicle #4 is successfully tracked in spite of missing detections



# Does it work?

- Do we track the whole path?

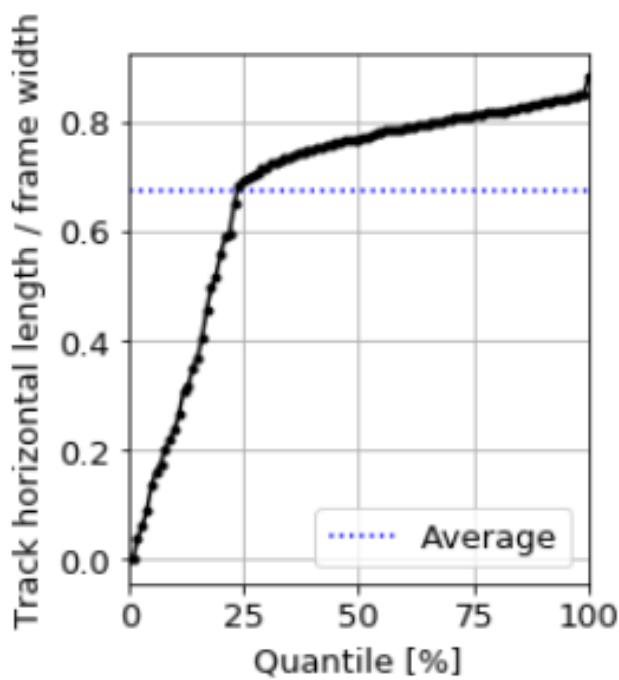
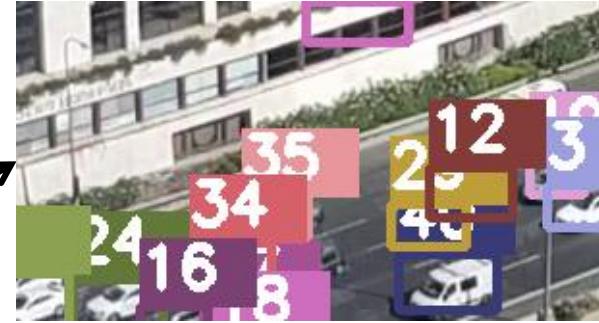


# Does it work?

- Do we track the whole path?

- Fake tracks: false detections

- Failed tracks: missing detections, hidings

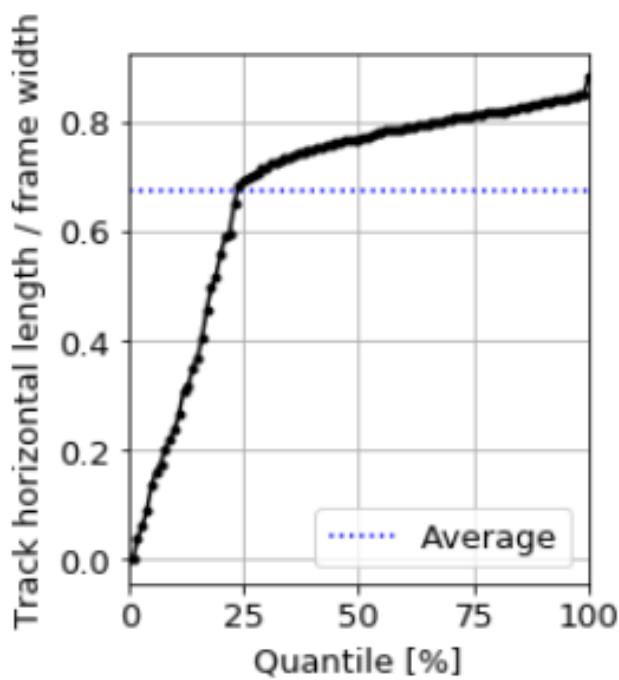
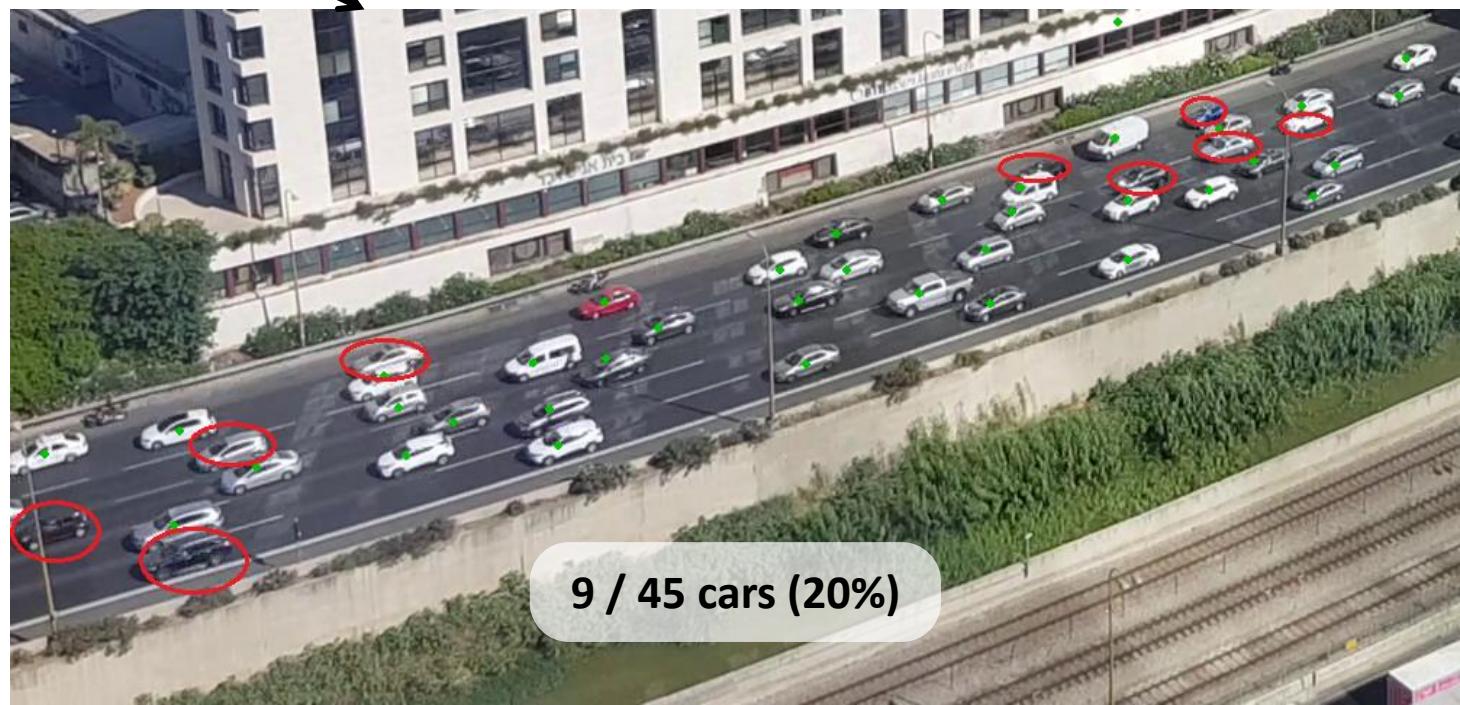


# Does it work?

- Do we track the whole path?

