Applying Computer Vision to traffic density estimation using public cameras

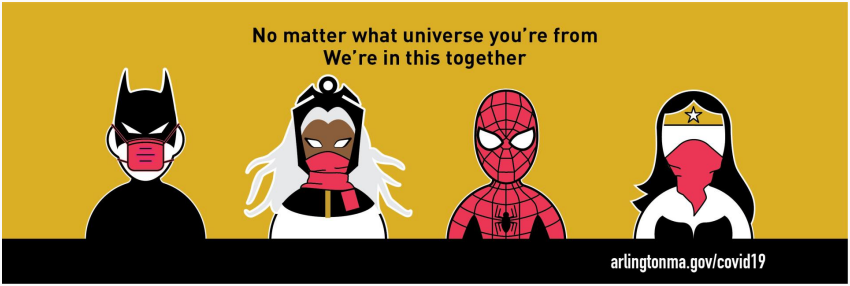
Nemanja Mićović

Introduction

Word of support to DS organizers

- Congratulations to the organizers of Data Science conference
- I always enjoy the event and talks, keep it going!

Covid 19 - We're in this together



- I hope all your family and friends are alright!
- Stay healthy and calm, we'll overcome this!
- Use this situation to re-evaluate the importance of friends and love
- And think about how little actually we need to be happy :)

About me

- Research Scientist at ML/AI team at Nordeus
- Teaching assistant at Faculty of Mathematics, Department of Informatics and Computer Science
- PhD student at Faculty of Mathematics
- Passionate about AI, ML, Gaming and Synthwave
- Contact:
  - Website
  - nmicovic@outlook.com, nemanjam@nordeus.com, nemanja_micovic@math.rs

About Nordeus

- Founded in 2010
- Crew of around 170 people and over 20 nationalities
- Based in Belgrade
- Games:
  - Top Eleven (over 220 million registered users)
  - Golden Boot (over 65 million people played so far)
  - Heroic (Unite talks: talk 1, talk 2, talk 3, talk 4, talk 5)
- ML/AI team:
  - Applying ML/AI to gaming
  - Recommendation engines, Media generation, Descriptive models, AI agents, Churn prediction...
- Visit our ML/AI blog!

About Faculty of Mathematics

- Among leading Computer Science academical institutions in Serbia
- I learned so much during my studies here, big thanks to all amazing professors and teaching assistants!
- Big thanks to all of my students who have been amazing and made my lectures so much more fun to hold!
- Collaboration with Department of Informatics and Computer Science
- Members of the Department

About Traffic Meister project

- Originally, I imagined doing it after getting seriously stuck in traffic
- Dreamed of having a system that would make sure to inform me when and where is traffic congestion
- Wanted to learn more about the problem of detecting traffic congestion, object detection and tracking...
- But most of all, wanted to help my local community and hometown!

Naxi Radio

- My favorite radio station in Belgrade
- Big thanks to Naxi for providing support and data for this project
- Thanks for a great idea and initiative for having publicly available cameras 24/7 since **2010**!

Naxi cameras

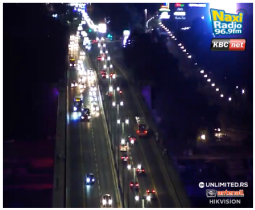
- 18 publicly available cameras available at naxi.rs
- Originally deployed in 2008 at 6 locations in Belgrade
- Motivation: Accurate and up-to-date data on traffic situation in critical points in Belgrade
- Today service for cameras has around 30 000 visits on average daily and can peak to over 200 000!
- My go-to place for checking out the traffic situation in Belgrade

Naxi cameras

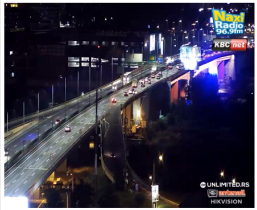
MAKSIMALNO DO 6 KAMERA ISTOVREMENO

- [x] Brankov most - Terazijski tunel
- [x] Most Gazela
- [x] Pančevački most
- [x] Bulevar Milutina Milankovića - Most na Adi
- [ ] Bulevar Mihajla Pupina
- [ ] Trg Republike
- [ ] Autokomanda
- [ ] Jurija Gagarina
- [ ] Slavija
- [ ] Trg Nikole Pašića
- [ ] Beogradski sajam - Mostarska petlja
- [ ] Takovska
- [ ] Bogoslovija
- [ ] Kružni tok Novi Beograd
- [ ] Železnička stanica
- [ ] Autoput Novi Beograd
- [ ] Bulevar Despota Stefana
- [ ] Vukov spomenik

Kvalitet prikaza: STANDARD | VISOK

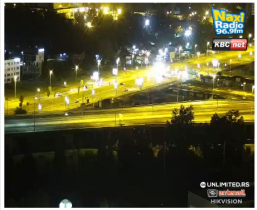


Brankov most - Terazijski tunel — Uvećaj



Most Gazela — Uvećaj

Pančevački most — Uvećaj



Bulevar Milutina Milankovića - Most na Adi — Uvećaj

Applying Computer Vision to traffic density estimation using public cameras

Defining problems to solve

Traffic situation detection problem

Given an input image, estimate the number of vehicles on the image and predict their bounding boxes.

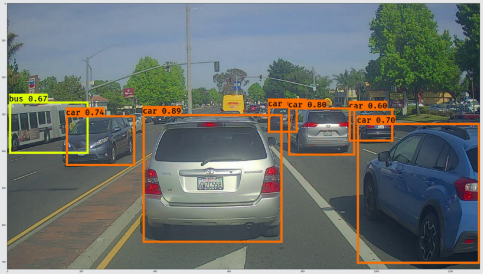


Figure 1: Vehicle detection illustration, taken from `mc.ai`

Traffic situation tracking problem

Given a sequence of video frames, detect vehicles and track their movement (assign them IDs).

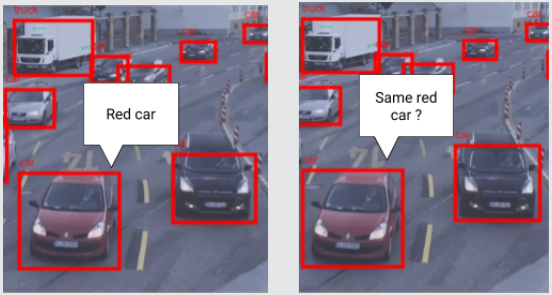


Figure 2: Vehicle tracking illustration, taken from `move-lab.com`

Vehicle detection and tracking

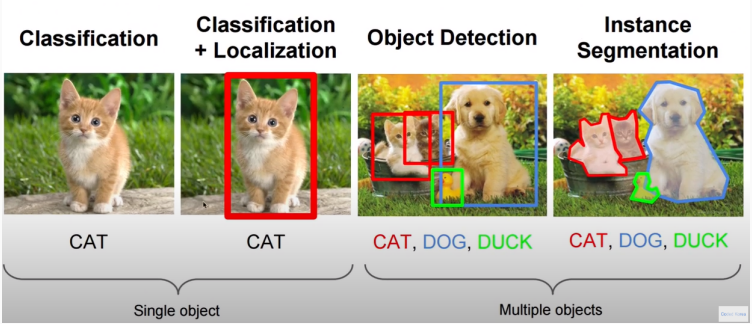


Figure 3: Computer vision problems, taken from lectures from Fei-Fei Li, Andrei Karpathy and Justin Johnson

