



Univerzitet u Sarajevu  
Elektrotehnički fakultet u Sarajevu  
Odsjek za računarstvo i informatiku



# Standalone App za prepoznavanje saobraćajnih znakova

Digitalno procesiranje signala

PROJEKAT

Studenti:  
Nedim Pojskić 19128  
Hamza Ramić 19147

Mentorica:  
vanr. prof. dr Amila Akagić

Sarajevo, juni 2025.

# Sadržaj

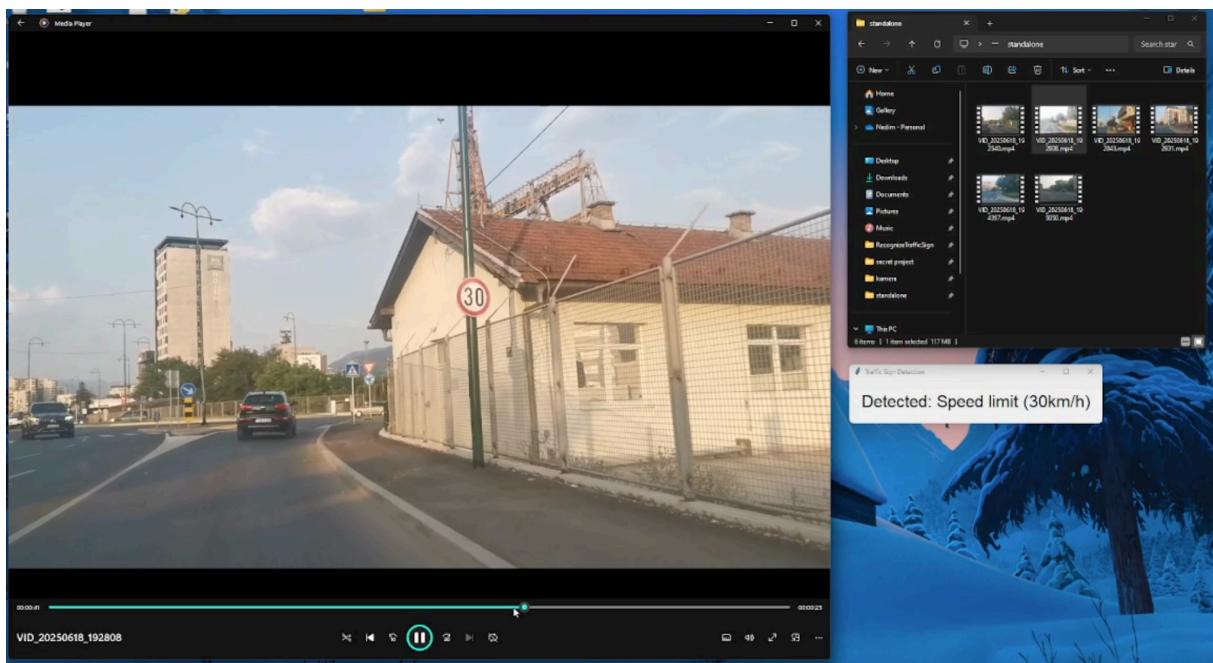
<b>Sadržaj.....</b>	<b>1</b>
<b>Uvod.....</b>	<b>2</b>
<b>Ciljevi projekta.....</b>	<b>3</b>
<b>Tehnologije i alati.....</b>	<b>4</b>
<b>Arhitektura sistema.....</b>	<b>5</b>
Modul za detekciju znakova (YOLO).....	5
Modul za klasifikaciju znakova (ResNet50).....	5
Grafički korisnički interfejs (GUI).....	5
Tok podataka kroz sistem:.....	6
<b>Detekcija i klasifikacija znakova - procesiranje slike.....</b>	<b>7</b>
1. Detekcija (YOLO model).....	7
2. Klasifikacija (ResNet50 model).....	7
<b>Treniranje modela.....</b>	<b>8</b>
YOLO model.....	8
ResNet50 klasifikator (klasa znaka).....	8
<b>Ograničenja i problemi.....</b>	<b>9</b>
<b>Zaključak.....</b>	<b>10</b>

# Uvod

"Standalone App za prepoznavanje saobraćajnih znakova" je projekat čiji je cilj razvoj samostalne desktop aplikacije koja u realnom vremenu detektuje i klasificiše saobraćajne znakove korištenjem savremenih metoda dubokog učenja. Ovaj sistem predstavlja osnovu za inteligentne transportne tehnologije i može poslužiti kao funkcionalna komponenta u naprednim sistemima pomoći vozaču (ADAS).