- Fake tracks: false detections

- Failed tracks: missing detections, hidings



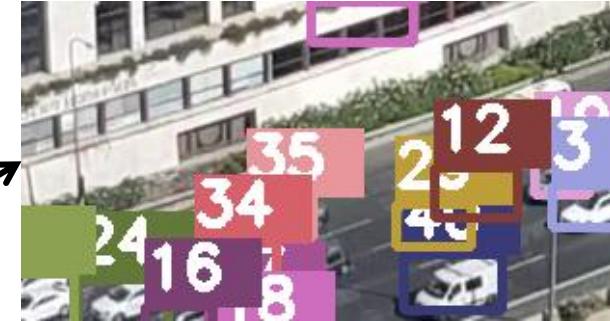
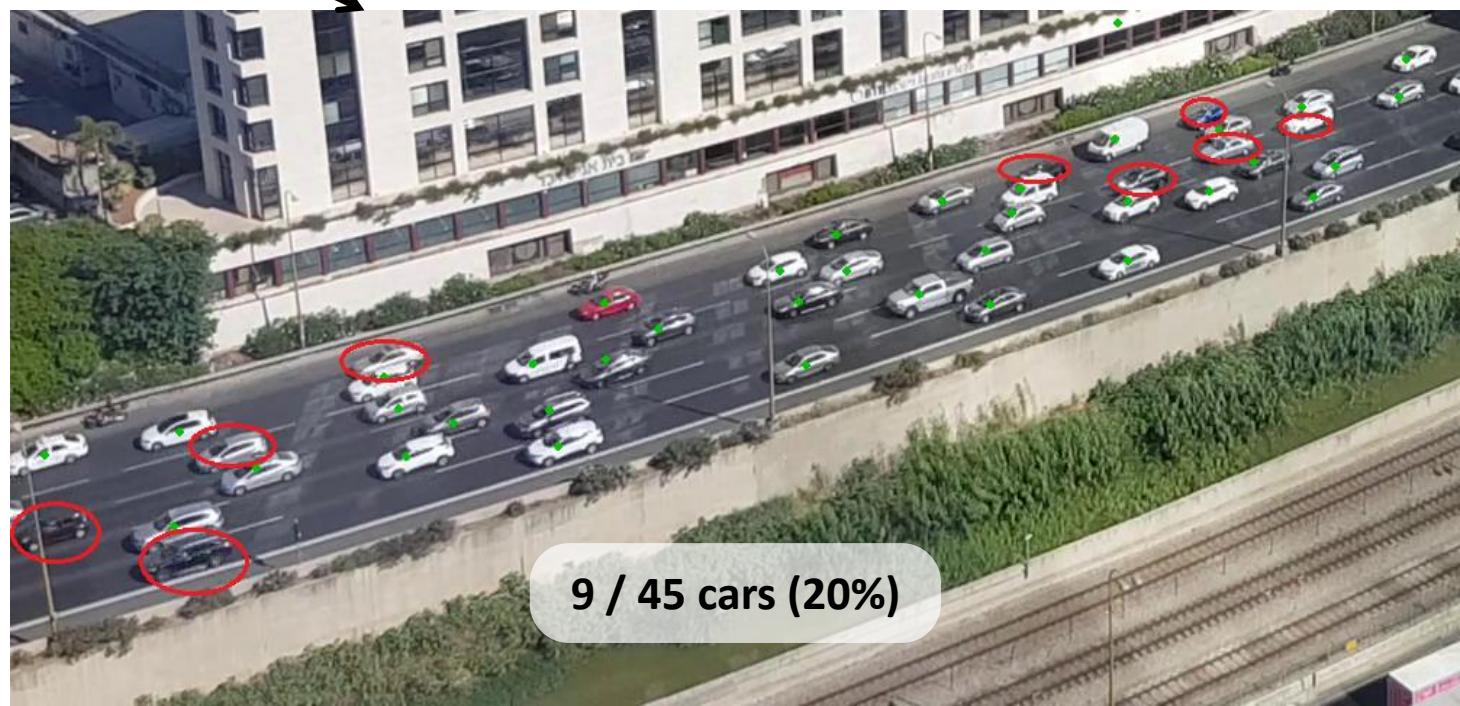
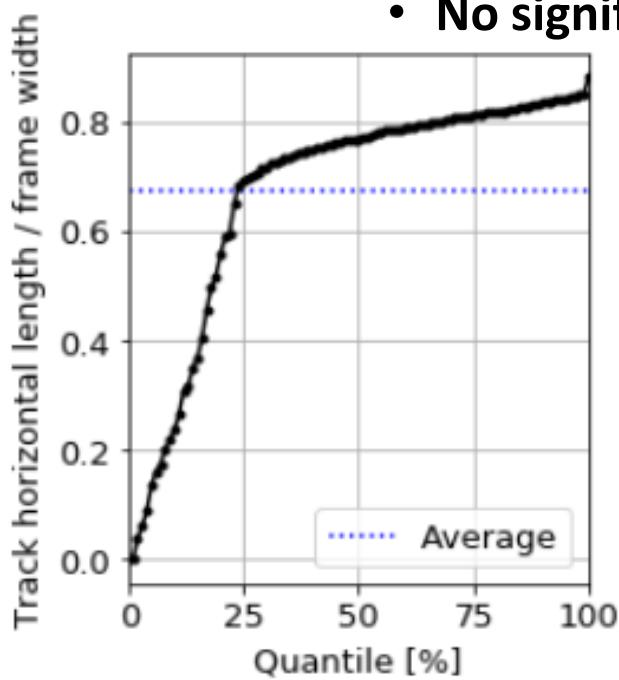
# Does it work?

- Do we track the whole path?

- Fake tracks: false detections
- Failed tracks: missing detections, hidings

- **Eliminated by removing short tracks**

- **No significant bias expected**



# Does it work?

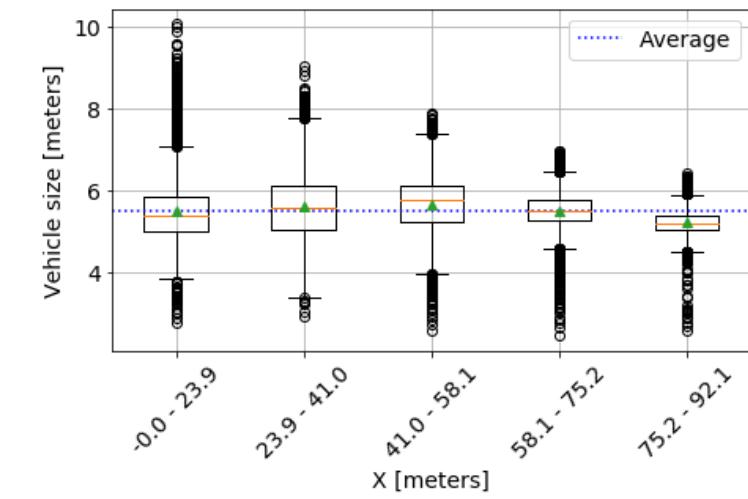
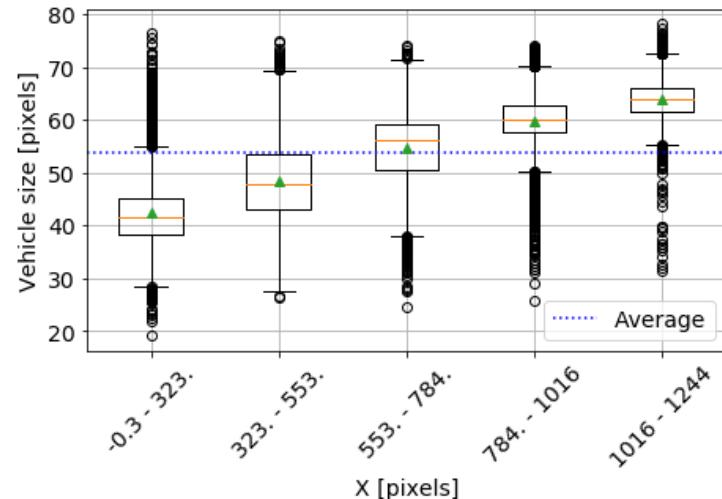
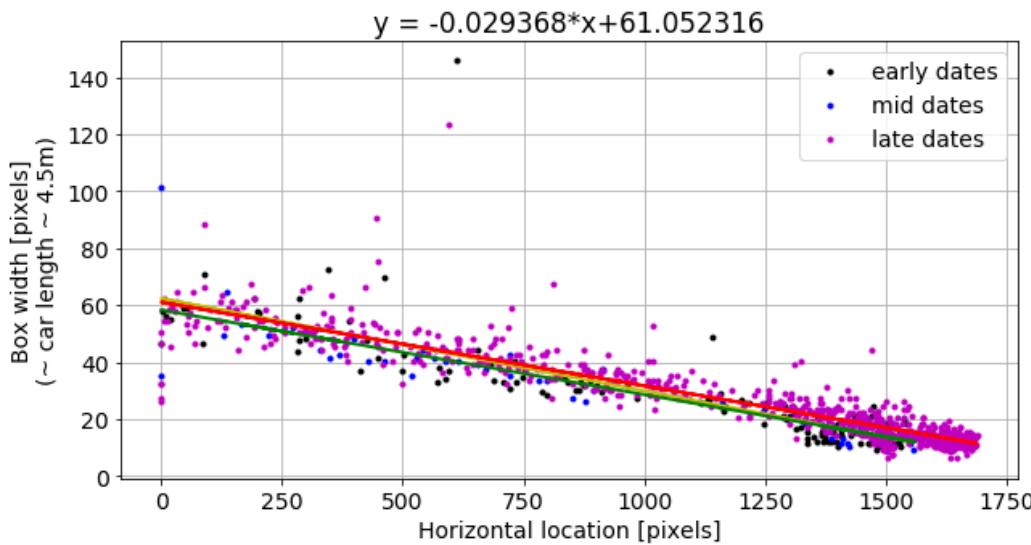
- **Running time** on my laptop:
  - Full frames: 1.2 frame/sec (~25 min/video)
  - **Cropped frames:** 3 frame/sec (~10 min/video)
- **Code profiling:**
  - **Do it** – there're always unexpected pitfalls
    - E.g. data-frames reallocation, GPU memory cleaning
  - **%lprun** in Jupyter Notebook
  - Sync GPU before measuring time



