PiTrafficEye

-SISTEME INCORPORATE-

Proiect realizat de:

Oxani Corina,
Pastore Helena,
Nistor Carina

An de studiu: III, CTI-RO, sgr. 5.1

Link github: https://github.com/CorinaOxani/Proiect_SI

I. Sistem automat de procesare de imagini pentru recunoașterea video a semnelor de circulație sau a culorilor de la semafor, folosind Raspberry PI și o cameră video atașată.

Explicarea Funcționalităților Sistemului:

Problema rezolvată: Sistemul creat are scopul de a demonstra capacitatea de recunoaștere automată a semnelor de circulație și a culorilor semaforului, utilizând un microcontroller Raspberry Pi și o cameră video. Acest lucru este esențial pentru dezvoltarea tehnologiilor de conducere autonomă și sisteme avansate de asistență a șoferului.

Modalități de rezolvare:

Capturarea video: Camera atașată la Raspberry Pi capturează fluxul video din mediul înconjurător.

Procesarea imaginii: Algoritmul implementat în Python analizează fiecare cadru video în timp real pentru a detecta și recunoaște semnele de circulație și culorile semaforului.

Afișarea rezultatelor: Pentru fiecare semn sau culoare detectată, sistemul afișează un mesaj corespunzător, informând utilizatorul despre identificarea corectă a semnului sau culorii.

II. Raspberry Pi 4: Caracteristici și Detalii Tehnice

Raspberry Pi 4 este o placă de dezvoltare puternică și versatilă, proiectată pentru a susține o gamă largă de proiecte, de la aplicații educaționale la sisteme embedded și soluții IoT. Iată o descriere detaliată a caracteristicilor sale și a microprocesorului ARM utilizat:

Caracteristici Generale

Procesor: Broadcom BCM2711, un SoC (System on Chip) care include un CPU Quad-Core Cortex-A72 (ARM v8) pe 64 de biți, la 1.5 GHz.

Memorie RAM: Disponibil în mai multe configurații:

- 2 GB LPDDR4-3200 SDRAM
- 4 GB LPDDR4-3200 SDRAM
- 8 GB LPDDR4-3200 SDRAM

GPU: VideoCore VI, capabil să decodeze video H.265 (HEVC) la 4Kp60, H.264 la 1080p60 și să suporte OpenGL ES 3.0.

Stocare: MicroSD pentru stocare principală și boot.

Conectivitate:

- Gigabit Ethernet
- Dual-band 802.11ac wireless (Wi-Fi 5)
- Bluetooth 5.0

Interfețe I/O:

- 2 x micro HDMI cu suport pentru până la 4Kp60
- 2 x USB 3.0 și 2 x USB 2.0
- GPIO cu 40 de pini compatibili cu versiunile anterioare

Alimentare: Port USB-C pentru alimentare, necesită un adaptor de 5V/3A.

Microprocesorul ARM: Broadcom BCM2711

Schema Bloc a Microprocesorului

CPU:

- Quad-Core ARM Cortex-A72
- Performanță ridicată, eficiență energetică bună
- Suport pentru arhitectura ARMv8-A

GPU: VideoCore VI

- Suportă decodare hardware pentru formate video moderne
- Performanțe grafice adecvate pentru aplicații multimedia

