

Járművek trajektóriáinak előrejelzése machine learning modellekkel

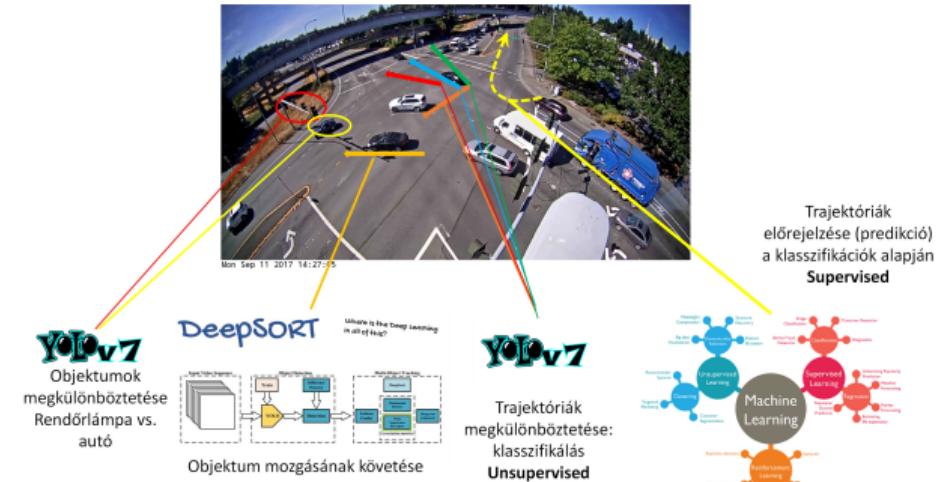
Szerző: Péter Bence Gábor

Témavezető: Dr. Horváth András

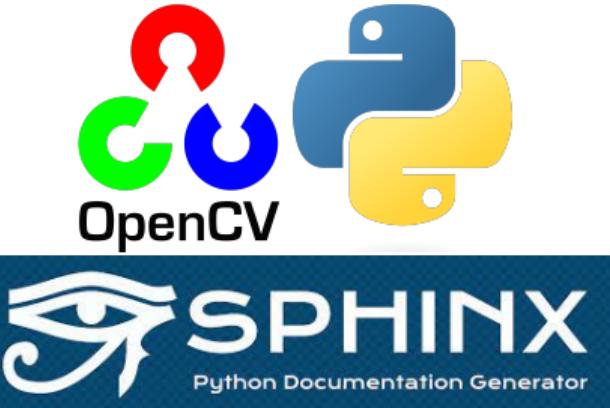
Széchenyi István Egyetem
Gépész, Informatikai és Villamosmérnöki Kar

Motiváció

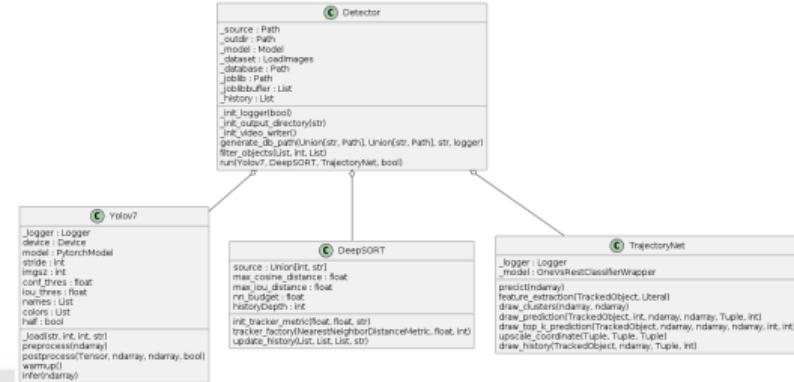
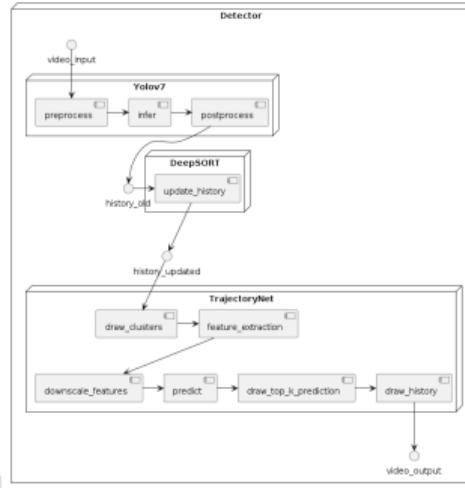
- ▶ Közlekedési csomópontok elemzése
 - ▶ Statisztikák készítése
 - ▶ Anomáliák detektálása
- ▶ Közlekedés irányítás optimalizálása
 - ▶ Közlekedési balesetek megelőzése
 - ▶ Forgalmi dugók csökkentése
 - ▶ Forgalom folyásának előrejelzése