Analysis

# Data representation

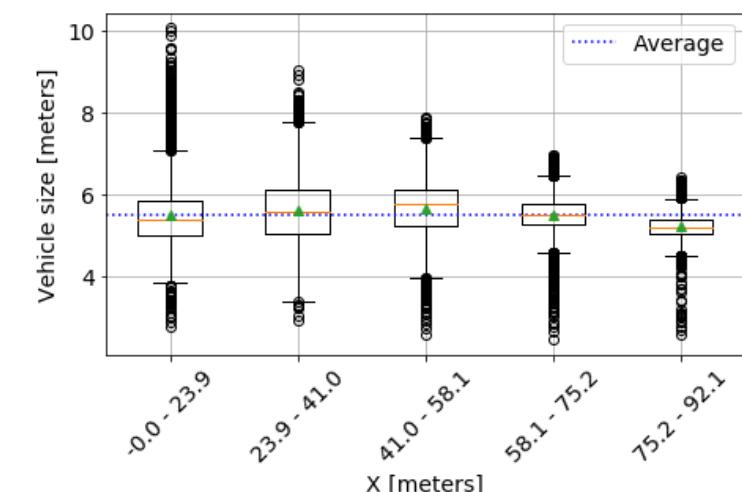
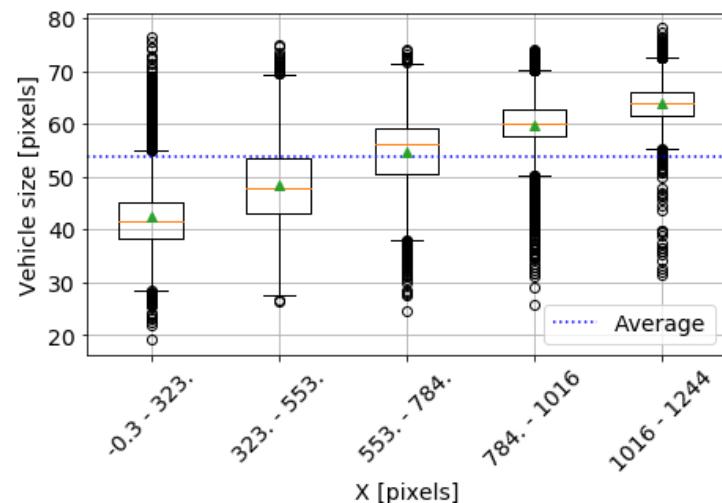
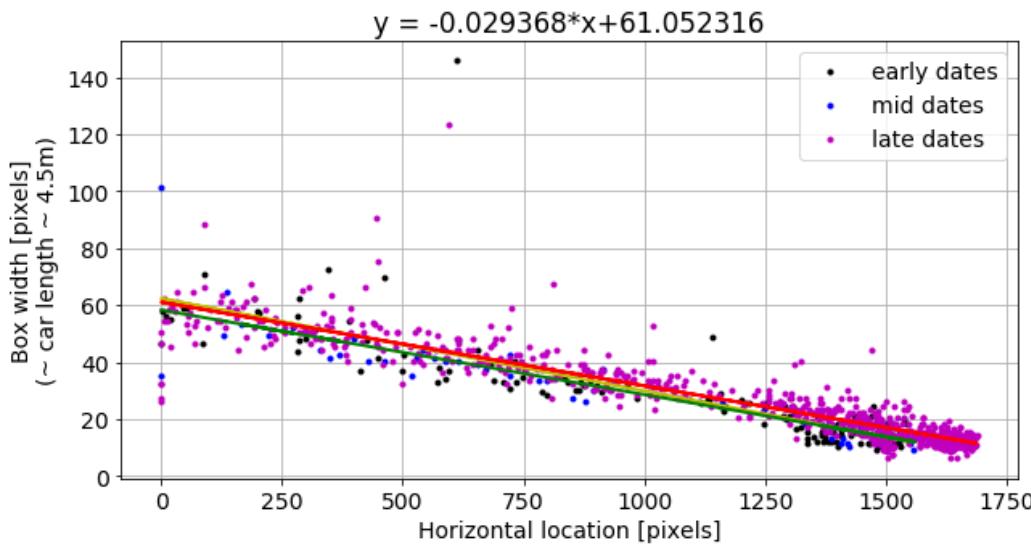
Data structure	Keys	Values	Processing needed
Raw tracking logs	time, vehicle	x, y	Non-linear transformation: pixels → meters



# Data representation

Data structure	Keys	Values	Processing needed
Raw tracking logs	time, vehicle	x, y	Non-linear transformation: pixels → meters
Per-vehicle*	vehicle, x-range	time, y, speed	Interpolation to grid points

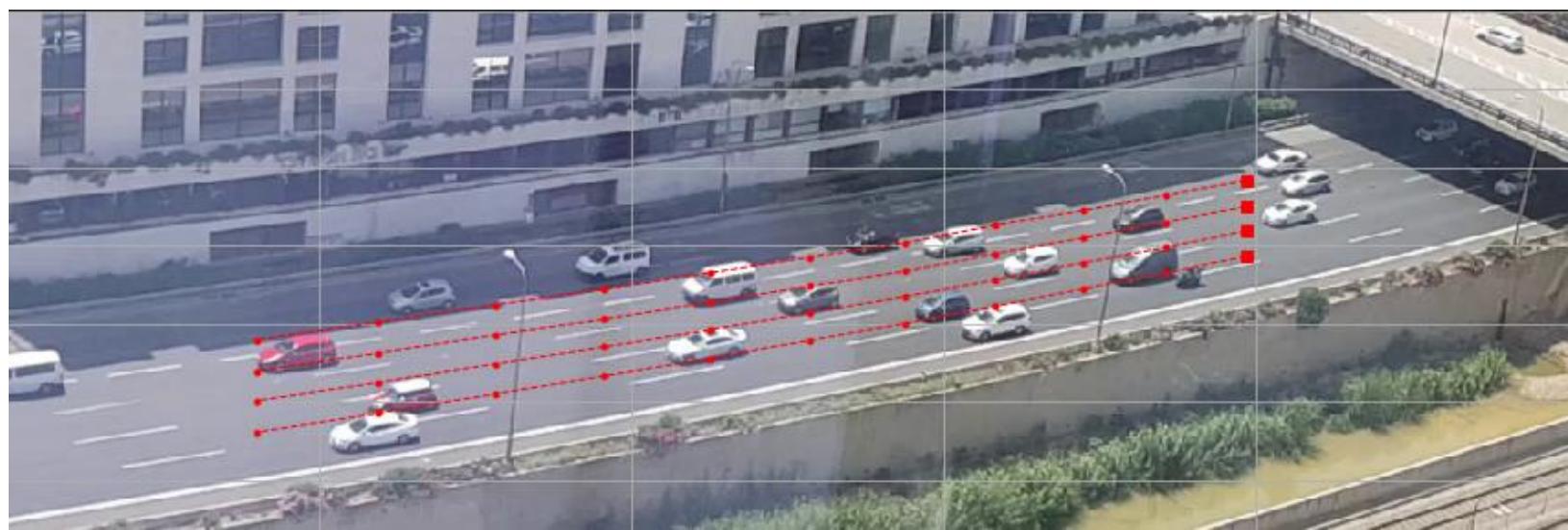
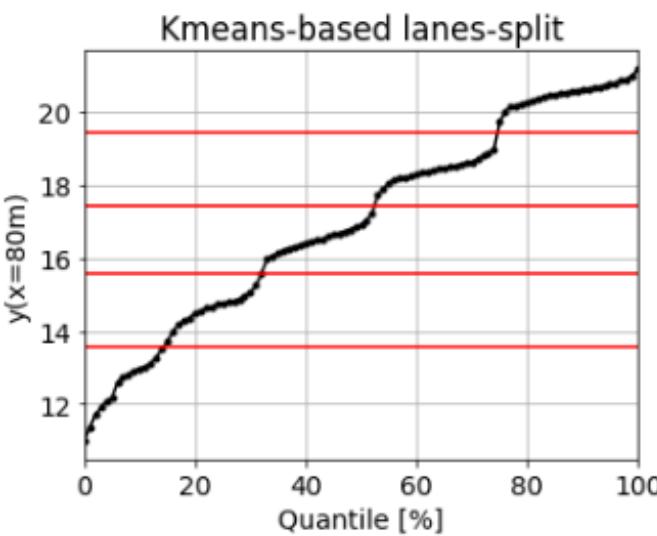
\* Can be concatenated over all videos