Memorie: Interfață LPDDR4 SDRAM

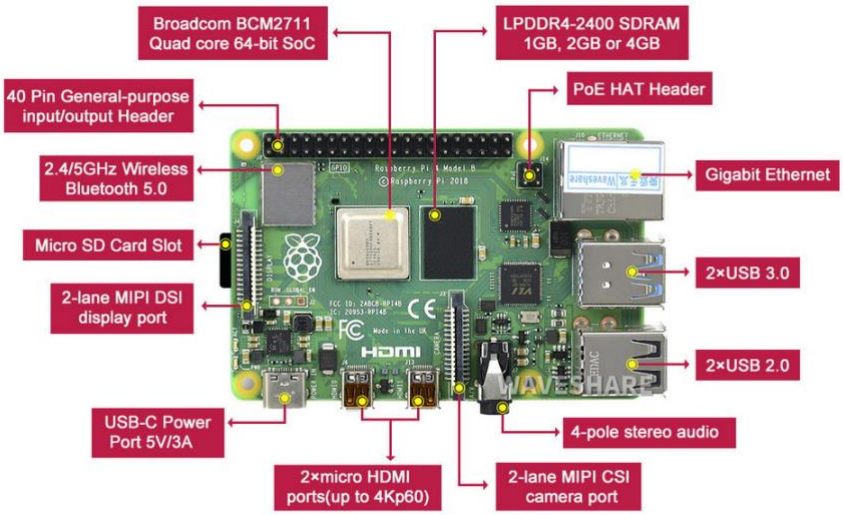
- Performanțe rapide de memorie
- Configurații multiple pentru flexibilitate în aplicații diverse

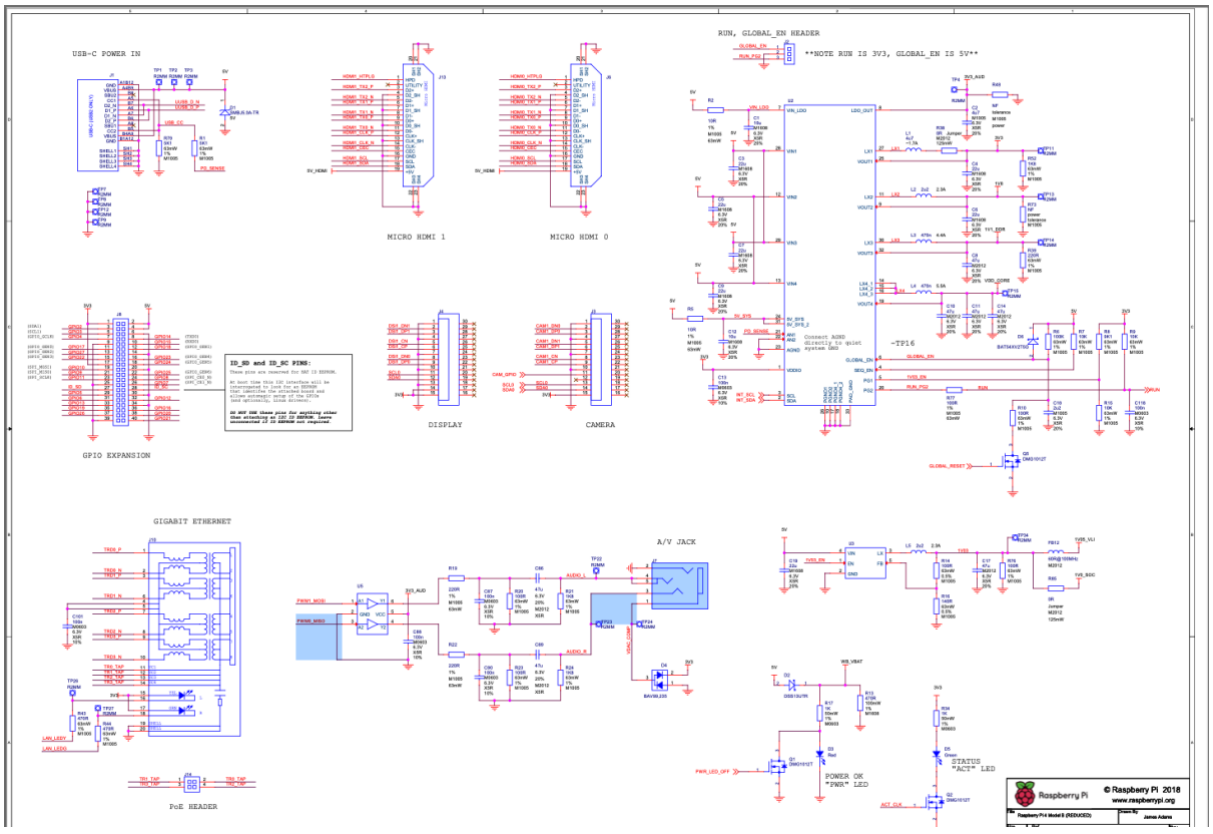
Interfețe de Comunicare:

- USB 3.0 și 2.0 pentru conectivitate rapidă cu periferice
- Gigabit Ethernet pentru rețea rapidă și fiabilă
- Interfețe wireless pentru conectivitate fără fir

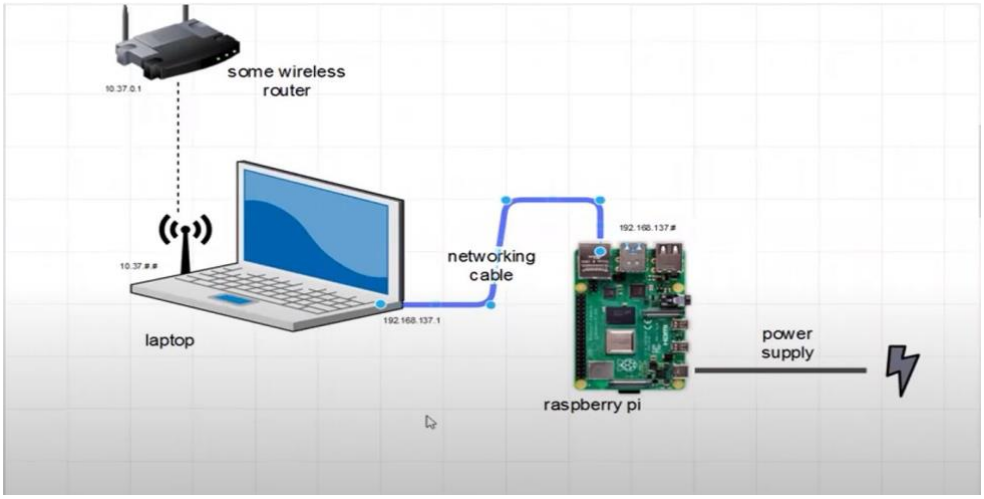
GPIO și alte I/O:

- 40 de pini GPIO pentru extensii și conectivitate cu alte dispozitive
- Suport pentru diverse protocoale de comunicație (SPI, I2C, UART)


Broadcom BCM2711
Quad core 64-bit SoC
LPDDR4-2400 SDRAM
1GB, 2GB or 4GB
40 Pin General-purpose
input/output Header
PoE HAT Header
2.4/5GHz Wireless
Bluetooth 5.0
Gigabit Ethernet
Micro SD Card Slot
2×USB 3.0
2-lane MIPI DSI
display port
2×USB 2.0
USB-C Power
Port 5V/3A
4-pole stereo audio
2×micro HDMI
ports(up to 4Kp60)
2-lane MIPI CSI
camera port


USB-C POWER IN
MICRO HDMI 1
MICRO HDMI 0
RUN, GLOBAL_EN HEADER
NOTE RUN IS 3V3, GLOBAL_EN IS 5V
GPIO EXPANSION
DISPLAY
CAMERA
GIGABIT ETHERNET
A/V JACK
PoE HEADER
STATUS
"ACT" LED
POWER OK
"PWR" LED
© Raspberry Pi 2018

III.Arhitectura sistemului:



IV.Programe, comentarii, descriere

Descriere Generală

Acest proiect este implementat pe un Raspberry Pi 4 echipat cu o cameră și folosește o rețea neuronală convoluțională (CNN) pentru a recunoaște semnele de circulație în timp real. Proiectul implică mai multe etape: preprocesarea datelor, antrenarea modelului și recunoașterea semnelor de circulație în timp real.

Biblioteci și Module Folosite

NumPy: pentru manipularea array-urilor.

Pandas: pentru manipularea datelor.

Matplotlib: pentru vizualizarea rezultatelor antrenamentului.

TensorFlow/Keras: pentru construirea și antrenarea modelului de rețea neuronală.

PIL (Python Imaging Library): pentru preprocesarea imaginilor.

OpenCV: pentru capturarea și procesarea imaginilor în timp real.

scikit-learn: pentru împărțirea seturilor de date și evaluarea modelului.