- We will call these problems *vehicle detection* and *vehicle tracking* for short

Vehicle detection

Vehicle detection

- GOAL: Perform object detection with 1 class called **vehicle**
- One class was chosen as a starting point
- Made labeling much easier and **faster**
  - In future, vehicle should be broken down into car, truck, bus...
- There are numerous object detection models available to try out
- Mask R CNN (He et al. 2017) was chosen for a few reasons:
  - My previous experience with region-based object detectors
  - Really nice implementation based on Keras (and TensorFlow) library
  - Accuracy of predictions
  - Setting a base-line for further work
  - Time constraints - I wanted a working prototype as soon as possible
- Region based object detector won't be able to work in real time - not a problem at the moment

Data set

- Using an existing data set isn't that easy
- Naxi cameras are rather unique with regards to angle, position and quality
- I wanted to discover **how much** images would I need to train a detector with transfer learning without training on other data sets
- By *without training on other data sets*, I mean without training on additional vehicle datasets
- So I wrote a few scripts to collect data and downloaded images from all available cameras
- Resolution of images is 704×576

Data set

Figure 4: Image from Naxi camera, location 0 - *Brankov most*

Data set

Figure 5: Image from Naxi camera, location 1 - *Gazela*

Data set

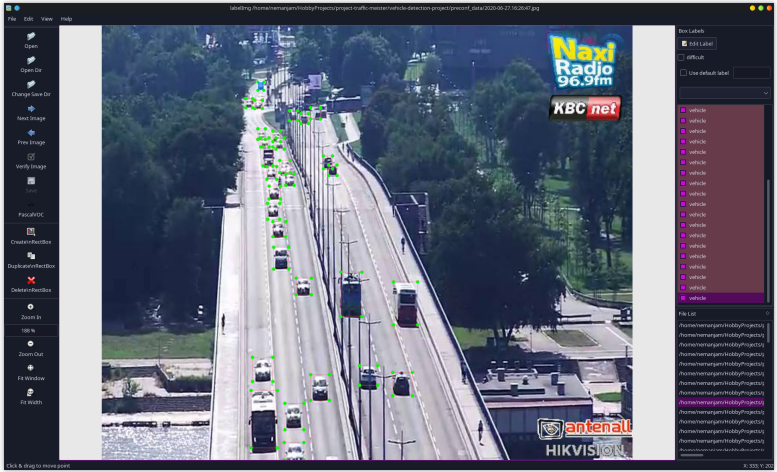
Figure 6: Image from Naxi camera, location 2 - *Pančevački most*

Labeling data set

- But to train an object detector, I we need bounding boxes
- Which someone needs to label. . . ok I'll do it! :)
- Goal: Prove things actually run and work (also TensorFlow, CUDA, cuDNN versions, saving/loading weights, saving progress) - that doesn't require too many labels
- There were two iterations of labeling
  - Iteration 1, dataset A: Labeled 54 photos from location *Gazela* (2176 vehicles)
    - Is this too little for decent accuracy? (It seems so, at least for that location)
    - Will it generalize on other cameras? (It did, though not to really amaze you)
  - Iteration 2, dataset B: Labeled 82 photos from locations 0, 1, 2, 3, 4 (3028 vehicles)
- Final data set C was a combination of previous two (some night images were removed)
  - 117 images
  - 5115 vehicles

Labeling data set

- Python based LabelImg tool was used
- So many rectangles were created. . . avoid this if you have OCD :)



Object Detection: Datasets

	PASCAL VOC	ImageNet	MS-COCO
Year	2010	2014	-
Number of classes	20	200	80
Number of images (train + val)	∼20k	∼470k	∼120k
Mean objects per image	2.4	1.1	7.2

Object Detection: IoU

- Metric used to measure the quality of a bounding box in object detection problems
- Calculated between ground truth (green squares) and candidates given by the detection algorithm (red squares)

Figure 7: IoU illustration, from pyimagesearch

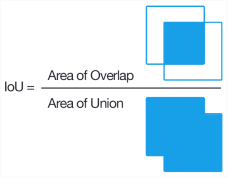


Figure 8: How to calculate IoU, from pyimagesearch

Object Detection: Evaluation

- Metric *mean average precision* - mAP
- Average of average precision (AP) per class
- Check out this article for quality comparison between different object detectors

Mask RCNN

- Mask RCNN comes from the family of *Region-based Convolutional Neural Networks*
- Multiple iterations throughout the last few years:
  - R-CNN (Girshick et al. 2013)
  - Fast R-CNN (Girshick 2015)
  - Faster R-CNN (Ren et al. 2015)
  - Mask R-CNN (He et al. 2017)

R-CNN

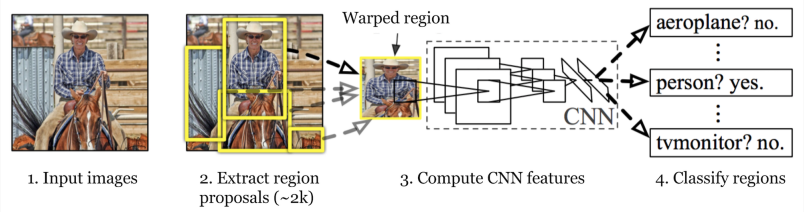


Figure 9: R-CNN architecture, from original paper

- Using selective search, identify multiple bounding box region candidates
- Run a pre-trained CNN through the region proposals (rescale them to fixed size) to get CNN features
- Run a classifier on top of CNN features - support vector machine was used **per class** in original paper

Non-Max Suppression

- Sort all bounding boxes of the same object type by confidence score
- Execute a greedy algorithm that drops bounding boxes with small confidence

Figure 10: Handling problem of multiple bounding boxes, from DRM paper

Fast R-CNN

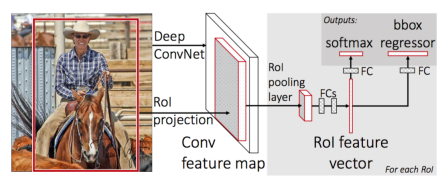


Figure 11: Fast R-CNN architecture, from original paper

- R-CNN comes with lots of computation overhead
  - CNN for image classification and feature extraction
  - Regression model for bounding boxes
  - SVM classifiers per object class
- Fast R-CNN tries to unify this into one system
- Avoids extracting feature vectors for each region proposal
- It aggregates region proposals into one CNN forward pass