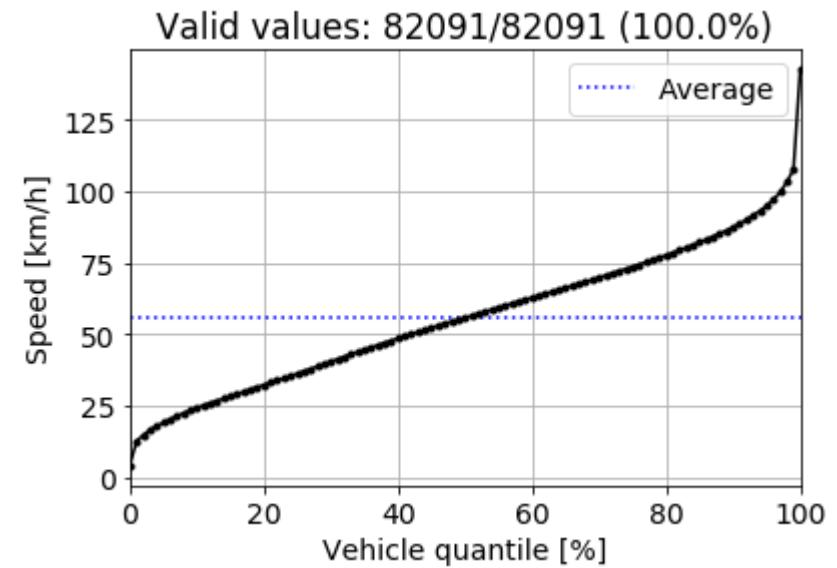
# Data representation

Data structure	Keys	Values	Processing needed
Raw tracking logs	time, vehicle	x, y	Non-linear transformation: pixels → meters
Per-vehicle*	vehicle, x-range	time, y, speed	Interpolation to grid points
Spatial*	time, lane, x-range	n_vehicles, speed	Clustering to lanes (K-means with uniform split of the vertical locations range as initial centers)

\* Can be concatenated over all videos



# Vehicles speed distribution



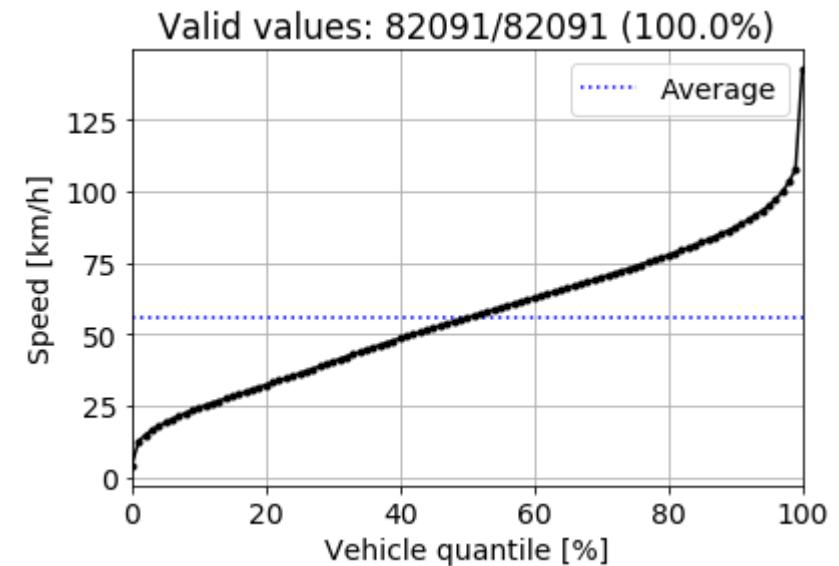
# Vehicles speed distribution



**143 km/h**



**3.9 km/h**



# More interesting frames



Sparsest frame (2 detected vehicles)

Densest frame (51 detected vehicles)

