Prepoznavanje objektov na Jetson Orin Nano

Predlog aplikacije: objektna detekcija ("object detection")
Namesto nečesa preveč kompleksnega, priporočam, da začneš z aplikacijo, ki:

- zajema video (npr. s kamero) ali bere slike,
- na vsaki sliki prepozna predmete (npr. "oseba", "avto", "stol", "knjiga" itd.),
- označi (bounding box) predmete in izpiše njihove kategorije in verjetnosti.

To je zelo univerzalna naloga, ki je osnova za veliko drugih sistemov (slednje, števec, opozorila, avtonomno delovanje …). In ker je široko uporabljana, je mnogo predhodno pripravljenih modelov in primerov, ki jih lahko uporabiš kot izhodišče.

Priporočena knjižnica za začetek: jetson-inference / Hello AI World
Ena od zelo uporabnih iztočnic je projekt **dusty-nv / jetson-inference**, ki vsebuje številne primere (v C++ in Python) za razne naloge: image recognition, object detection, segmentacija, itd. GitHub

Zakaj je dober za začetek:

- že delno urejen projekti, ki jih lahko poženete in razumete, kako delo poteka (vstopna raven).
- prihaja s že predtreniranimi modeli, ki jih lahko preizkusiš takoj. GitHub
- ko razumeš osnovno delovanje, lahko zamenjaš model, narediš svojo zbirko podatkov, treniraš in nadgradiš.

Kako zagnati "detectnet" primer:

1. Namesti jetson-inference (s kloniranjem repozitorija, buildanjem, kar je opisano v repozitoriju).
2. Poženite primer detectnet.py ali detectnet.cpp z vhodno sliko ali video toku — že boš videl zaznane objekte.
3. Nato lahko začneš manipulirati s parametri — npr. glede zanesljivosti (confidence threshold), izbirati želene kategorije ipd.

To ti že daje delujoč sistem, ki ga lahko "nadgradiš".

Nadgradnje in izboljšave po začetku
Ko boš imel osnovno aplikacijo delujočo, lahko postopoma gradiš naprej:

	Kaj nadgraditi	Zakaj / kako
1	Zamenjaj model z boljšim (hitrejšim, natančnejšim)	Lahko preizkusiš modele YOLOv5, YOLOv8 ali novejše, ki so prilagojeni za hitro inferenco na Jetson platformi.
2	Optimacija s TensorRT, kvantizacija (FP16 / INT8)	To zmanjša zakasnitve in poveča FPS — pogosto ključno na robnih napravah.
3	Uporaba DeepStream SDK	DeepStream olajša delo z video cevkami, obdelavo več tokov, integracijo detekcije + sledenja + analitiko.
4	Izdelava lastne zbirke podatkov / učenje modela po meri	Npr. če želiš prepoznati specifične objekte (npr. samo jabolka, samo določene modele avtomobilov), snemi slike, označi (label), treniraj novi model in ga nasadi.
5	Sledenje (tracking) in poglobljene analize	Ko že zaznavaš objekte, lahko dodaš sledenje (da veš, kateri objekt je isti skozi več sličic), analizo poti, števec, opozorila itd.

Aktualni modeli in smernice za Jetson Orin Nano

- Model **YOLO11** je trenutno zelo popularen model za uporabo na Jetson platformi zaradi svoje učinkovitosti in dobre podpore. docs.ultralytics.com
- Ultralytics dokumentacija ima vodnik, kako uvesti YOLO11 z DeepStream + TensorRT na Jetson naprave. docs.ultralytics.com
- Če želiš narediti live video detekcijo s kamero, so primeri, kako zajeti okvirje iz CSI kamere ali USB kamere, in usmeriti modelu detekcije. NVIDIA Developer Forums
- Če želiš praktičen video vodič: "Object Recognition with Jetson Orin Nano using YOLOv8" je en dober primer. YouTube

Korak-po-korak vodič, kako na tvojem **Jetson Orin Nano 8GB** postaviti preprosto aplikacijo za prepoznavanje predmetov v Pythonu.