Fast R-CNN - train time

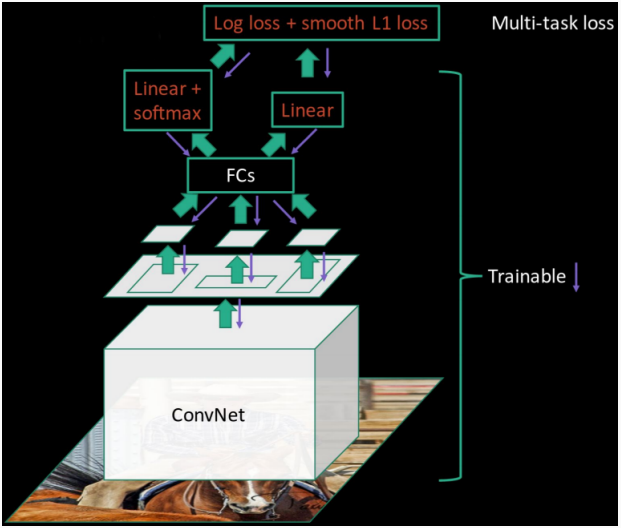


Figure 12: Fast R-CNN architecture, train time, from original paper

Fast R-CNN - test time

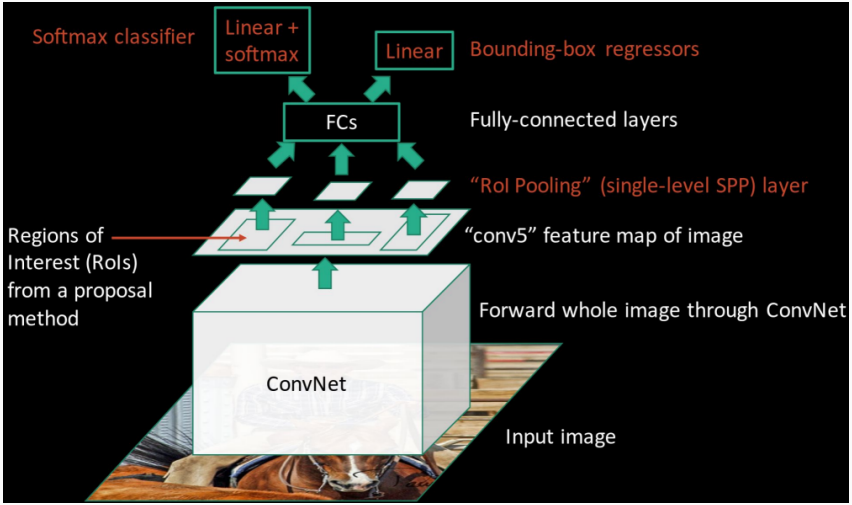


Figure 13: Fast R-CNN architecture, test time, from original paper

Fast R-CNN - RoI pooling

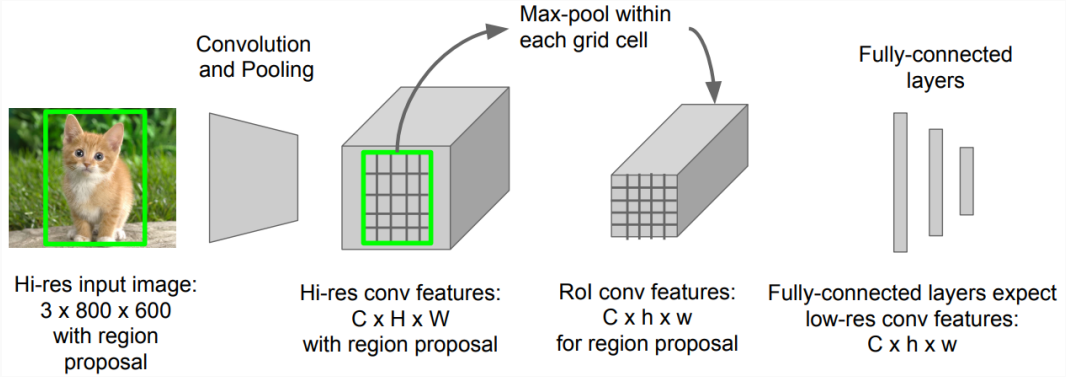


Figure 14: RoI pooling, from Stanford CS231n slides

Fast R-CNN vs R-CNN

	R-CNN	Fast R-CNN
Training Time:	84 hours	9.5 hours
(Speedup)	1x	8.8x
Test time per image	47s	0.32s
(Speedup)	1x	146x
mAP (VOC 2007)	66.0	66.9

Faster R-CNN

- Let's go faster
- Make CNN do region proposals also
- It was a bottleneck for Fast R-CNN
- Region Proposal Network (RPN) is added after last convolutional layer

Faster R-CNN

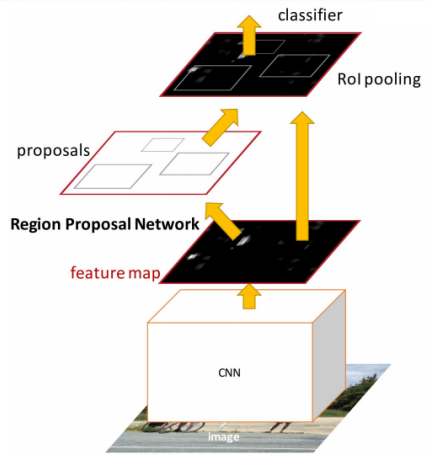


Figure 15: RoI pooling, from Stanford CS231n slides

Faster R-CNN

- There is additional complexity and details hidden away that isn't too much relevant for this talk
- Training process is also rather complex
- More details in papers, in this article and in Stanford Object Detection lectures

Mask R-CNN

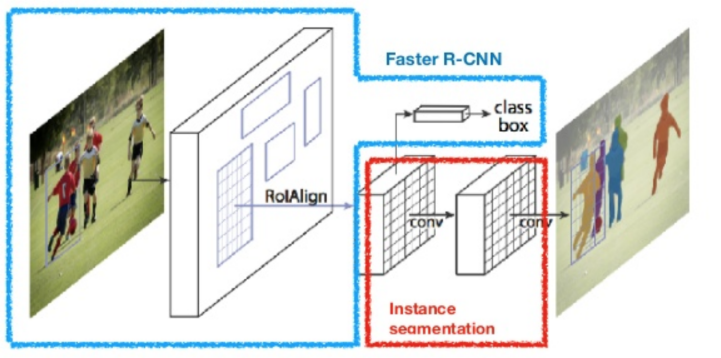


Figure 16: Mask R-CNN, from original paper

- Extends the Faster R-CNN model with pixel-level image segmentation
- Adds a RoI align layer because segmentation requires additional accuracy