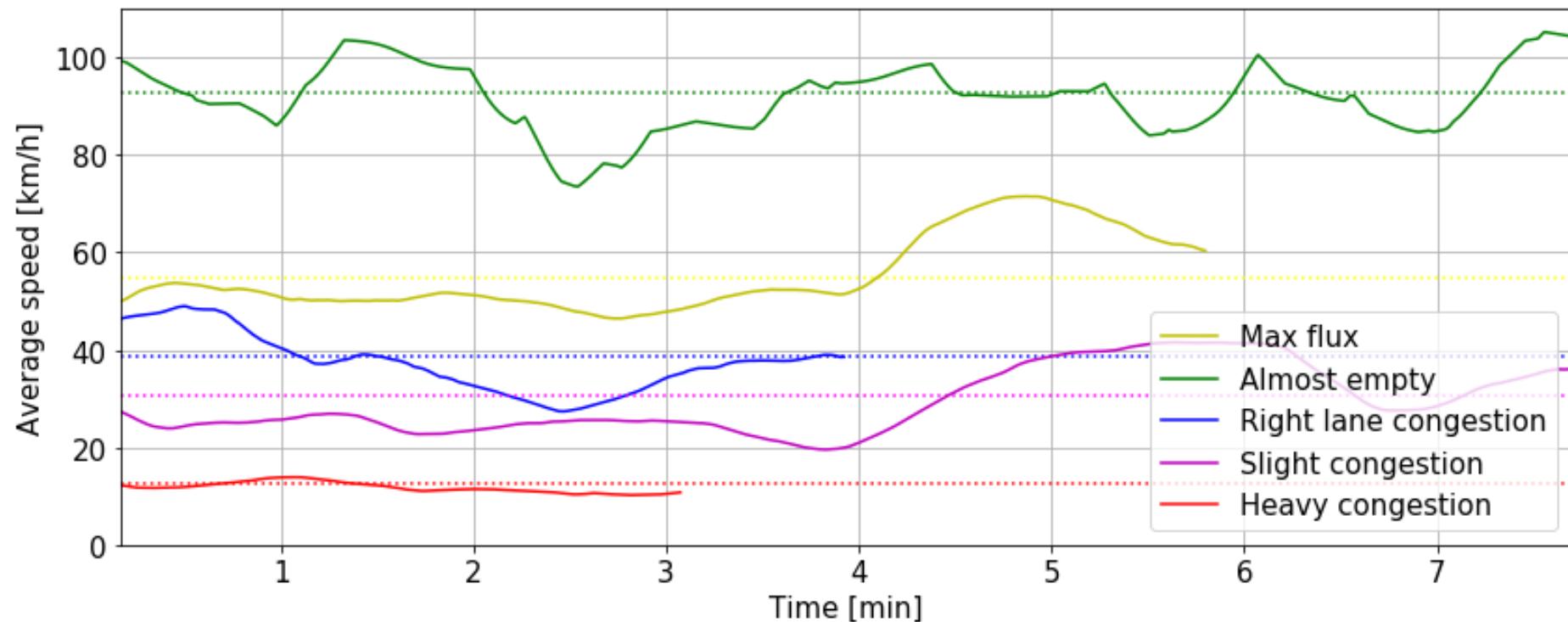


Unclear congestion on the left



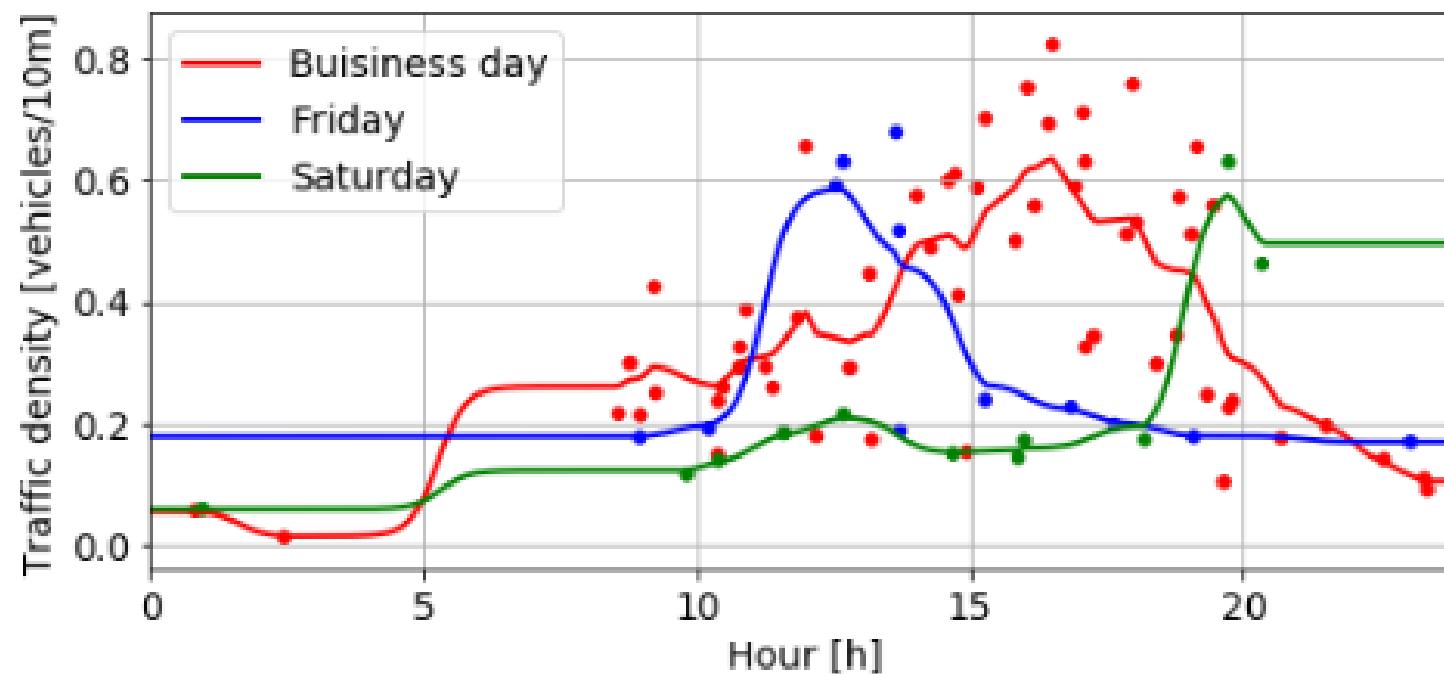
# Speed trends within videos

- Stable compared to differences between videos



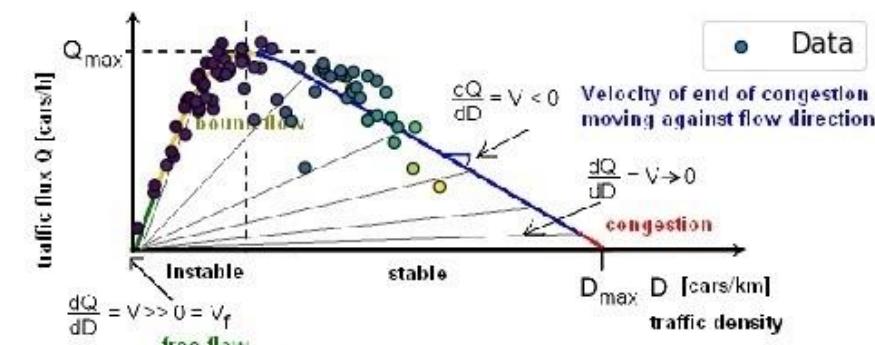
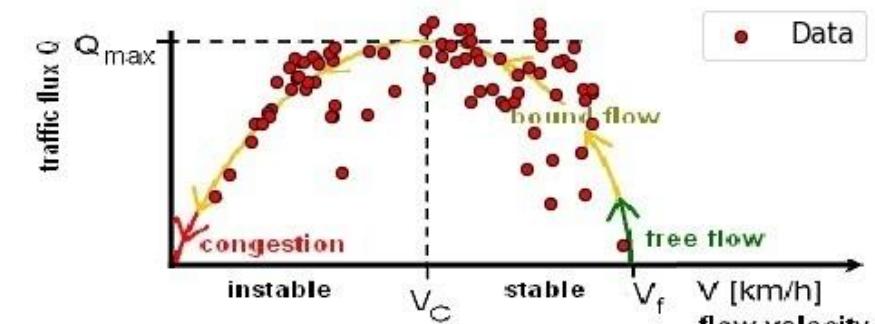
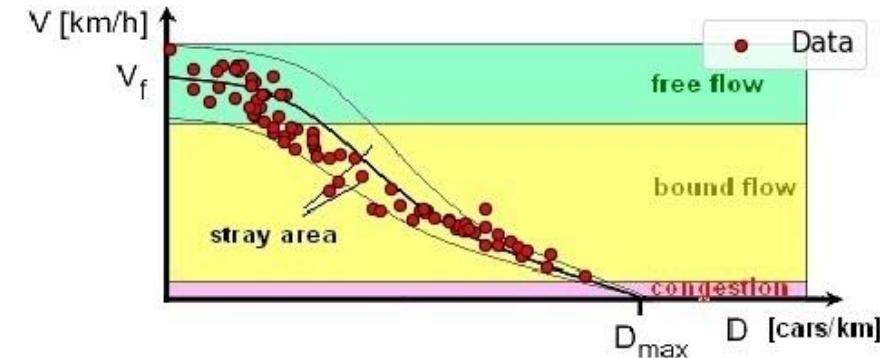
# Traffic density over days and hours

- Rush hours:
  - Business days: 16:00-17:00
  - Fridays: 12:00-13:00
  - Saturdays: 19:00-20:00



# The Fundamental Traffic Diagram

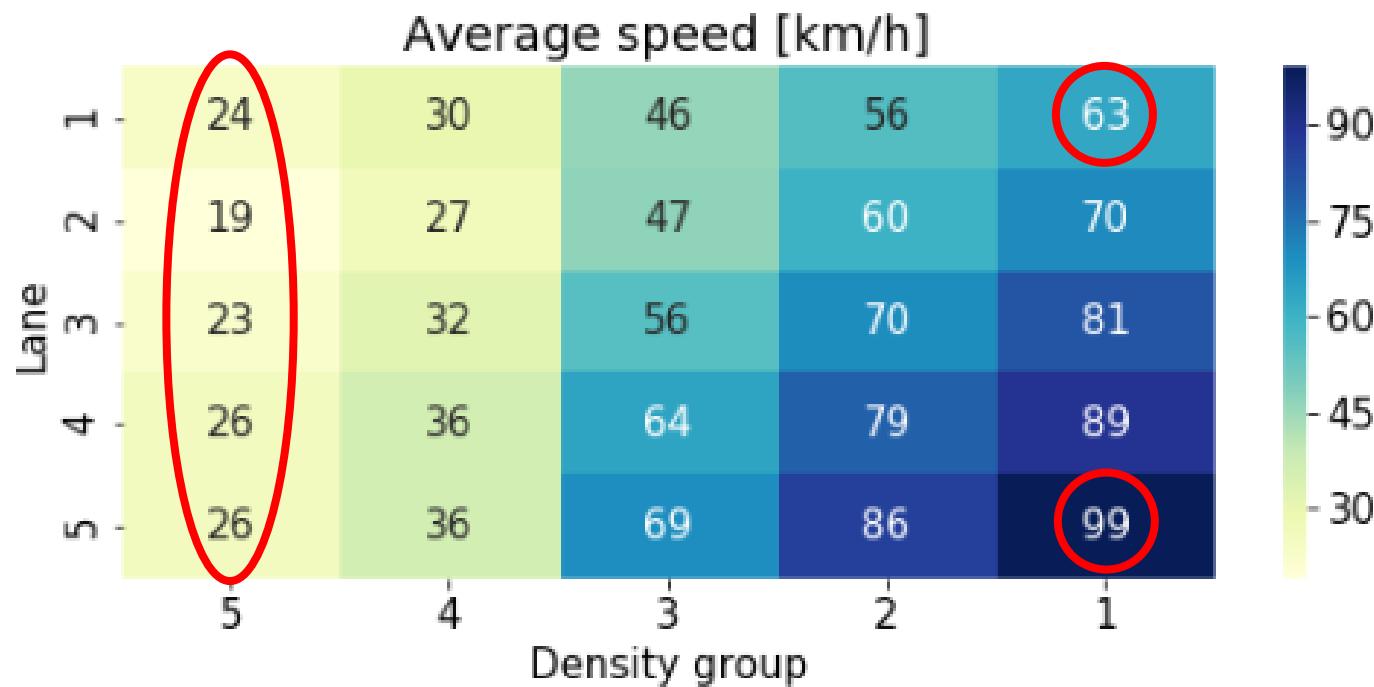
- Speed vs. density vs. flux
  - Original diagram by Hendrik Ammoser
- Data points **fit very well**
  - Clear separation between free-flow & congestion
- **Critical speed (max flux):  $\sim 60 \text{ km/h}$**
- Flux vs. density outlier
  - Congestion on the right lane only