Structura Proiectului
1.Preprocesarea Datelor și Antrenarea Modelului

traffic.py

Acest script încarcă și preprocesează datele de antrenament, definește arhitectura modelului CNN, antrenează modelul și evaluează performanța acestuia pe un set de date de test. Modelul antrenat este apoi salvat pentru utilizare ulterioară. (salvat in traffic_classifier.h5)

2. Recunoașterea Semnelor de Circulație în Timp Real

detect.py

Acest script încarcă modelul antrenat și utilizează camera conectată la Raspberry Pi pentru a captura imagini în timp real. Imaginile sunt preprocesate și clasificate folosind modelul CNN. Rezultatele sunt afișate pe ecran împreună cu probabilitatea de clasificare. Detectarea se face la apasarea tastei ‘r’,
numele semnului detectat apare apoi in terminal.

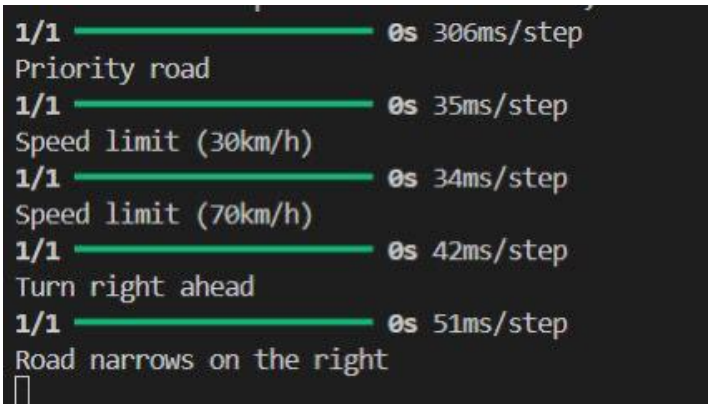
4. Scriptul gui.py

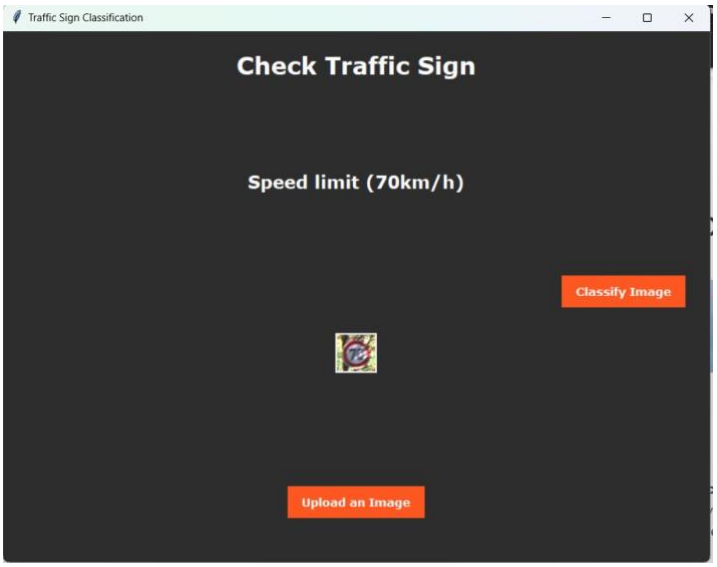
Scriptul gui.py este responsabil pentru interfața grafică a utilizatorului (GUI) a proiectului de recunoaștere a semnelor de circulație. Folosind biblioteca tkinter, acest script creează o fereastră interactivă în care utilizatorii pot încărca imagini și pot vizualiza clasificarea semnelor de circulație.