Mask R-CNN

Figure 17: Mask R-CNN on COCO data set, from original paper

To recap

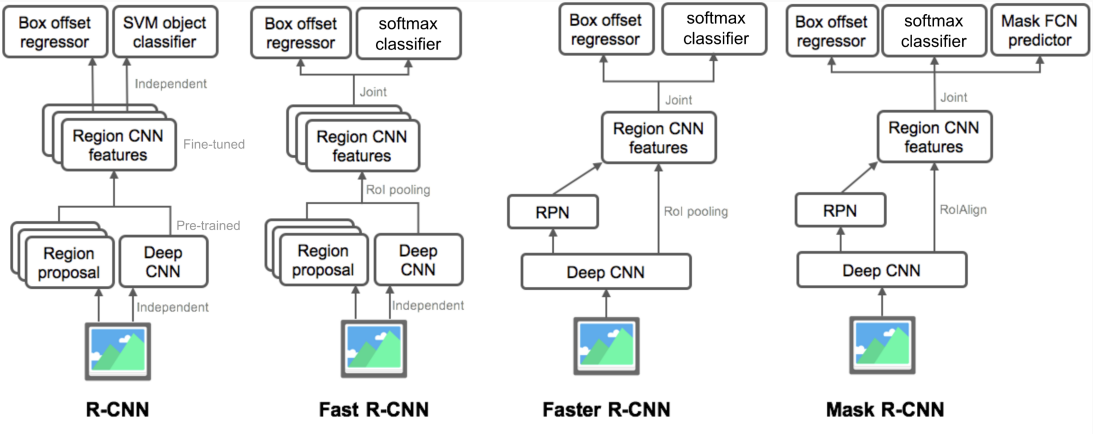


Figure 18: Region proposal approach models, taken from here

Applying Mask R-CNN to vehicle detection

Model 1

- First demo model was trained on dataset A (54 images, 2176 vehicles)
- Transfer learning was used with weights from COCO data set (link - 245.6MB)
- Model was trained on RTX 2070 for 5 epochs (to test things out)
- Out of 54 images, 20% was taken random for validation