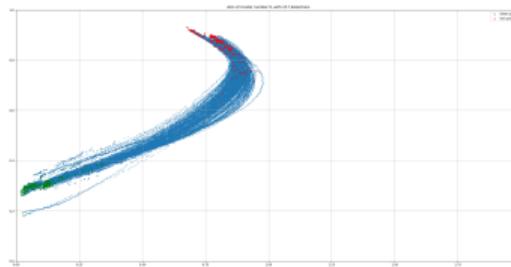
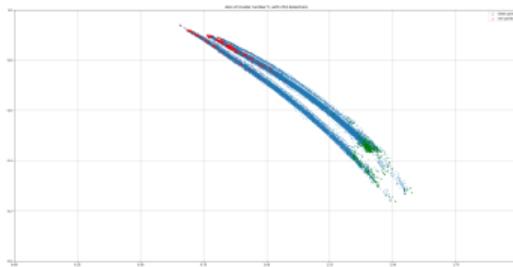
- ▶ YOLOv7: You Only Look Once v7
- ▶ DeepSORT: Simple Online and Realtime Tracking
- ▶ Klaszterezés: OPTICS
- ▶ Klasszifikáció: Support Vector Machine, K-NearestNeighbors, DecisionTree
- ▶ Adattárolás: SQLite, Joblib
- ▶ Megjelenítés: OpenCV
- ▶ Linux
- ▶ Python, LaTeX, Bash
- ▶ Codium, VSCode
- ▶ Git
- ▶ Sphynx



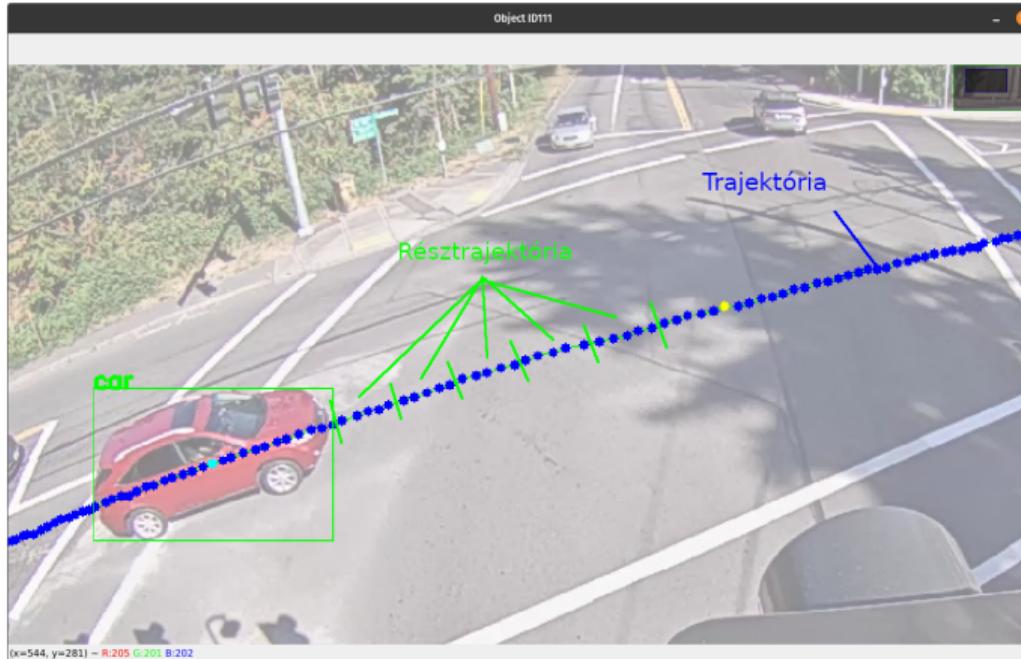
- ▶ Yolo által detektált objektumok valós idejű követése
- ▶ DeepSORT algoritmus használata
- ▶ Trajektóriák tárolása
 - ▶ Python osztályok
 - ▶ Trajektóriák adatbázisba mentése
 - ▶ Trajektóriák betöltése adatbázisból
 - ▶ Python osztályok szerializálása: Pickle, Joblib