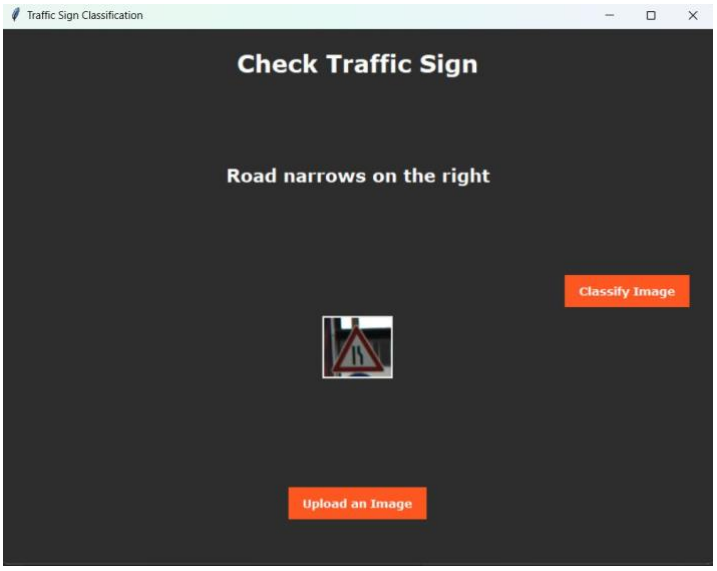
Scriptul începe prin încărcarea modelului de clasificare antrenat traffic_classifier.h5 folosind keras.models.load_model.

Un dicționar numit ‘classes’ este folosit pentru a eticheta fiecare clasă de semn de circulație cu descrierea sa corespunzătoare.
(am folosit un GUI pentru testarea functionalitatii si fara camera).

Exemplu practic din GUI:

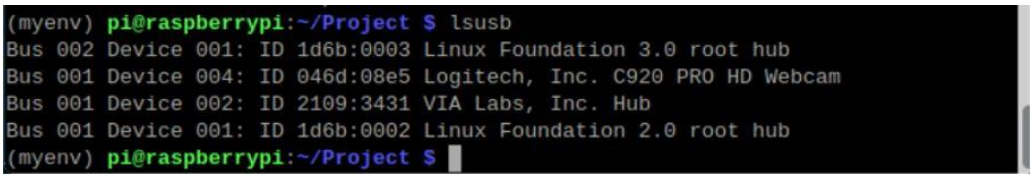






Recunoaștere semne:

Am creat un program în Python care identifică semnele de circulație utilizand un Raspberry Pi conectat la o cameră prin USB. Conectăm Raspberry Pi-ul la laptop folosind un cablu Ethernet și alimentăm placa cu un cablu de alimentare conectat la priză.
Camera conectata poate fi detectata utilizant urmatoarea comanda:



Programul captează imagini în timp real și le analizează pentru a detecta și recunoaște diverse semne de circulație. Aceste semne includ limitări de viteză, indicatoare de prioritate, semne de oprire, și alte semne comune pe drumuri.

Programul nostru folosește tehnici de procesare a imaginii și algoritmi de învățare automată pentru a analiza imaginile capturate de cameră. După ce un semn de circulație este identificat, informația este afișată pe ecran, împreună cu tipul semnului.

Algoritmii de recunoaștere sunt implementați în Python folosind biblioteci precum OpenCV și TensorFlow/Keras. Programul este împărțit în două fișiere principale: detect.py pentru detectarea semnelor și traffic.py pentru antrenarea modelului și generarea diagramelor de performanță.