Model 1 - training details

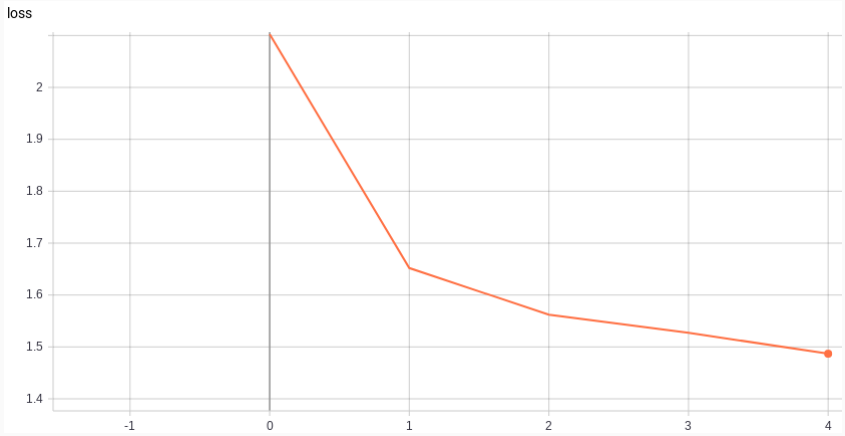


Figure 19: Loss changes through training

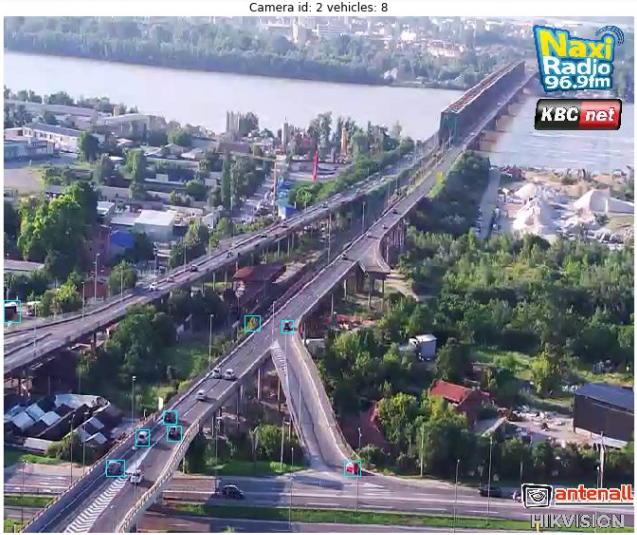
Model 1 - training details

Figure 20: Val loss changes through training

Model 1 - some results

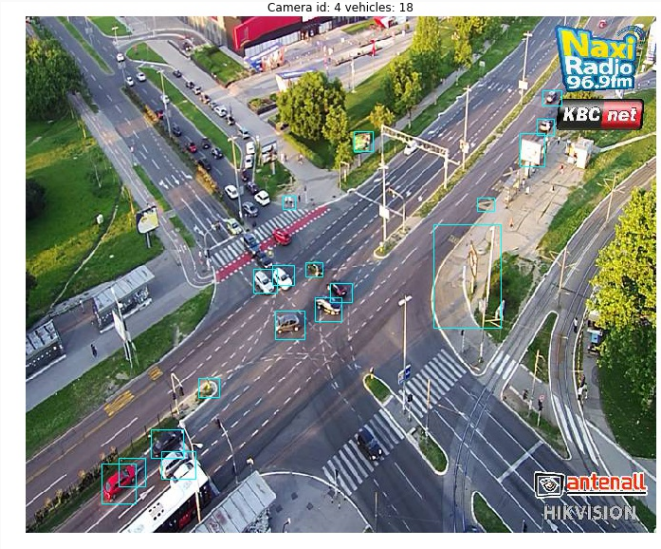
Model 1 - some results

Model 1 - some results



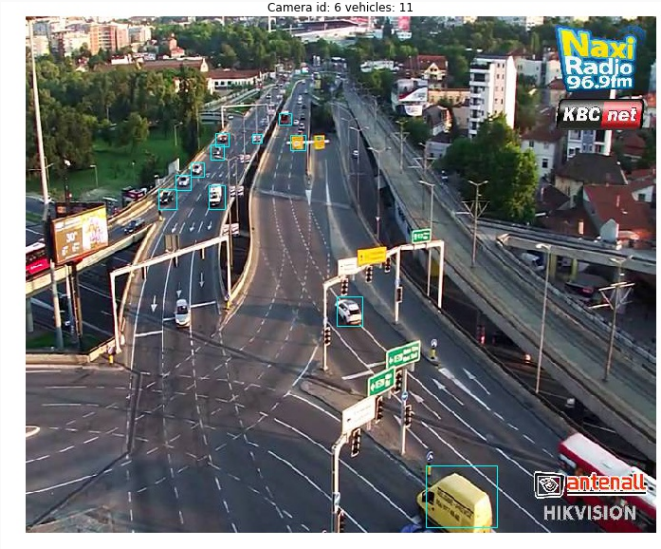
Model 1 - some results

Model 1 - some results

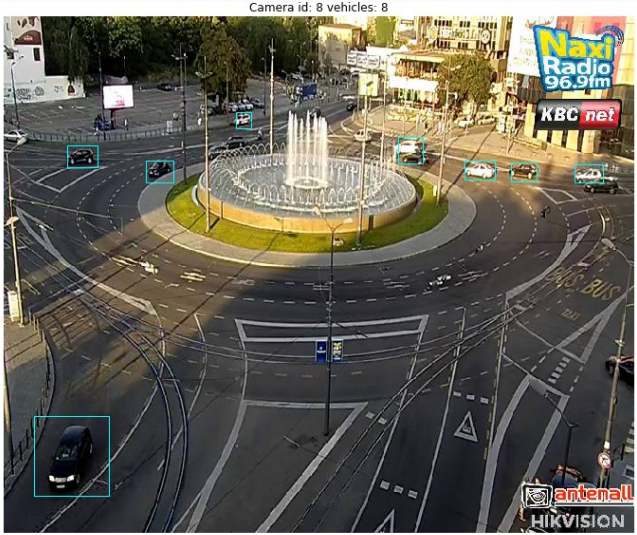


Model 1 - some results

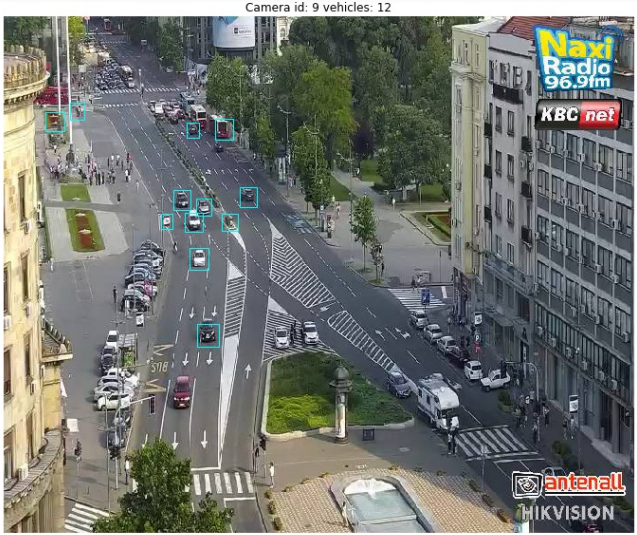
Model 1 - some results



Model 1 - some results

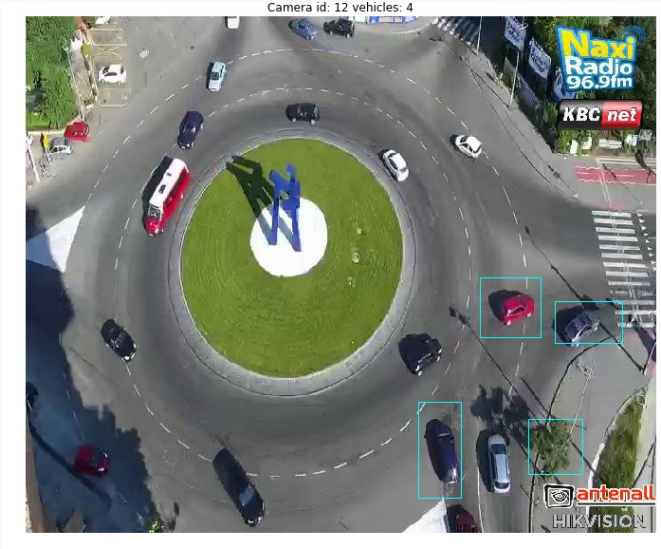


Model 1 - some results



Model 1 - some results

Model 1 - some results



Model 1 - comment

- To remind you, model was only trained on images from camera 0

Model 2

- So things work!
- Time for more data
- Dataset B was created and dataset C was used for training (117 images, 5115 vehicles)
- Random 20% of data was used for validation set
- Training was run for 50 epochs

Model 2 - training details

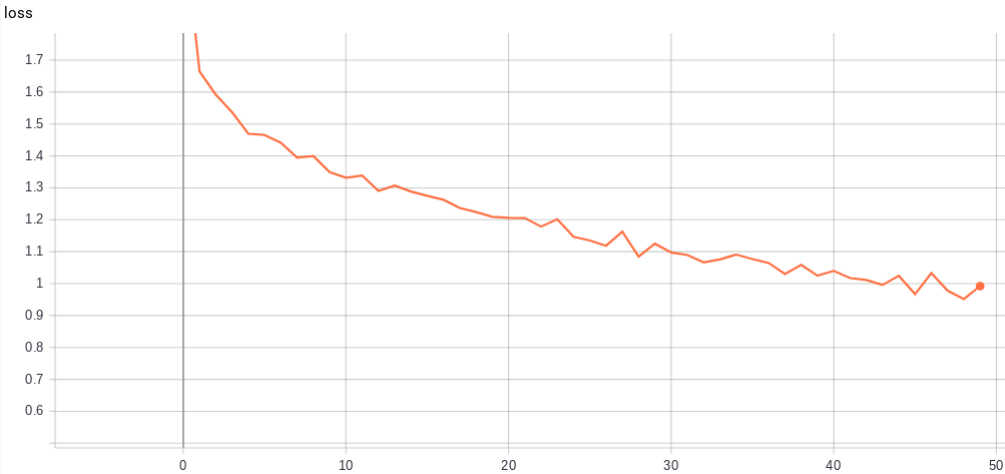


Figure 22: Loss changes through training

Model 2 - training details

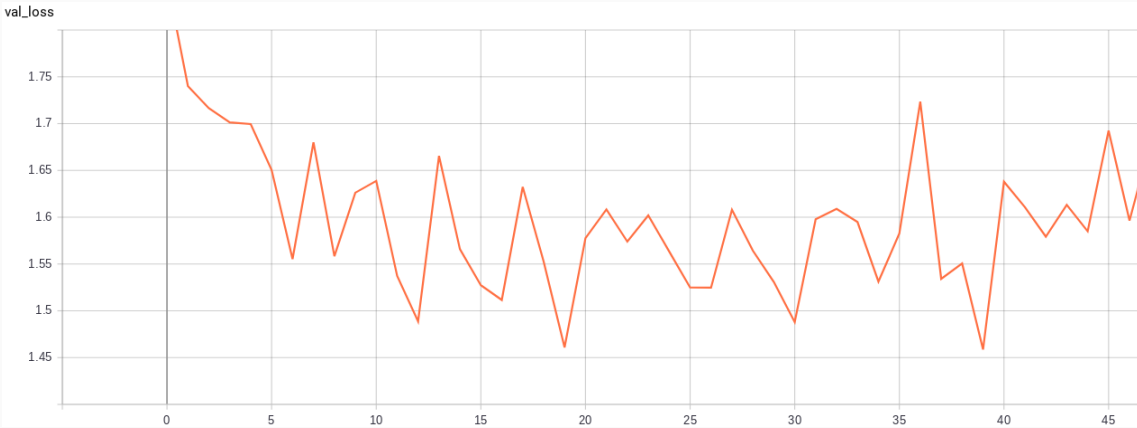
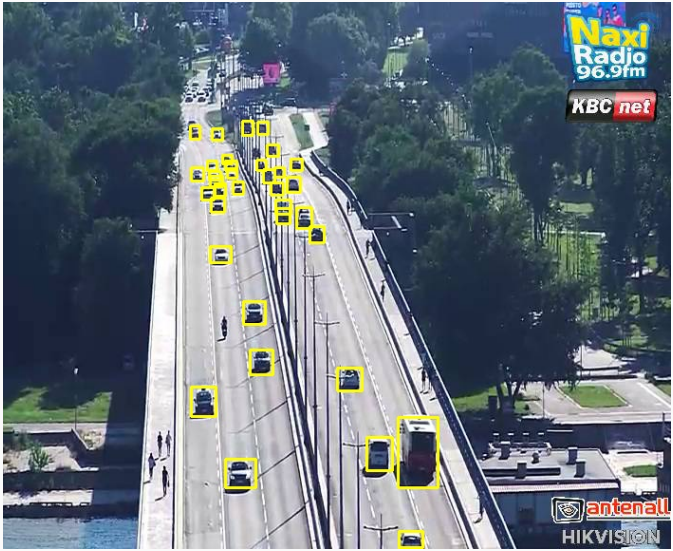