Source: Hendrik Ammoser, Faculty of Transport and Traffic Sciences, Dresden, Germany

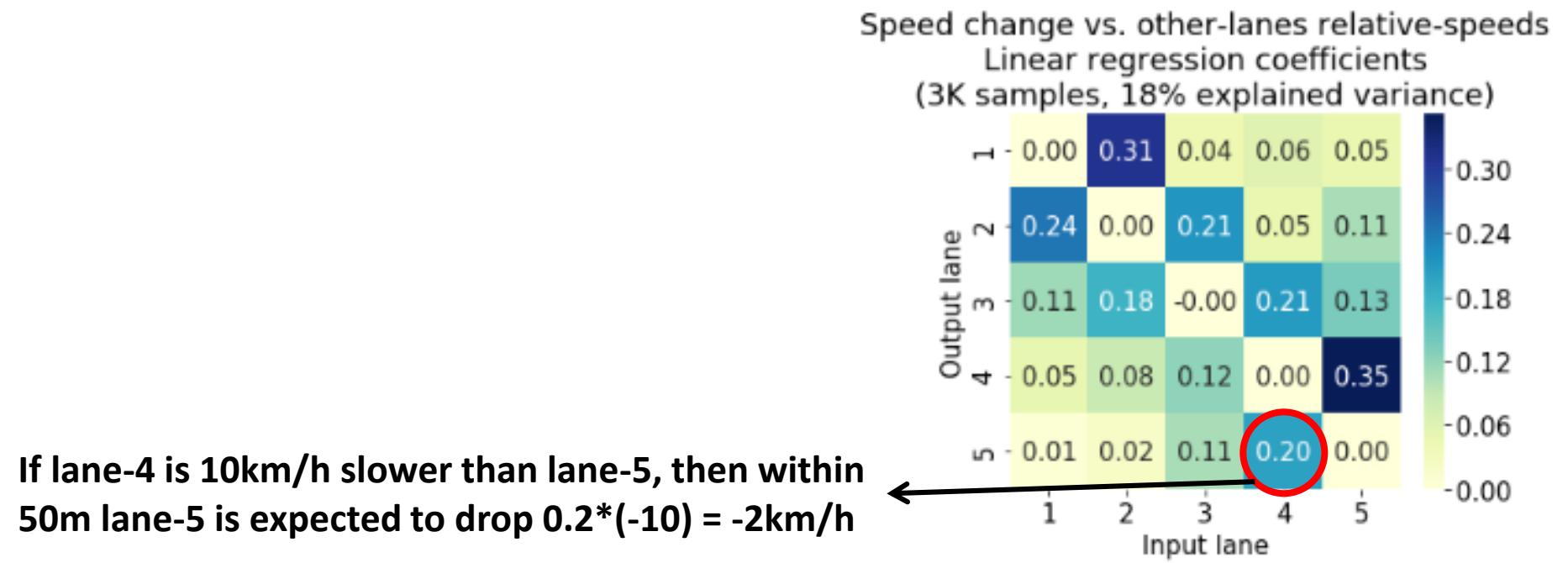
# Speed vs. lane

- Left lanes are faster – in particular in free-flow



# Can a driver trust currently-faster lanes?

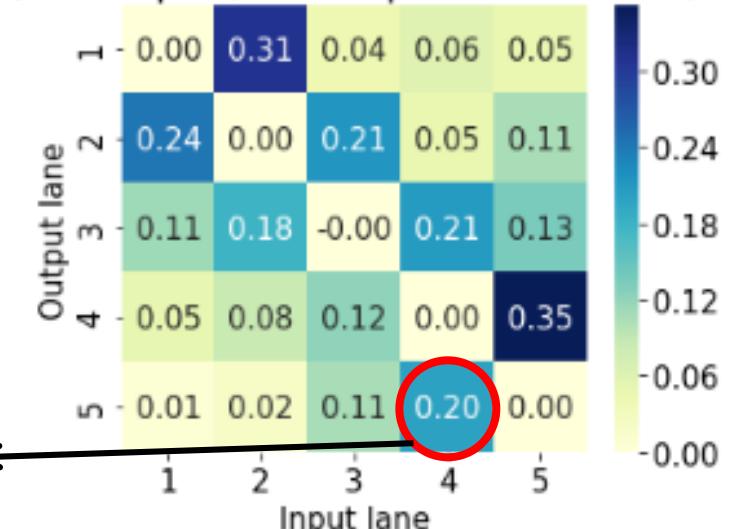
- I drive and the lanes around me move faster – will they keep moving faster?
  - **Output:** prediction of speed-changes in current lane within 50m
  - **Input:** speed differences of adjacent lanes – linearly-explain 18% of the changes out-of-sample



# Can a driver trust currently-faster lanes?

- I drive and the lanes around me move faster – will they keep moving faster?
  - **Output:** prediction of speed-changes in current lane within 50m
  - **Input:** speed differences of adjacent lanes – linearly-explain 18% of the changes out-of-sample
  - Within 50m, my speed is expected to move 20-30% towards the adjacent speeds
    - More than 50% of the speed-difference between lanes diminishes within 50m
- Note:
  - Only videos of congestions
  - Memoryless model (can't know historical speeds as the driver observed them up the road)

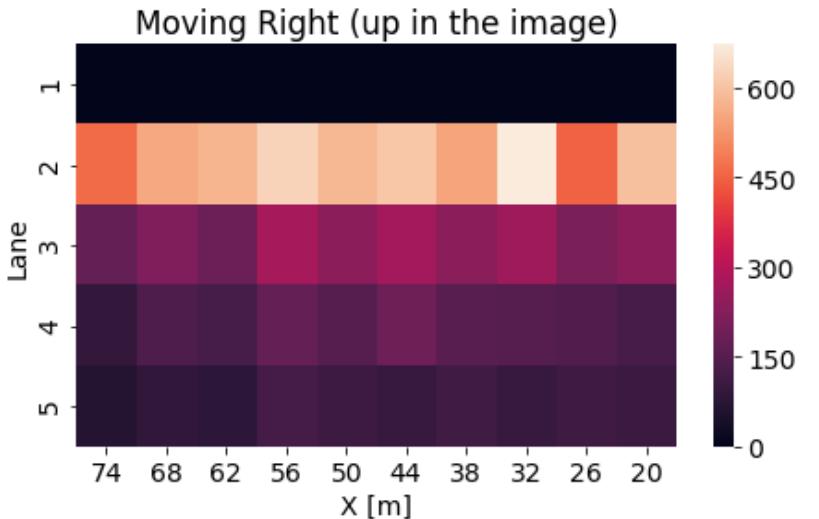
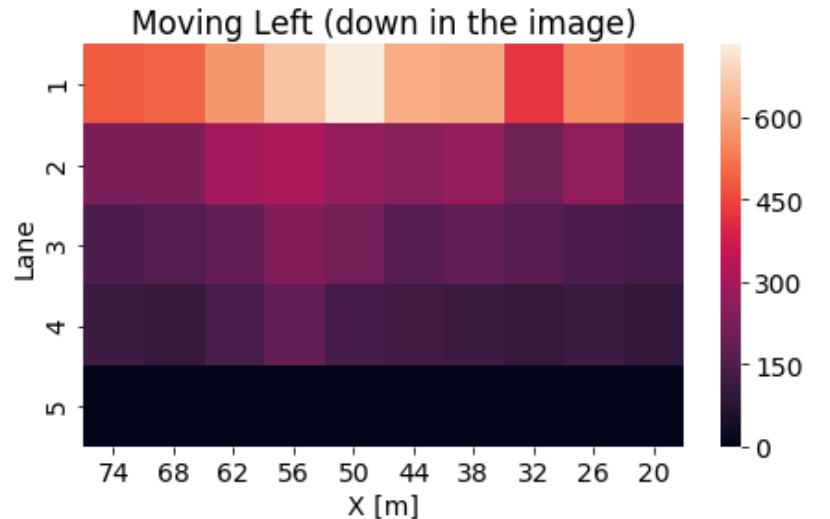
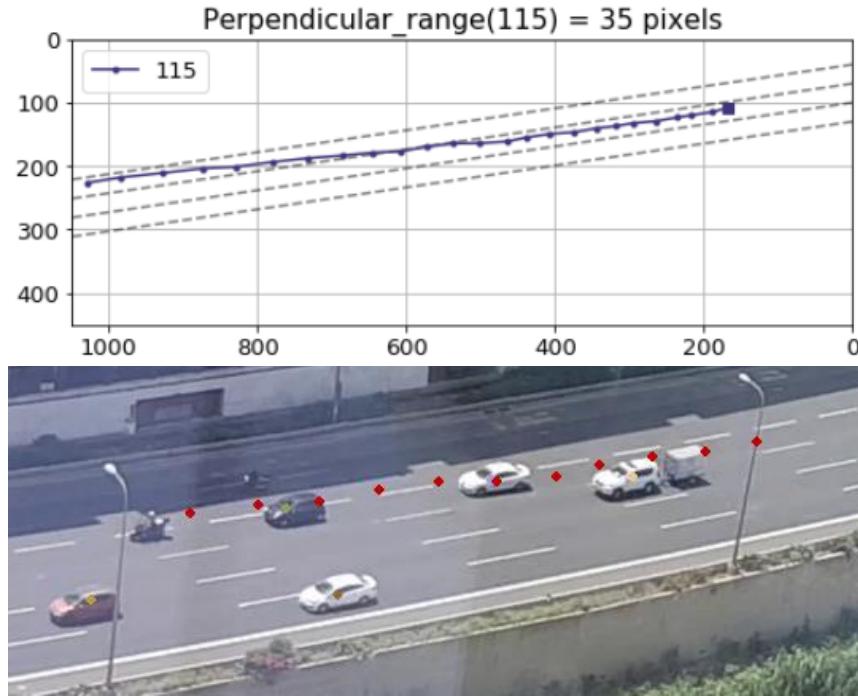
Speed change vs. other-lanes relative-speeds  
Linear regression coefficients  
(3K samples, 18% explained variance)