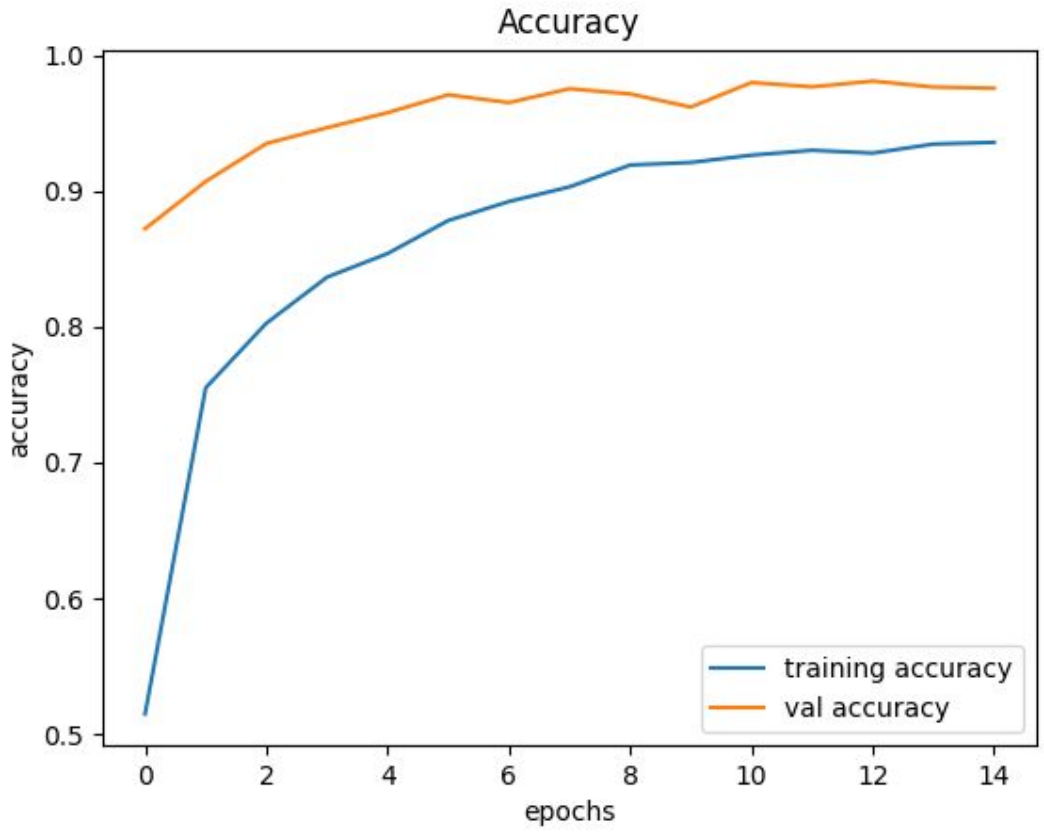
Interpretarea Diagramelor generate in urma rularii traffic.py

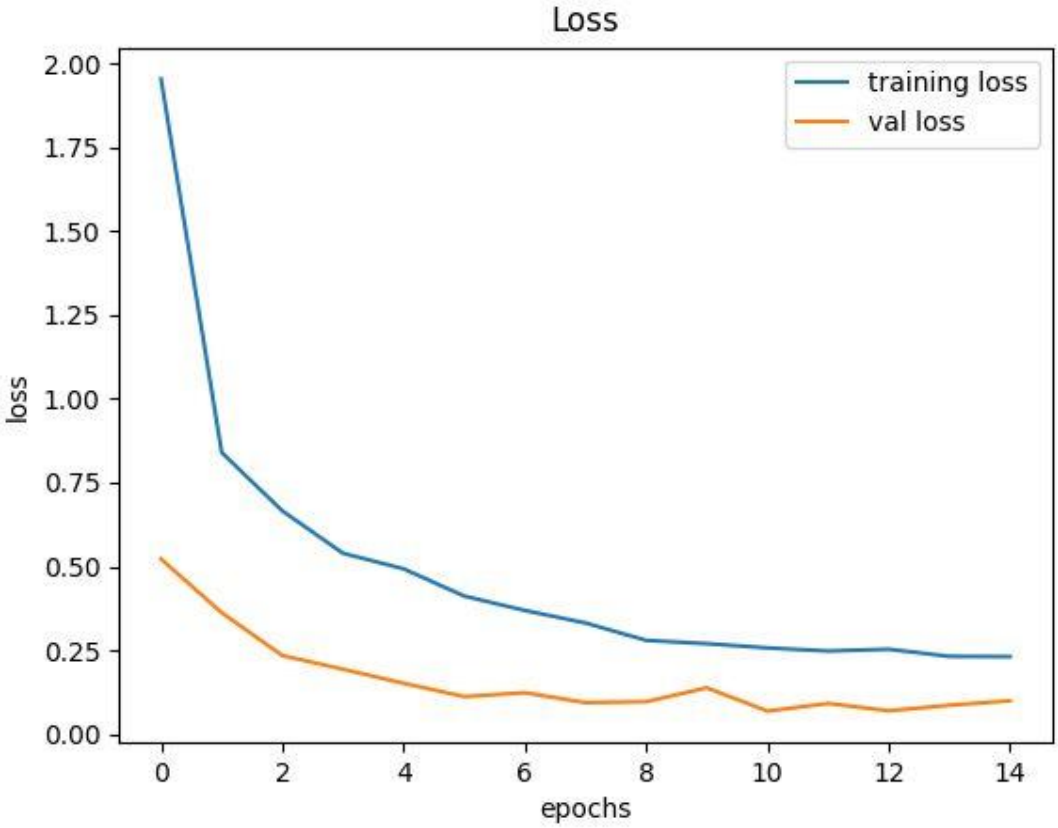
1. Diagrama Acurateței (Accuracy):