Figure 23: Val loss changes through training

Model 2 - training details

- Model at epoch 40 was chosen as it had lowest validation loss (1.459)
- This is the final model that is used currently

Model 2 - Brankov most (seen during training)

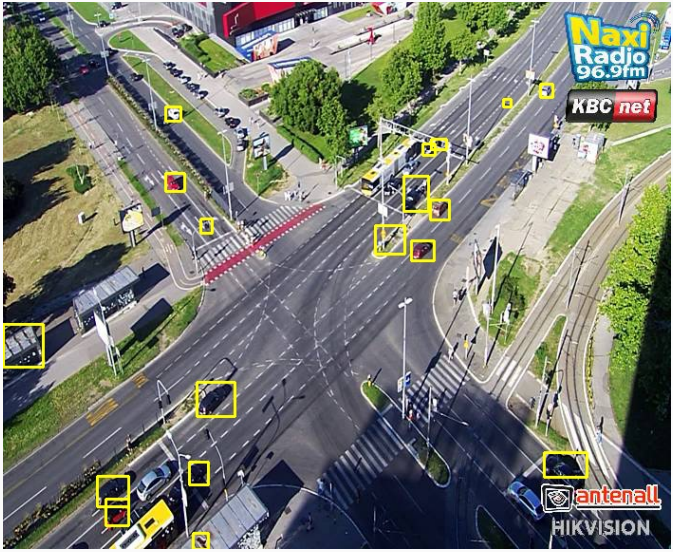


Model 2 - Gazela (seen during training)

Model 2 - Pačevački most (seen during training)

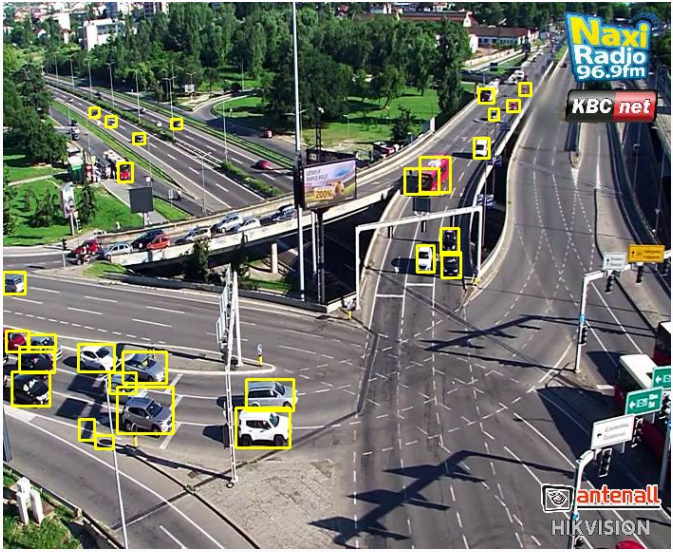
Model 2 - Bul. Milutina Milankovića (seen during training)

Model 2 - Bul. Mihajla Pupina (seen during training)

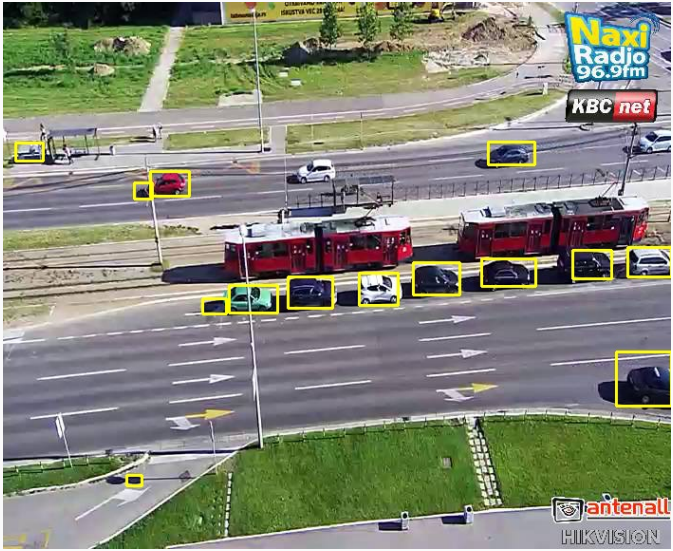


Model 2 - Trg Republike (not seen during training)

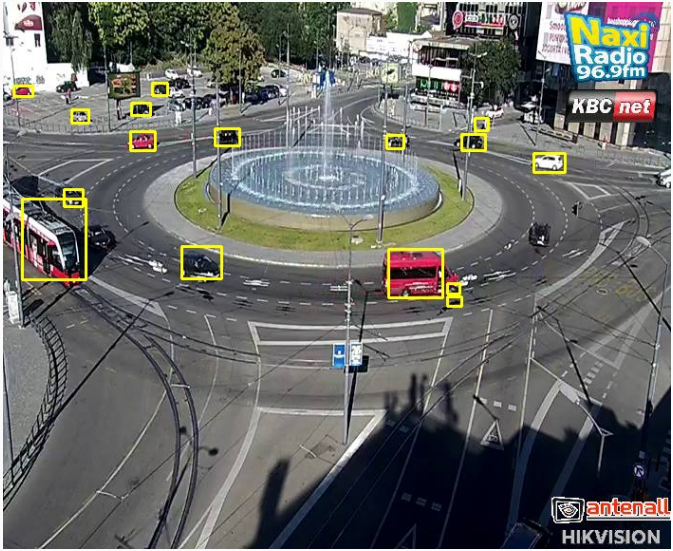
Model 2 - Autokomanda (not seen during training)



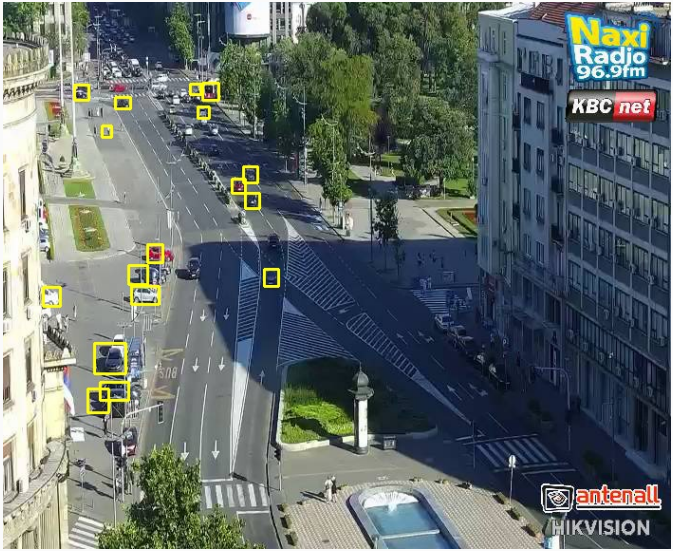
Model 2 - Jurija Gagarina (seen during training)



