Universitatea Națională de Știință și Tehnologie POLITEHNICA
București

Sistem de monitorizare a traficului cu raportare pe platforma web

Realizat de Sava Rareș-Andrei
Grupa 441A

Prof. coordonator: Manciu Irina Petra

Cuprins

P0. Definirea ideii de dezvoltare

Scopul principal al proiectului este dezvoltarea unui sistem inteligent ce monitorizează traficul în timp real, identificând și numărând mașinile care traversează o anumită zonă cuprinsă de o cameră video. Produsul va utiliza fluxuri video live (live streams) provenite de la camere publice ce pot fi accesate online și se va baza pe algoritmi de inteligență artificială pentru a realiza detecția mașinilor. Sistemul va analiza fluxul video, va aplica algoritmul de detecție, va diferenția mașinile în mișcare de cele staționare și va trimite datele colectate la dashbord-ul web accesibil din browser.

P1. Identificarea resurselor necesare

Surse de date:

- Fluxuri video live publice provenite de la camere de supraveghere a traficului (ex: 511ny.org, EarthCam)
- Sistemul este capabil să proceseze atât link-uri directe, cât și pagini web complexe prin intermediul librăriilor de extragere a link-urilor corespunzătoare live stream-urilor

Cerințe Hardware:

- Computer personal (laptop/desktop) pentru dezvoltare și rulare
- Capacitate de procesare AI, de preferat placă video dedicată sau CPU performant, pentru aplicarea modelului YOLO (You Only Look Once) în timp real

Cerințe software:

- Limbaj de programare: Python 3.10+
- `ultralytics` - biblioteca software dezvoltată de compania Ultralytics, care permite utilizarea ușoară a modelelor YOLO în Python
- `YOLOv8n` (You Only Look Once - version 8 nano) - modelul de inteligență artificială propriu-zis, este considerat cel mai performant la momentul actual pentru detecția în timp real, oferind cel mai bun raport viteză/precizie
- `opencv-python` - pentru manipularea matricială a imaginilor, desenarea bounding box-urilor și afișarea ferestrei de debug
- `numpy` - folosit intern de OpenCV pentru reprezentarea frame-urilor ca tablouri multidimensionale.
- `pytz` - pentru gestionarea fusurilor orare (Timezone Aware), esențial pentru sincronizarea orei camerelor internaționale

- `re` (Regular Expressions) - pentru parsing-ul avansat al paginilor HTML, mai exact extragerea titlurilor din pagina corespunzătoare live stream-ului
- `math și sys` - pentru calcule geometrice și gestionarea sistemului
- `streamlink` - bibliotecă esențială pentru gestionarea protocolului HLS (HTTP Live Streaming)
- `requests` - pentru comunicarea HTTP (API calls și HTML fetching)
- `Flask` - server WSGI care gestionează rutele, template-urile Jinja2 și contextul aplicației
- `SQLite` - stocare persistentă "Serverless", ideală pentru prototipare rapidă
- `HTML5, CSS3` (Flexbox Layout, Backdrop Filter pentru efecte de sticlă a chenarelor în interfață)
- `JavaScript` - logica asincronă (`async/await`) pentru preluarea datelor fără reîncărcarea paginii
- `Chart.js` - randarea graficelor HTML5 Canvas reactive

P2. Stabilirea modelului de dezvoltare

S-a ales un model de dezvoltare bazat pe procesarea fluxurilor video în timp real. Spre deosebire de analiza seturilor de date deja existente și, implicit, mai ușor de accesat (fișiere video pre-înregistrate), acest model implică conectarea continuă la o sursă de date în direct. Astfel, modelul se remarcă prin relevanță, întrucât datele reflectă situația actuală a traficului, continuitate prin rularea lui pe perioadă nedeterminată, generând statistici pe termen lung, și prin adaptabilitate, sistemul putând funcționa utilizând orice cameră accesibilă de pe glob.