• Acuratețea la antrenare (training accuracy): Curba albastră arată cum crește acuratețea modelului pe setul de antrenament pe măsură ce numărul de epoci crește. Putem observa că acuratețea crește rapid la început, apoi continuă să crească mai lent, stabilizându-se aproape de 1.0.
• Acuratețea la validare (validation accuracy): Curba portocalie arată acuratețea modelului pe setul de validare. Acuratețea este mai mare decât cea de antrenament la început, ceea ce poate indica un model bine generalizat care funcționează bine și pe date noi.

2. Diagrama Pierderilor (Loss):

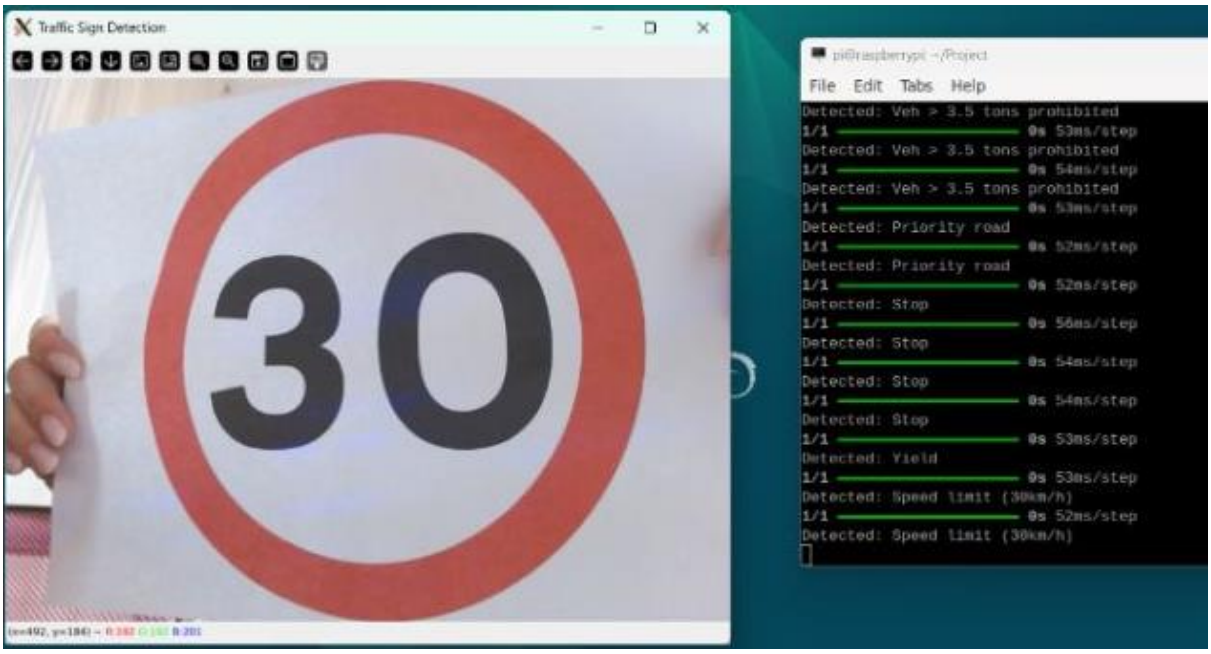
• Pierderile la antrenare (training loss): Curba albastră arată cum scade valoarea pierderilor pe setul de antrenament pe măsură ce modelul se antrenează. Pierderile scad rapid la început și apoi se stabilizează.
• Pierderile la validare (validation loss): Curba portocalie arată valoarea pierderilor pe setul de validare. Similar cu curba de antrenament, pierderile scad și ele rapid și apoi se stabilizează la o valoare mai mică.
Aceste diagrame sunt importante pentru a evalua performanța modelului. O acuratețe înaltă și pierderi scăzute indică faptul că modelul este bine antrenat și poate recunoaște corect semnele de circulație.