- ▶ Feature Engineering: [bemeneti x,y koordináta, kimeneti x,y koordináta]
- ▶ Algoritmus választás: OPTICS, DBSCAN, BIRCH, KMeans
- ▶ OPTICS: Order Points To Identify Clustering Structure
 - ▶ Adatok előfeldolgozása: Zaj szűrése, Értékek skálázása
 - ▶ Hyperparaméterek finomhangolása: min-samples, max-eps, xi



- ▶ Megfigyelt autó besorolása a klaszterezés alapján meghatározott csoportba.
 - ▶ $v1: [x_0, y_0, v_{x_0}, v_{y_0}, x_{n/2}, y_{n/2}, x_n, y_n, v_{x_n}, v_{y_n}]$
 - ▶ $v7: [x_0, y_0, v_{x_0} \cdot 100, v_{y_0} \cdot 100, x_n \cdot 2, y_n \cdot 2, v_{x_n} \cdot 200, v_{y_n} \cdot 200]$

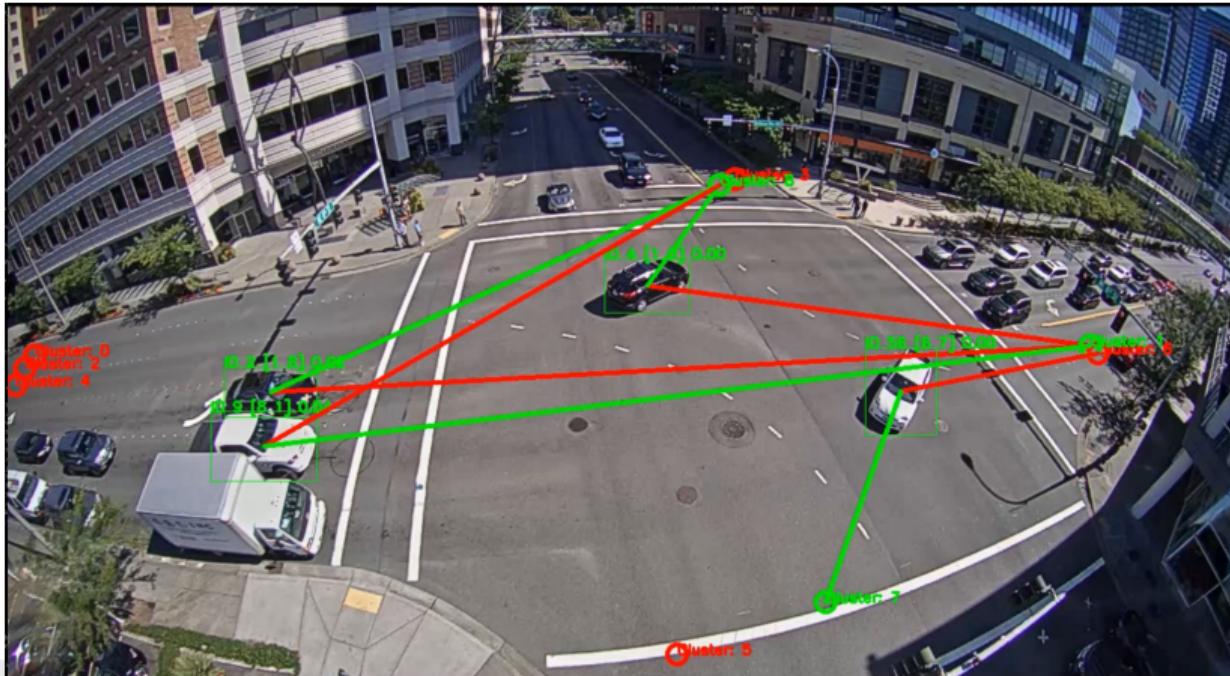


- ▶ K-NearestNeighbors, SupportVectorMachine, DecisionTree