If lane-4 is 10km/h slower than lane-5, then within 50m lane-5 is expected to drop  $0.2 * (-10) = -2\text{km/h}$

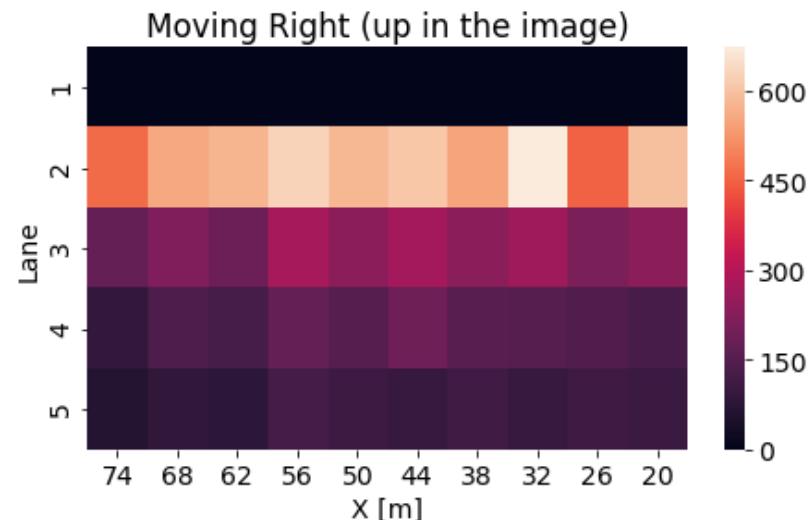
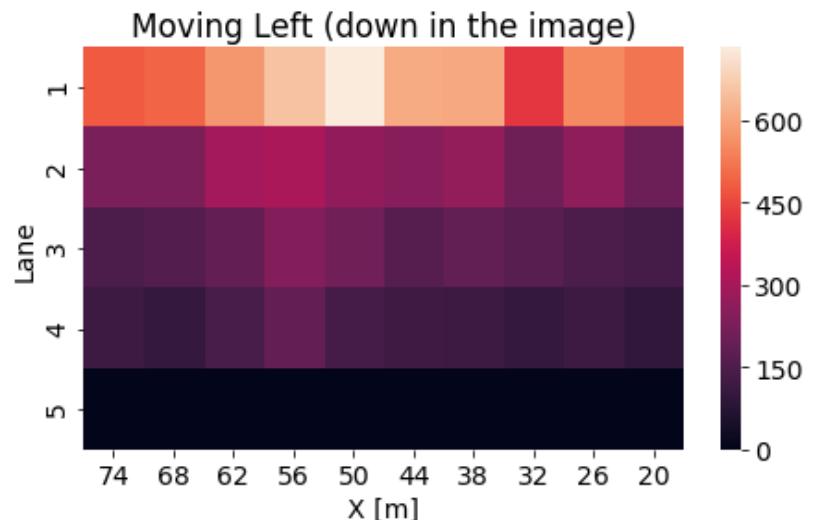
# Lane transitions

- 24K lane-transitions detected over 60m of road
- Mostly in the right – near the entries & exits



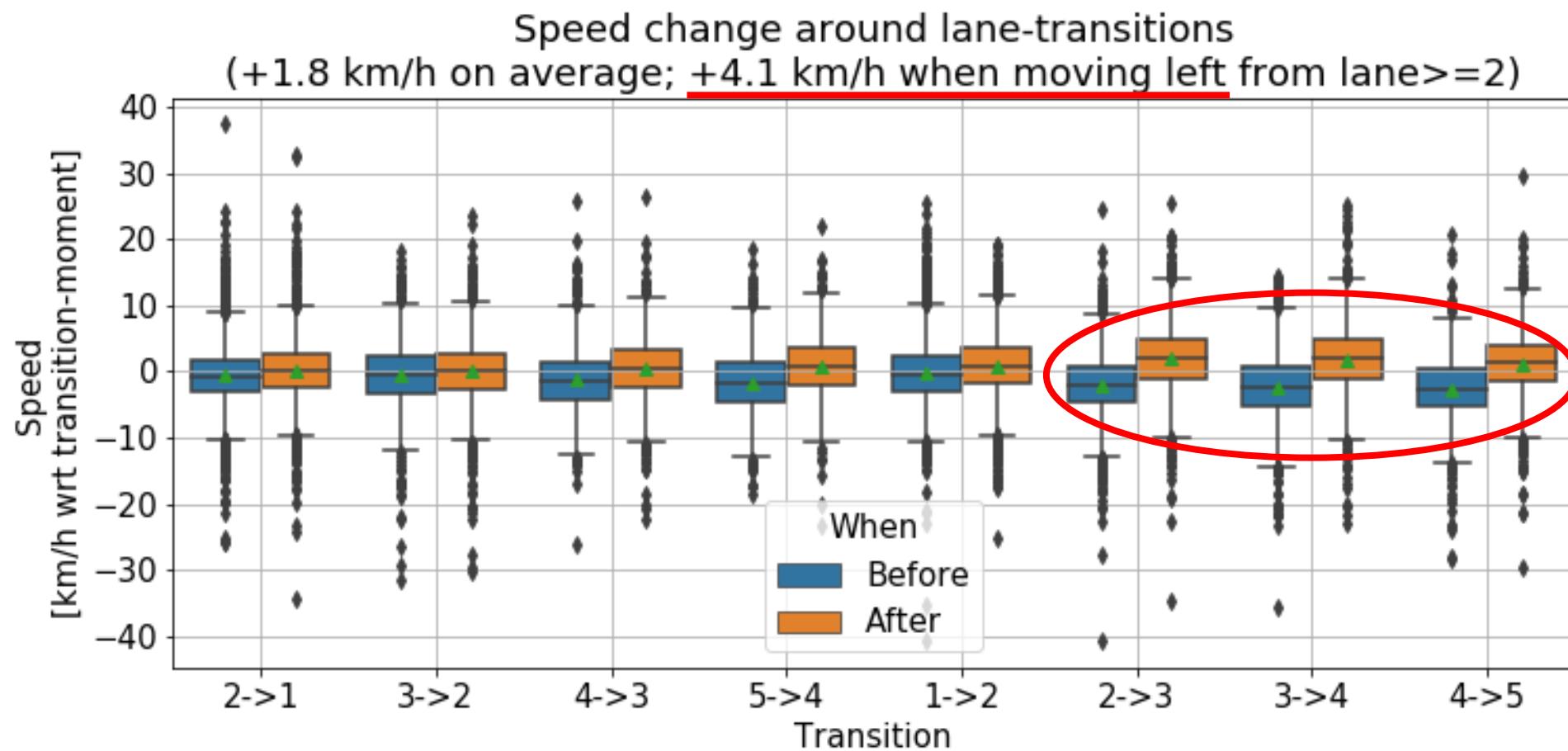
# Lane transitions

- 24K lane-transitions detected over 60m of road
- Mostly in the right – near the entries & exits
- Manual sanity check: **5/10 FP transitions** 😞
  - Close to edge of a lane (e.g. high vehicles)
  - Tracking confusions on dense traffic
- 12K true transitions / 82K vehicles = **transition per 7 vehicles**