Za detekciju znakova koristi se YOLO (You Only Look Once) model, treniran da prepozna pozicije saobraćajnih znakova na video snimcima. Nakon detekcije, klasifikacija se vrši pomoći konvolucione neuronske mreže (CNN) bazirane na ResNet50 arhitekturi, trenirane na skupu od 43 klase saobraćajnih znakova.

Aplikacija je razvijena u programskom jeziku Python i koristi biblioteke PyTorch, OpenCV, Ultralytics YOLO te Tkinter za grafički interfejs. Sistem funkcioniše potpuno lokalno, bez potrebe za internet konekcijom, što omogućava njegovu primjenu u stvarnom vremenu i na uređajima s ograničenim resursima.



Slika [1] - Prikaz rada aplikacije na pre-recorded videu

## Ciljevi projekta

Cilj projekta "Standalone App za prepoznavanje saobraćajnih znakova" jeste razviti lokalnu aplikaciju koja omogućava:

- Detekciju saobraćajnih znakova u stvarnom vremenu koristeći kameru.
- Klasifikaciju detektovanih znakova u jednu od unaprijed definisanih 43 klase.
- Prikaz rezultata korisniku putem jednostavnog i intuitivnog grafičkog interfejsa.
- Fokus na ključne znakove kao što su: ograničenja brzine, stop i znak „daj prednost“.
- Rad bez internet konekcije i bez potrebe za vanjskim servisima — potpuno lokalno izvršavanje.
- Jednostavno pokretanje i upotrebu, što čini aplikaciju pogodnom za edukativne svrhe, testiranje AI rješenja ili demonstraciju u embedded/IoT sistemima

## Tehnologije i alati

U razvoju aplikacije korišten je set modernih alata i biblioteka iz oblasti računarske vizije i deep learning-a. Ključne tehnologije uključuju:

**Python** – glavni programski jezik korišten za implementaciju svih komponenti aplikacije zbog svoje jednostavnosti i bogatog ekosistema biblioteka za obradu slike i mašinsko učenje.

**PyTorch** – biblioteka za deep learning korištena za treniranje i izvođenje klasifikacionog modela (ResNet50).

**Ultralytics YOLO (You Only Look Once)** – model za detekciju objekata, korišten za lociranje saobraćajnih znakova na slici ili videu.

**Torchvision** – dodatna biblioteka uz PyTorch, korištena za rad s predefinisanim arhitekturama (kao što je ResNet) i transformacije slike.

**OpenCV** – biblioteka za obradu slike i rad sa video strimovima iz kamere.

**Tkinter** – standardna Python biblioteka za izradu jednostavnog grafičkog korisničkog interfejsa.

**PIL (Pillow)** – biblioteka za rad sa slikama (učitavanje, konverzija i manipulacija formatima).

**Threading (Python threading modul)** – omogućava paralelno izvršavanje video obrade i GUI-ja bez blokiranja interfejsa.

# Arhitektura sistema

Aplikacija je modularno strukturirana i sastoji se od tri glavne funkcionalne komponente:

## Modul za detekciju znakova (YOLO)

- Ulaz: Slika iz video strima kamere.
- Funkcija: Identificuje pozicije (bounding box) potencijalnih saobraćajnih znakova na slici.
- Tehnologija: YOL0v8 model treniran i sačuvan kao `bestDetectTrafficSign.pt`.
- Rezultat: Koordinate detektovanih znakova koje se proslijeđuju klasifikatoru.