- ▶ A legnagyobb adathalmazon (Bellevue Newport) tanított modellek pontossága
- ▶ Top1: legvalószínűbb előrejelzés tényleg az elvárt osztály
- ▶ Top2: 2 legvalószínűbb előrejelzésben benne van az elvárt osztály
- ▶ Balanced Accuracy: $\frac{\sum_{i=1}^n tp}{n} / (tp+fn)$

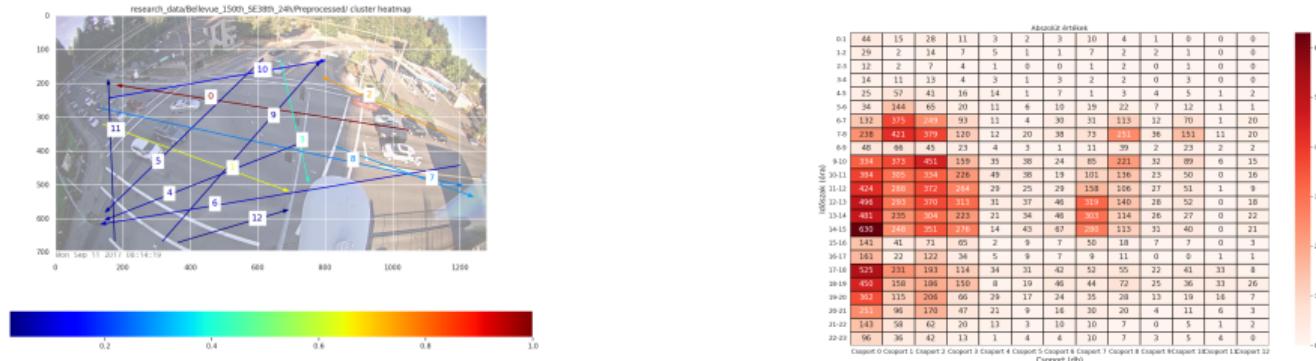
	Top-1	Top-2	Top-3	Balanced		Top-1	Top-2	Top-3	Balanced
SVM	84.06%	96.45%	99.82%	74.45%	SVM	82.04%	96.38%	98.82%	73.42%
KNN	93.27%	98.59%	98.93%	88.09%	KNN	82.64%	97.27%	98.33%	79.07%
DT	89.24%	91.88%	92.08%	86.93%	DT	77.02%	80.17%	81.12%	71.72%

Grafikus megjelenítés

- ▶ Saját fejlesztésű megjelenítő alkalmazás
- ▶ Zöld: legvalószínűbb előrejelzés
- ▶ Piros: második legvalószínűbb előrejelzés