P3. Analiza SWOT pentru soluția propusă

Strengths:

- Precizie ridicată: utilizarea YOLOv8 oferă o detecție mult mai robustă a vehiculelor comparativ cu metodele clasice de procesare de imagine, cum ar fi background subtraction, fiind imun la schimbări de lumină sau umbre
- Arhitectură decuplată: software-ul de procesare a imaginilor și serverul web sunt componente separate care comunică prin API, permițând rularea lor pe stații diferite, dacă este necesar

- User Experience: dashboard simplu, intuitiv, cu refresh rate de 1 minut și selecție de intervale pentru vizualizarea datelor pe 1, 5, 10, 15 și 30 de minute pentru o experiență plăcută, fără întreruperi și fără intervenții din partea utilizatorului

Weaknesses:

- Resurse computaționale generoase: procesarea AI în timp real necesită resurse hardware semnificative cum ar fi un procesor sau o placă video performante
- Dependență de sursă: dacă camera publică devine indisponibilă sau își schimbă formatul stream-ului, sistemul necesită reconfigurare

Opportunities:

- Integrarea în sisteme de semaforizare inteligentă pentru fluidizarea traficului
- Extinderea detecției pentru alte clase - pietoni, bicicliști - suportate deja de modelul YOLO
- Crearea de alerte automate la depășirea unor valori critice de trafic

Threats:

- Schimbări în politicile de acces ale furnizorilor de stream-uri precum criptarea stream-urilor
- Condiții meteo extreme cum ar fi ceață densă sau viscol care pot bloca vizibilitatea camerei, făcând detecția mașinilor dificilă spre imposibilă

P4. Proiectarea soluției

Arhitectura sistemului este de tip pipeline, unde datele curg unidirecțional de la sursă spre utilizator prin trei module interconectate: modulul de achiziție al datelor, serverul virtual și modulul de vizualizare. Această separare asigură că o problemă într-un modul nu blochează funcționarea celorlalte. De exemplu, dacă achiziția datelor este sistată din cauza unei defecțiuni a camerei, utilizatorul poate vedea în continuare raportările anterioare pe platforma web. De asemenea, dacă partea de frontend este la un moment dat afectată, înregistrarea rapoartelor se realizează în continuare independent, utilizatorul urmând să aibă acces la informații în urma remedierii problemei, acestea fiind stocate în baza de date.

Modulul de achiziție - `procesor_trafic.py`

- Acesta este componenta activă care monitorizează și detectează traficul.
- Funcționare: se comportă ca un observator, preia fluxul video de la camera stradală, frame by frame
- Procesare: folosește Inteligența Artificială (YOLO) pentru a recunoaște formele mașinilor în fiecare cadru video; când o mașină este identificată, o urmărește până când este validată ca fiind un vehicul real în mișcare, făcând diferența între cele staționare și cele în deplasare
- Conexiunea cu restul sistemului: nu stochează nimic local; o dată pe minut, împachetează numărătoarea într-un mesaj digital (JSON) și îl trimite către server

Modulul server și stocarea - `api_server.py`

- Acesta este punctul central de legătură
- Rol de recepție: ascultă permanent mesajele trimise de modulul de achiziție; când primește un raport, îl validează și îl scrie în baza de date
- Rol de arhivare: gestionează jurnalul de trafic, asigurându-se că datele vechi sunt păstrate în siguranță
- Rol de distribuție: când un utilizator deschide site-ul, serverul interoghează baza de date și livrează istoricul cerut

Modulul de vizualizare - frontend-ul

- Aceasta este interfața cu care interacționează utilizatorul
- Interacțiune: rulează în browserul utilizatorului; la deschidere, trimite o cerere de acces la date către server
- Vizualizare: primește datele brute (numere) și le transformă în grafice vizuale intuitive, care permit înțelegerea rapidă a tendințelor de trafic
- Dinamism: își actualizează automat conținutul fără a necesita reîncărcarea paginii, oferind o experiență plăcută, fără întreruperi și fără intervenții din partea utilizatorului

P5. Implementare

Etapa de implementare s-a concentrat pe modularizare și robustețe. Soluțile tehnice adoptate au fost:

- Centralizarea configurării - `config.py`: s-a implementat o arhitectură ce decuplează parametrii variabili URL sursă și timezone de logica backend-ului, facilitând mentenanța fără modificarea codului sursă
- Sincronizare temporală globală: s-a integrat librăria `pytz` pentru a rezolva problema discrepanței dintre ora serverului și ora locației monitorizate, asigurând consistența datelor statistice
- Adaptabilitate dinamică: frontend-ul a fost proiectat să randeze elementele DOM (titlu, grafice) dinamic, pe baza metadatelor primite prin API, eliminând nevoia de hardcodare
- Automatizarea extracției de metadate: s-a dezvoltat un modul de "Web Scraping" în `procesor_trafic.py` care încearcă deducerea automată a contextului (numele locației) din sursele web, îmbunătățind experiența de configurare.