## Modul za klasifikaciju znakova (ResNet50)

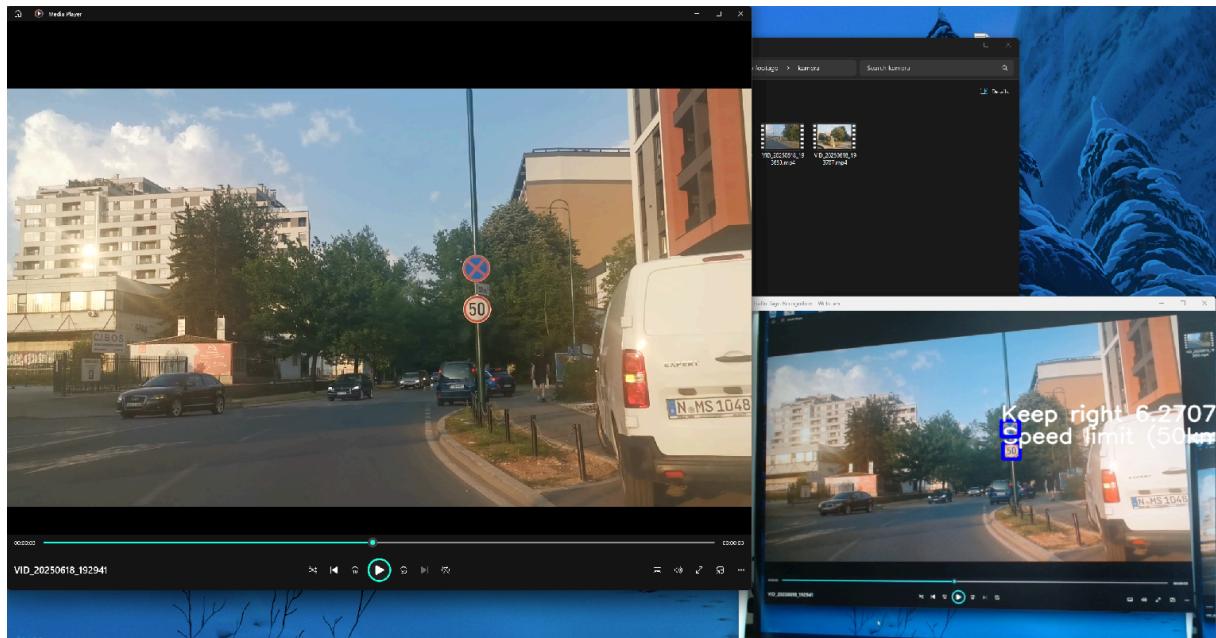
- Ulaz: Izdvojeni region slike (crop znaka).
- Funkcija: Prepoznaje tačnu vrstu znaka iz 43 definisane klase.
- Tehnologija: ResNet50 model treniran u PyTorch-u, učitan iz fajla `checkpoint224x224_3epoch.pth`.
- Rezultat: Tekstualna oznaka klase (npr. „Speed limit (50km/h)“, „Stop“, itd.).

## Grafički korisnički interfejs (GUI)

- Tehnologija: Tkinter (Python standardna biblioteka).
- Funkcija: Prikazuje korisniku rezultat klasifikacije u realnom vremenu.
- Dodatno: Koristi threading kako bi GUI ostao responzivan dok se video obrada paralelno izvršava.

## Tok podataka kroz sistem:

Kamera šalje sliku → YOLO model detektuje znak(ove) → Svaki znak se „cropuje“ i šalje ResNet modelu → Dobijena klasa se prikazuje u GUI-ju ako pripada ciljanim znakovima (Speed limit, Stop, Yield).



Slika [2] - Prepoznavanje znakova u real time-u

# Detekcija i klasifikacija znakova - procesiranje slike

Proces detekcije i prepoznavanja saobraćajnog znaka odvija se u dva koraka: **detekcija regiona znaka (gdje se znak nalazi na slici)** i **klasifikacija tog regiona (koji je to tačno znak)**.

## 1. Detekcija (YOLO model)

Kada se slika preuzeće sa kamere (putem `OpenCV`), ona se šalje u YOLO model (`bestDetectTrafficSign.pt`), koji je prethodno treniran da prepoznae saobraćajne znakove. YOLO vraća **bounding box** koordinate za svaki prepoznati znak u slici: (`x1, y1, x2, y2`). Te koordinate označavaju **pravougaoni region slike** unutar kojeg se nalazi znak.

```
x1, y1, x2, y2 = map(int, box)
sign_crop = frame[y1:y2, x1:x2]
```

Na ovaj način se izreže dio slike koji sadrži samo saobraćajni znak.

## 2. Klasifikacija (ResNet50 model)

Taj izrezani znak (`sign_crop`) se potom snima kao privremena slika (`temp_sign.jpg`) i šalje u klasifikator. Prije klasifikacije, ta slika prolazi kroz niz koraka poznatih kao **preprocessing**:

```
transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
])
```

Slika se skalira na dimenzije koje model očekuje (224x224). Potom se pretvara u tenzor i normalizuje (vrijednosti piksela se skaliraju). Zatim model predviđa klasu pomoću izlaznog vektora dimenzije 43 (za 43 klase). Na kraju se uzima indeks klase sa najvećom vjerovatnoćom (`torch.max(output, 1)`).

Rezultat je ime znaka koje odgovara indeksu iz `TabTrafficSign` liste.

# Treniranje modela

## YOLO model

Korišten je `bestDetectTrafficSign.pt` koji je posuđen iz projekta sa ovog [linka](#).