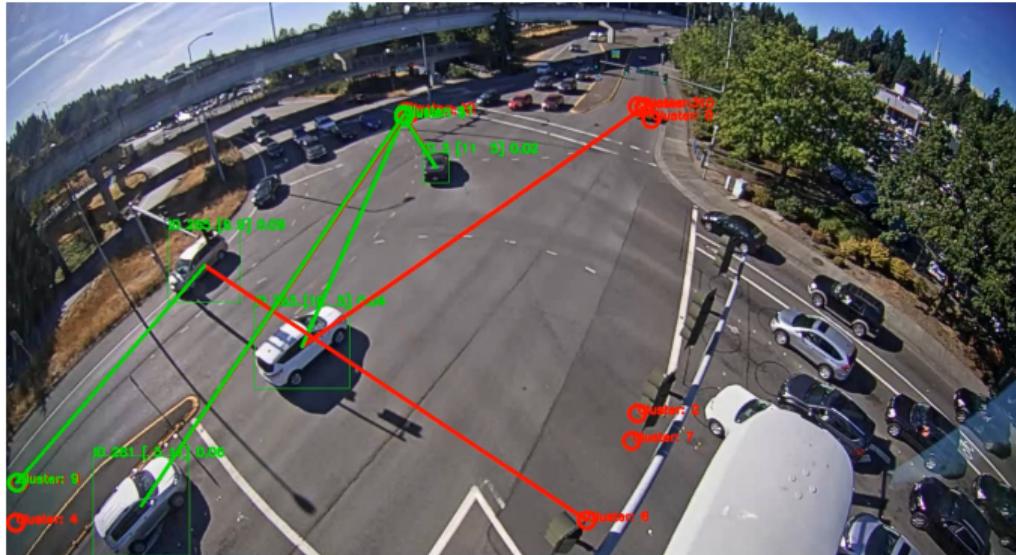


- ▶ Forgalmi statisztikai kimutatások készítése
 - ▶ Forgalom számláló modul implementálása
 - ▶ Forgalmi statisztikák készítése: óránkénti és csoportonkénti felbontásban
 - ▶ Cikk publikálása: **A. Horváth Á. Agg B. G. Péter.** "Automata forgalmi statisztika objektumdetektálás és adaptív járműtrajektória klaszterezés alapján". *Közlekedés és Mobilitás 2:1 (2013)*. https://github.com/Pecneb/computer_vision_research
- ▶ Osztályozási módszerek összehasonlítása
- ▶ Valós idejű detektálás, követés és osztályozás szoftveres megvalósítása



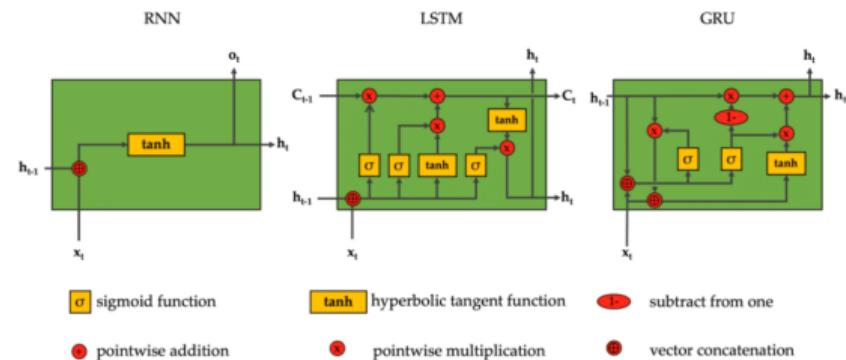
- ▶ Tobábbi feature engineering módszerek kipróbálása
- ▶ Paraméterek finomhangolása GridSearch segítségével
- ▶ Mély tanulás alkalmazása
 - ▶ LSTM, Transofmer modellek kipróbálása
- ▶ További keretrendszerek integrálása
 - ▶ Pytorch, Keras, Jax, Triton
- ▶ Saját keretrendszer optimalizálása
 - ▶ Rust modulok implementálása Candle keretrendszerrel

Köszönöm a figyelmet!



1. Kérdés - Miért machine learning módszereket alkalmazott az osztályozáshoz és a klaszterezéshez? Tervezi továbbiakban neurális hálózat alapú módszerek alkalmazását a machine learning eljárások helyett?

- ▶ Scikit-Learn ismerete
- ▶ Rendelkezésre álló adathalmaz mérete
- ▶ YOLOv7 deep learning modell
- ▶ Továbbfejlesztési irányok 10



2. Kérdés - Tesztelt más objektumkövető módszereket is? Milyen eredményeket sikerült elérni azokkal? Milyen szempontok miatt választotta a DeepSORT-t?