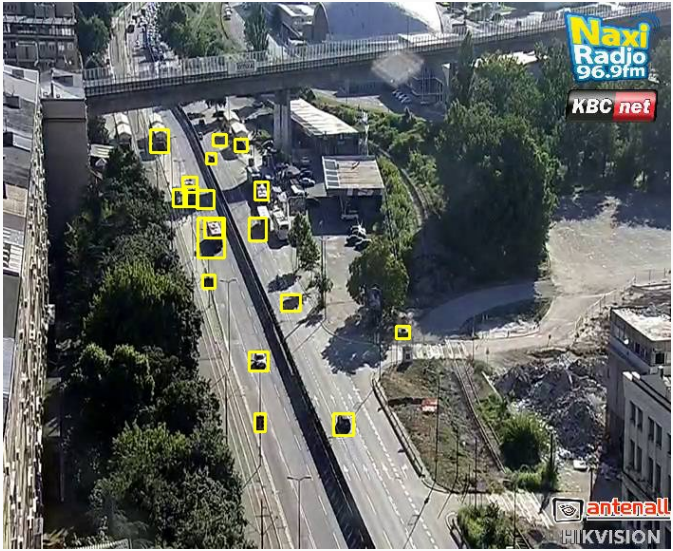
Model 2 - Slavija (not seen during training)



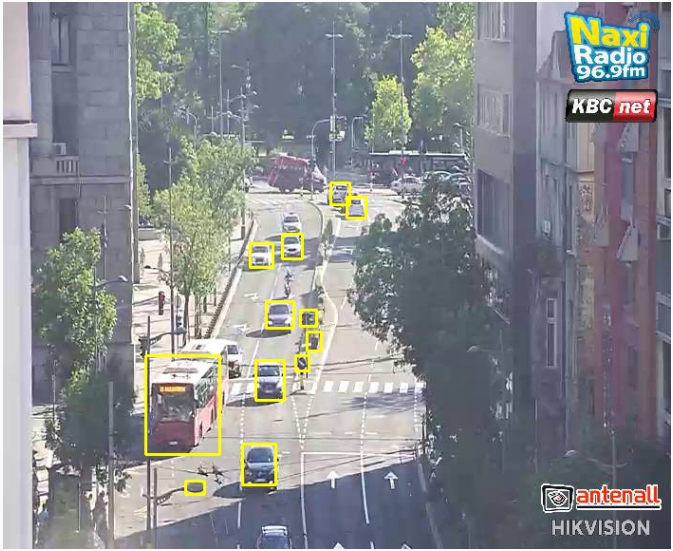
Model 2 - Trg Nikole Pašića (not seen during training)



Model 2 - Beogradski Sajam (not seen during training)

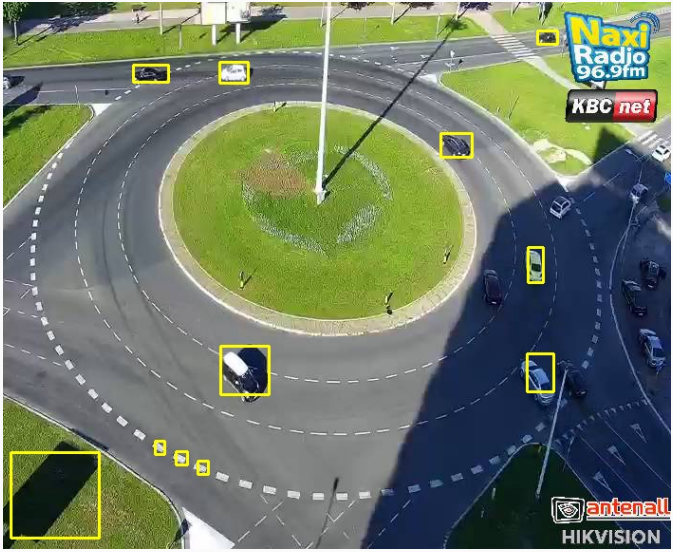


Model 2 - Takovska (not seen during training)



Model 2 - Bogoslovija (not seen during training)

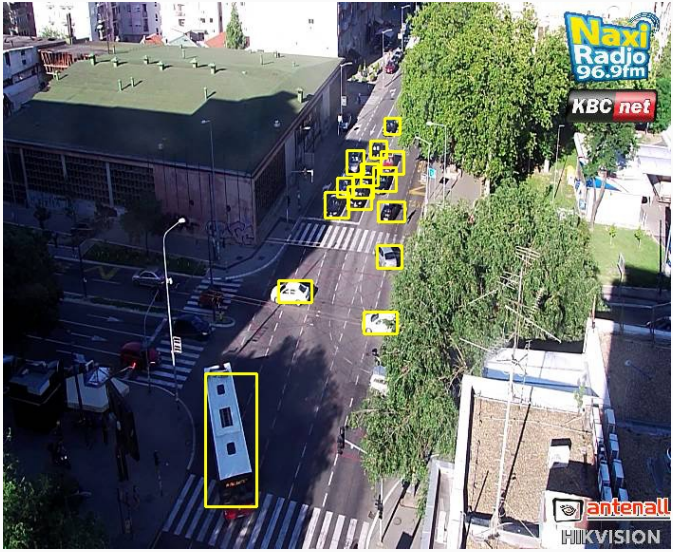
Model 2 - Opština Novi Beograd (not seen during training)



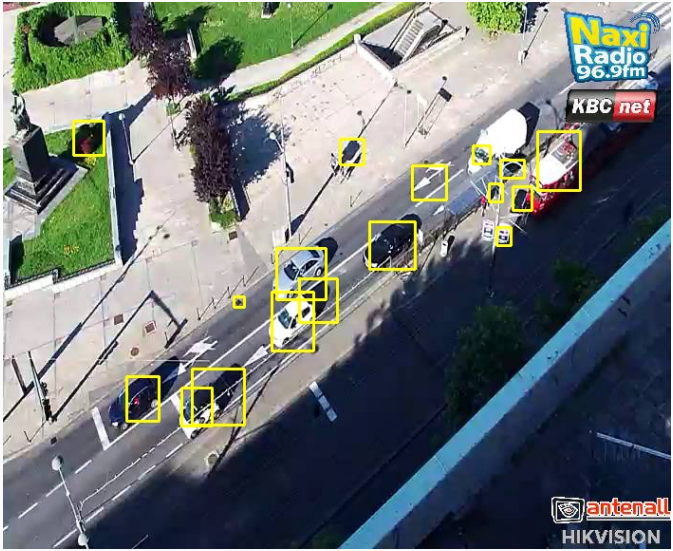
Model 2 - Železnička Stanica (not seen during training)

Model 2 - Autoput Novi Beograd (not seen during training)