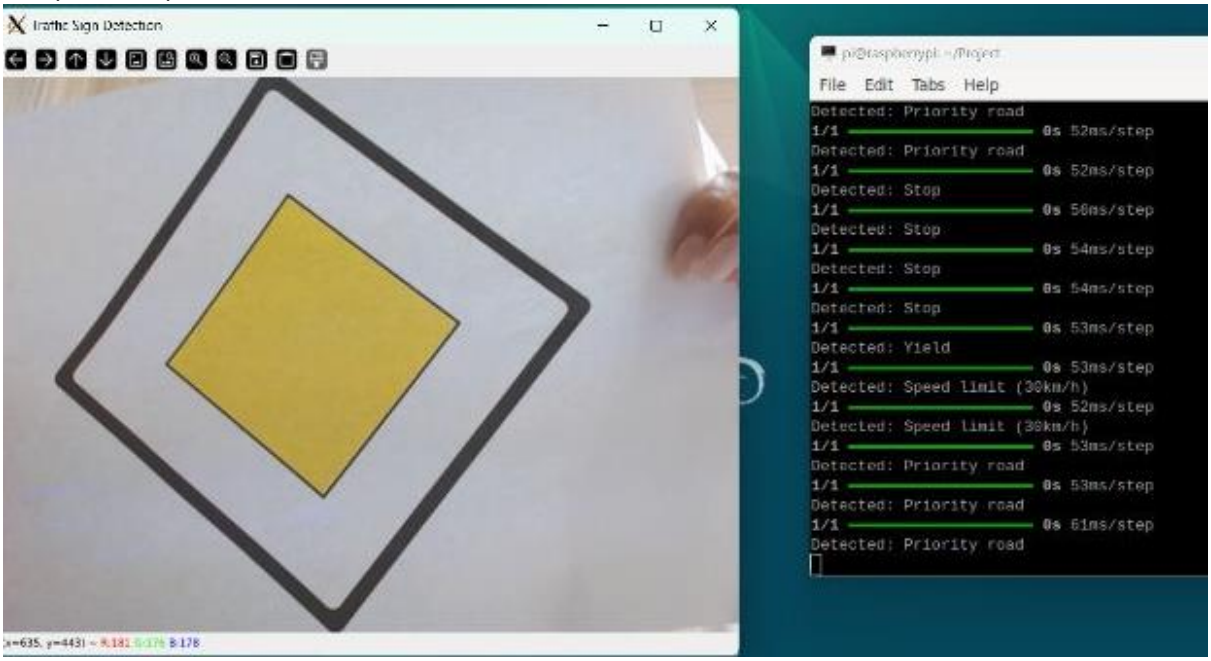

Loss
training loss
val loss
loss
epochs
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
2
4
6
8
10
12
14

Fișierul detect.py

Acest fișier se ocupă de procesarea în timp real a imaginilor capturate de cameră și detectarea semnelor de circulație. Exemplele de ieșiri din imagini arată că programul poate identifica corect semnele și afișa rezultatele în terminal.



1. Prima imagine arată detectarea unui semn de limitare a vitezei (30 km/h). Programul recunoaște corect semnul și afișează "Detected: Speed limit (30km/h)" în terminal.



2. A doua imagine arată detectarea unui semn de drum cu prioritate. Programul recunoaște corect semnul și afișează "Detected: Priority road" în terminal.

Bibliografie:

1. *How to Install Raspberry Pi OS on Raspberry Pi 4*. Link
2. *Setting Up Your Raspberry Pi for Machine Learning*. Link
3. *Raspberry Pi Software*. Link