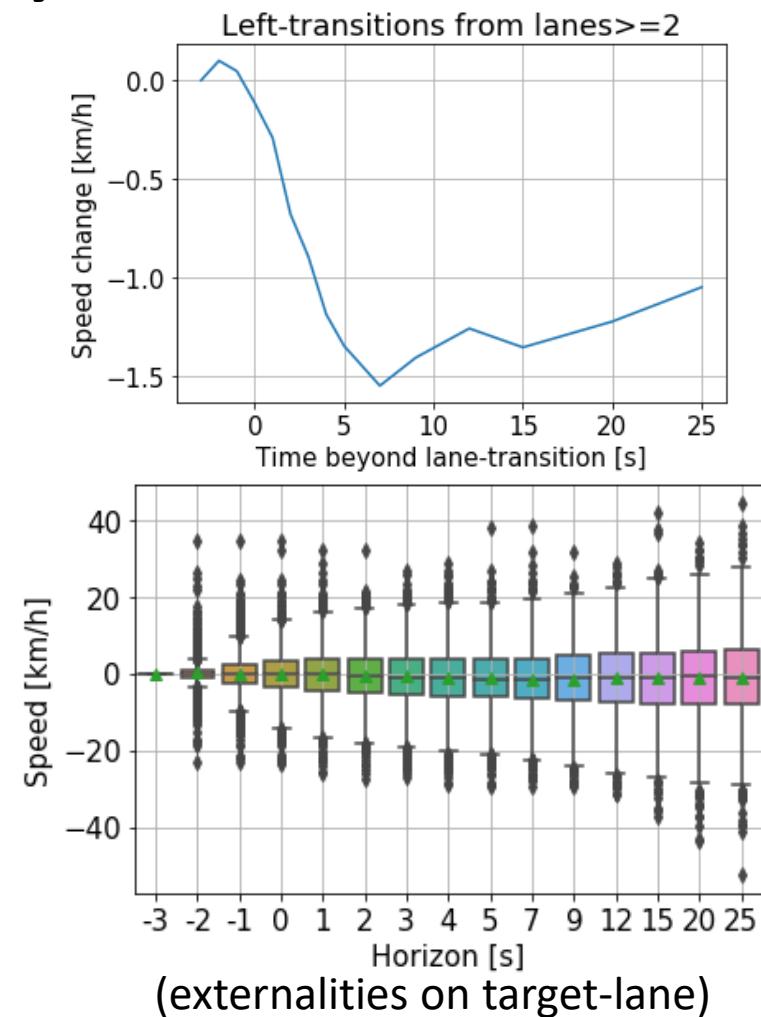
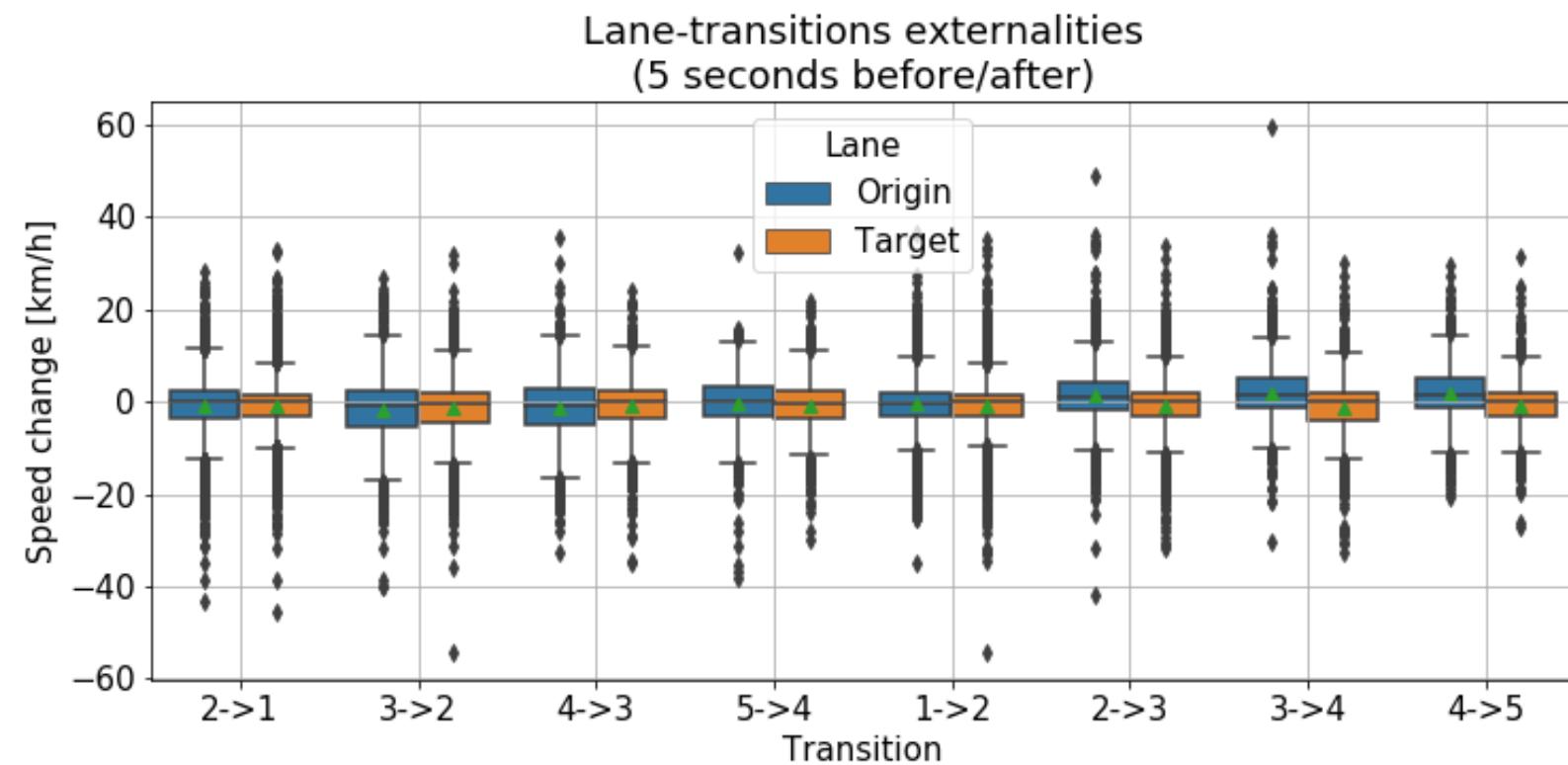
# Lane transitions

- Transition is **usually followed by speed increase** (of the same vehicle)



# Lane transitions

- Externalities (effect on other vehicles) are **too noisy** to be studied



# Summary

# Main contributions

- Demonstration of home-made applied traffic research
- Original dataset of traffic in a major road
- Detection
  - Small, crowded objects in noisy images
  - Trainable from little data
- Tracking
  - Fast-moving objects in low frame-rate videos
  - Robustness to missing detections
- Analysis
  - Validation of the *Fundamental Traffic Diagram*
  - Critical speed (max flux) – 60 km/h
  - Initial understanding of lane-transitions effects

# Methodological insights (nothing surprising...)

- Data are everywhere
- **Draw/plot/print everything:** it's often not what you expected
- **Transfer learning** is insanely powerful
- **Sampling** may be crucial for training
- Have a **pipeline** for anything useful – you'll always run it again
- **Running time profiling:** many unexpected computational pitfalls

# Possible extensions

- Data:
  - Tag before & after Hashalom interchange
- Detection:
  - Tag more data from "difficult" videos
  - Further tuning of hyper-params & architecture
  - Further output filtering (e.g. stricter manual constraints)
- Tracking:
  - Exploit size in addition to location
  - Non-linearly-additive models instead of KF:
    - Model parameters in meters (not pixels) – requiring location-dependent noise matrix
    - Large negative acceleration is more probable in higher speeds and in direction of motion
  - “Deep SORT”: exploit visual info
- Analysis:
  - ARIMA models for traffic speed or density
  - Effects of additional or removed lanes

# References

- [Project repo in GitHub](#)
- [VGG Image Annotator](#) / Abhishek Dutta et al., Oxford
- [Guide to build Faster RCNN in PyTorch](#) / Fractal research group
- [Object detection and tracking in PyTorch](#) / Chris Fotache
- [Simple Online and Realtime Tracking](#) / Alex Bewley et al.
- [Kalman Filter](#) / Wikipedia
- [The Fundamental Traffic Diagram](#) / Wikipedia



Thank you