Model 2 - Bul. Despota Stefana (not seen during training)



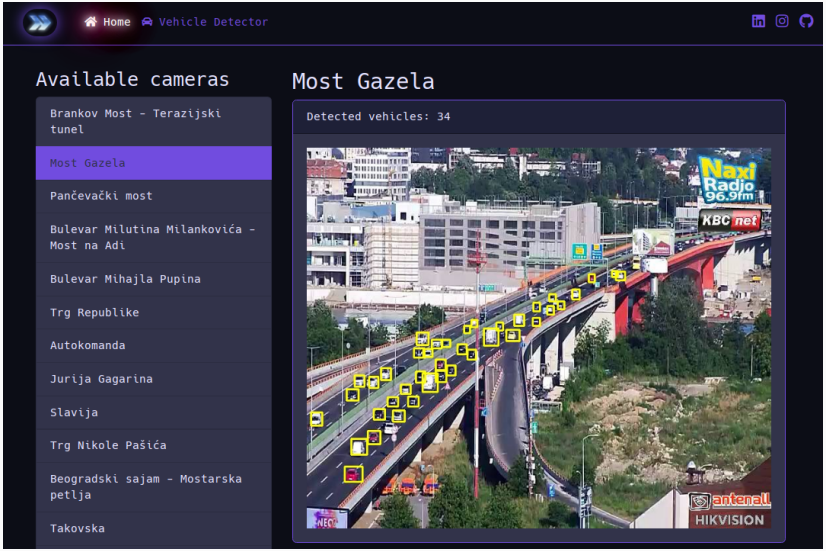
Model 2 - Vukov Spomenik (not seen during training)



Web app demo

- A small web app demo was created to follow how model works
- Library flask was used for serving the model
- Video is available here

Web app demo


Home
Vehicle Detector
Available cameras
Brankov Most - Terazijski tunel
Most Gazela
Pančevački most
Bulevar Milutina Milankovića - Most na Adi
Bulevar Mihajla Pupina
Trg Republike
Autokomanda
Jurija Gagarina
Slavija
Trg Nikole Pašića
Beogradski sajam - Mostarska petlja
Takovska
Most Gazela
Detected vehicles: 34
Naxi Radio 96.9fm
KBC net
antenall
HIKVISION

Learnings

- Works rather ok for such a small data set
- Trains fast on RTX 2070, took around 30 minutes for 50 epochs
- Don't forget, transfer learning from COCO was used
- Night was avoided as it required labeling data, more work with evaluating etc.
- Rain, snow and storm would probably break model to some degree - it doesn't know what those things are!
- Some artifacts would get detected as vehicles - this is ok, small data set, and unseen cameras
- It doesn't work in real time - this was known to happen when decision to use this model was made

What next

- Go real-time object detection!
  - Yolo v4 (Bochkovskiy, Wang, and Liao 2020)
  - Single Shot Detector (Liu et al. 2015) with Mobile Net v2 (Sandler et al. 2018)
- Label data on all cameras
- Label data covering situations like:
  - rain, snow, storm, fog
  - massive traffic jams
  - empty streets
- Deploy as a service

Vehicle Tracking

Computer Vision: Object tracking

- Important problem in the field of computer vision
- Generally divided into:
  - Single object tracking
  - Multiple object tracking
- Problems to solve:
  - Detecting objects to track
  - How often object detection algorithm is executed
  - How to handle object disappearing for a few frames
- This sub-field of computer vision is still knew to me
- I just barely started this problem for this project
- So today I present only a simple solution that uses *Centroid based* object tracking

Vehicle Tracking (to remind you)

Traffic situation tracking problem

Given a sequence of video frames, detect vehicles and track their movement (assign them IDs).

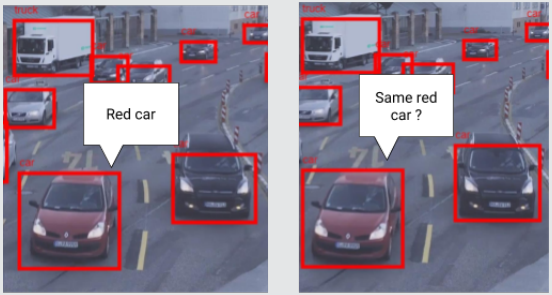


Figure 24: Vehicle tracking illustration, taken from `move-lab.com`

Centroid based object tracking

About algorithm

- Based on pyimagesearch article
- Assigns an id to every new centroid it appears (unless it's bounding box flickering)
- Tries to match closest centroids between adjacent frames using Euclid distance
- When object disappears after k frames, it removes it from tracking
- Very easy to implement and runs fast

Centroid based object tracking

Pros

- Actually works rather well!
- Problem of objects going *through* each other doesn't really happen in my use case
  - Because it would cause a traffic accident!
  - So vehicles (and their bounding boxes) actually have nice distance between them
- Simple to implement
- Works fast (real time)

Cons

- Requires calling object detector **every frame**
  - Problem if this detector can't work in real time
- Assumes that object **didn't move much** between subsequent frames
- It doesn't take **speed vector** into consideration
- It doesn't assume where the object will move
- It doesn't handle overlapping and occlusion

Centroid based object tracking - Results

Some videos:

- Brankov most
- Most Gazela
- Pančevački most
- Bulevar Mihajla Pupina
- Autokomanda
- Slavjia

Further work

Further work

- Create a complete (multiple weather and light conditions) data set
- Use an object detector which can work in real time
- Improve object tracking (try out SotA methods)
- Add support to count vehicles
- Add support to save and visualize trajectories of vehicles
- Integrate all into a web application - SHIP IT!
- Gather data from system in production - be able to analyze it later!

Vehicle counter

Figure 25: Vehicle counter, user would draw the lines - proof of concept

Lane detector

Figure 26: Lane detector - proof of concept

Thanks for your attention!

Questions?

References i

Bochkovskiy, Alexey, Chien-Yao Wang, and Hong-Yuan Mark Liao. 2020. "YOLOv4: Optimal Speed and Accuracy of Object Detection." *Arxiv*. https://arxiv.org/abs/2004.10934.

Girshick, Ross B. 2015. "Fast R-CNN." *CoRR* abs/1504.08083. http://arxiv.org/abs/1504.08083.

Girshick, Ross B., Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2013. "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation." *CoRR* abs/1311.2524. http://arxiv.org/abs/1311.2524.

He, Kaiming, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. 2017. "Mask R-CNN." *CoRR* abs/1703.06870. http://arxiv.org/abs/1703.06870.

References ii

Liu, Wei, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. 2015. "SSD: Single Shot Multibox Detector." *CoRR* abs/1512.02325. http://arxiv.org/abs/1512.02325.

Ren, Shaoqing, Kaiming He, Ross B. Girshick, and Jian Sun. 2015. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks." *CoRR* abs/1506.01497. http://arxiv.org/abs/1506.01497.

Sandler, Mark, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. "Inverted Residuals and Linear Bottlenecks: Mobile Networks for Classification, Detection and Segmentation." *CoRR* abs/1801.04381. http://arxiv.org/abs/1801.04381.