- ▶ Koordináta távolság alapú követés
- ▶ Eltűnő majd felbukkanó objektumok követési problémája
- ▶ Adathalmazba sok zajos trajektória került
- ▶ DeepSORT pontosabb eredményeket adott

```
from computer_vision_research.dataManagementClasses import TrackedObject
21
22
23
24 def calcDist(prev, act):
25     """Function to calculate distance between an object on previous frame and actual frame"""
26     Args:
27         prev (Detection): Object from previous frame
28         act (Detection): Object from actual frame
29
30     Returns:
31         xDist, yDist: distance of x coordinates and distance of y coordinates
32         ...
33     xDist = abs(prev.X - act.X)
34     yDist = abs(prev.Y - act.Y)
35     return xDist, yDist
36
37 def updateHistory(detections, history, frameNumber, historyDepth=3, disThresh=0.05):
38     """Function to update detection history
39
40     Args:
41         detections (list[Detection]): a list of new detection
42         history (list[TrackedObject]): the tracking history
43         frameNumber (int): number of the current video frame
44         historyDepth (int): length of the history to be stored
45         thresh (float, optional): Threshold to be able to tell if next obj is already detected or is a new one. Defaults to 0.1.
46
47     for next in detections:
48         added = False
49         for objHistory in history:
50             try:
51                 prev = objHistory.history[-historyDepth]
52             except:
53                 prev = objHistory.history[-1]
54             xDist, yDist = calcDist(prev, next)
55             if (xDist < (prev.X * disThresh)) and (yDist < (prev.Y * disThresh)) and objHistory.label == next.label:
56                 objHistory.history.append(next)
57                 added = True
58             # the threshold for the non moving objects is still hardcoded
59             # TODO: find a good way to tell what objects are still or in motion
60             if (xDist > (disThresh * prev.X * 0.25)) or (yDist > (disThresh * prev.Y * 0.25)):
61                 objHistory.isMoving = True
62             else:
63                 print("ObjID: {} with xDist: {} and yDist: {} is not moving".format(objHistory.objID, xDist, yDist))
64                 objHistory.isMoving = False
65
66             # remove objects that are older than frameNumber-historyDepth
67             if objHistory.history[-1].frameID < (frameNumber - historyDepth):
68                 try:
69                     print("ID {} object is removed from history".format(objHistory.objID))
70                     history.remove(objHistory)
71                 except:
72                     continue
73
74             if not added:
75                 history.append(TrackedObject(len(history)+1, next))
```

3. Kérdés - A létrehozott módszer alkalmas lehet közlekedési anomáliák észlelésére? Pl.: balesetek, rendellenesen mozgó járművek. Ha igen, akkor az eljárás mely funkciója tenné ezt lehetővé?

- ▶ Közlekedési anomáliák nem valós idejű detektálása
 - ▶ Klaszterezés: outlier detektálás
 - ▶ Statisztikai kimutatások készítése
- ▶ Közlekedési anomáliák valós idejű detektálása
 - ▶ Osztályozás: valós időben történő osztályozás bizonytalansága

4. Kérdés - Ha az eljárás valós idejű alkalmazásának későbbi alternatívája lenne egy adaptív közlekedési lámpa vezérlés, akkor milyen szoftveres és hardveres környezetet választana ennek megvalósításához?

- ▶ Operációs rendszer: Linux
- ▶ Programozási nyelv: Python
- ▶ Hardver: Nvidia Jetson Orin AGX

Jetson AGX Orin Jetson Orin NX MLPerf v3.0 Results

Model	NVIDIA Jetson AGX Orin (TensorRT)			NVIDIA Orin MaxQ (TensorRT)		NVIDIA Jetson Orin NX
	Single Stream (Samples/s)	Offline (Samples/s)	Multi Stream (Samples/s)	Offline (Samples/s)	System Power(W)	Offline (Samples/s)
Image Classification ResNet-50	1538	6438.10	3686	3525.91	23.06	2517.99
Object Detection Retinanet	51.57	92.40	60.00	34.6	22.4	36.14
Medical Imaging 3D-Unet	.26	.51	N/A	3.28	28.64	.19
Speech-to-text RNN-T	9.822	1170.23	N/A	14472	25.64	405.27
Natural Language Processing BERT	144.36	544.24	N/A	3685.36	25.91	163.57