## ResNet50 klasifikator (klasa znaka)

Za klasifikaciju detektovanih znakova korišten je ResNet50 model, prenaučen (fine-tuned) na [GTSRB dataset](#) (German Traffic Sign Recognition Benchmark), koji sadrži 43 klase saobraćajnih znakova.

- **Ulazne dimenzije slike:** 224x224
- **Batch size:** 32
- **Broj epoha:** 3
- **Optimizator:** Adam
- **Gubitak (loss):** CrossEntropyLoss
- **Tačnost postignuta na validaciji:** 98.87%

Model je sačuvan kao `checkpoint224x224_3epoch.pth`.

Output treniranja modela:

No. epochs: 1,	Training Loss: 2.058	Valid Loss: 0.914	Valid Accuracy: 0.746
No. epochs: 1,	Training Loss: 2.472	Valid Loss: 0.286	Valid Accuracy: 0.912
No. epochs: 1,	Training Loss: 2.628	Valid Loss: 0.265	Valid Accuracy: 0.919
No. epochs: 1,	Training Loss: 2.735	Valid Loss: 0.127	Valid Accuracy: 0.957
No. epochs: 1,	Training Loss: 2.827	Valid Loss: 0.102	Valid Accuracy: 0.966
No. epochs: 2,	Training Loss: 0.03	Valid Loss: 0.053	Valid Accuracy: 0.982
No. epochs: 2,	Training Loss: 0.076	Valid Loss: 0.045	Valid Accuracy: 0.986
No. epochs: 2,	Training Loss: 0.115	Valid Loss: 0.042	Valid Accuracy: 0.987
No. epochs: 2,	Training Loss: 0.152	Valid Loss: 0.037	Valid Accuracy: 0.989
No. epochs: 2,	Training Loss: 0.188	Valid Loss: 0.039	Valid Accuracy: 0.988
No. epochs: 3,	Training Loss: 0.01	Valid Loss: 0.039	Valid Accuracy: 0.989
No. epochs: 3,	Training Loss: 0.042	Valid Loss: 0.038	Valid Accuracy: 0.988
No. epochs: 3,	Training Loss: 0.075	Valid Loss: 0.039	Valid Accuracy: 0.988
No. epochs: 3,	Training Loss: 0.111	Valid Loss: 0.038	Valid Accuracy: 0.988
No. epochs: 3,	Training Loss: 0.143	Valid Loss: 0.038	Valid Accuracy: 0.988
No. epochs: 3,	Training Loss: 0.177	Valid Loss: 0.037	Valid Accuracy: 0.989

Test accuracy of model: 98.87%

# Ograničenja i problemi

Tokom razvoja sistema prepoznata su određena ograničenja i izazovi:

**Ograničena klasa detekcije:** YOLO model je treniran samo za određene znakove, pa može ignorisati ostale.

**Preciznost na realnim podacima:** Modeli pokazuju dobre rezultate na standardnim slikama, ali performanse mogu opasti u uslovima kao što su:

- loše osvjetljenje,
- zamućenost slike,
- djelimično zaklonjeni znakovi.

**Lažne detekcije:** Povremeno se javljaju lažno pozitivne detekcije (false positives), naročito na pozadinama koje podsjećaju na znakove. (Slika 2 prikazuje taj problem).

**Performanse u realnom vremenu:** Na računarima sa slabijim CPU-om može doći do kašnjenja u prikazu detekcija.

**Privremeno snimanje slike:** Trenutno se svaki znak snima kao fajl `temp_sign.jpg`, što nije optimalno za brzinu sistema (moguće rješenje: direktna obrada iz OpenCV matrice bez snimanja na disk).

## Zaključak

U okviru projekta "**Standalone App za prepoznavanje saobraćajnih znakova**" razvijen je funkcionalan sistem koji u realnom vremenu prepoznaje ključne saobraćajne znakove koristeći kombinaciju dubokih neuronskih mreža — YOLO za detekciju i ResNet50 za klasifikaciju.

Sistem funkcioniše lokalno i ne zahtijeva internet konekciju, što ga čini pogodnim za primjenu u embedded sistemima i edukativne svrhe. Iako postoje ograničenja u detekciji pri složenim uslovima okoline, rezultati na testnim podacima i uživo pokazuju da sistem pouzdano prepozna znakove kao što su "**Stop**", "**Yield**" i **ograničenja brzine**.

Ovaj projekat demonstrira praktičnu primjenu konvolucijskih neuronskih mreža i može poslužiti kao osnova za dalje proširenje, uključujući mobilne verzije, dodatne klase znakova ili integraciju sa drugim ADAS komponentama.