1. Priprava okolja

- Poskrbi, da imaš zadnjo verzijo **JetPack** nameščeno na Jetson Orin Nano.
  (Običajno pride že z Ubuntu 20.04/22.04 in CUDA/cuDNN podporo).
- Preveri verzijo: `nvcc --version`
- Posodobi sistem: `sudo apt update && sudo apt upgrade -y`
- Namesti osnovna orodja: `sudo apt install python3-pip git cmake -y`

2. Namestitev knjižnice jetson-inference

To je NVIDIA-jev primer projektov za Jetson naprave.

- Kloniraj repozitorij:

```
git clone --recursive https://github.com/dusty-nv/jetson-inference
cd jetson-inference
```

- Buildaj projekt:

```
mkdir build
cd build
cmake ../
make -j$(nproc)
sudo make install
```

- Namesti Python pakete:

```
sudo pip3 install -r ../python/training/detection/ssd/requirements.txt
```

3. Zagon preproste aplikacije za prepoznavanje predmetov

Knjižnica ima že pripravljen primer detectnet.py.

Zaženi na sliki:

```
cd ~/jetson-inference/python/examples
python3 detectnet.py --network=ssd-mobilenet images/car.jpg output.jpg
```

→ Izhod: output.jpg bo imel označene predmete (npr. avto, oseba).

Zaženi na kamero (USB ali CSI):

```
python3 detectnet.py --network=ssd-mobilenet /dev/video0
```

Če imaš CSI kamero (npr. Raspberry Pi Camera Module), uporabi:

```
python3 detectnet.py --network=ssd-mobilenet csi://0
```

4. Razlaga kode (osnovni primer)

Če odpreš detectnet.py, boš našel nekaj takega:

```
import jetson.inference
import jetson.utils

# naloži model
net = jetson.inference.detectNet("ssd-mobilenet", threshold=0.5)

# odpri kamero
camera = jetson.utils.videoSource("csi://0")  # ali /dev/video0
display = jetson.utils.videoOutput("display://0")

while display.IsStreaming():
    img = camera.Capture()
    detections = net.Detect(img)  # prepoznaj predmete
    display.Render(img)
    display.SetStatus("Object Detection | Network {:.0f}
FPS".format(net.GetNetworkFPS()))
```

Kaj nadgraditi

- ✅ Spremeni ssd-mobilenet v močnejši model (npr. ssd-inception-v2, yolov5).
- ✅ Dodaj **shranjevanje rezultatov v datoteko** (npr. JSON log).
- ✅ Dodaj **števec oseb** ali **alarm**, če se pojavi določen objekt.
- ✅ Integriraj s **DeepStream SDK**, če želiš bolj napredne video pipeline.

Minimalna Python aplikacija z YOLOv8 modelom (z uporabo Ultralytics YOLO knjižnice)

Ta aplikacija na **Jetson Orin Nano 8GB** zaznava predmete. To je lažja pot, ker ne rabiš graditi jetson-inference repozitorija – dovolj je, da uporabiš **Ultralytics YOLO** knjižnico, ki ima že pripravljene modele.

1. Namestitev YOLOv8

Najprej namesti **Ultralytics** knjižnico:

```
pip3 install ultralytics opencv-python
```

Preveri, da imaš **Python ≥ 3.8** in da CUDA deluje (JetPack to že vsebuje).

2. Minimalna koda za zaznavo predmetov

Shrani spodnjo kodo kot yolo_detect.py:

```
from ultralytics import YOLO
import cv2

# naloži model (lahki in hiter model - YOLOv8n)
model = YOLO("yolov8n.pt")

# odpri kamero (USB kamera na /dev/video0 ali CSI kamera na 0)
cap = cv2.VideoCapture(0)  # USB kamera
# cap = cv2.VideoCapture("csi://0")  # če uporabljaš CSI kamero

while True:
    ret, frame = cap.read()
    if not ret:
        break

    # zaznaj objekte
    results = model(frame)

    # nariši rezultate na sliko
    annotated_frame = results[0].plot()

    # prikaži
    cv2.imshow("YOLOv8 Detection", annotated_frame)

    # izhod s tipko Q
    if cv2.waitKey(1) & 0xFF == ord("q"):
        break

cap.release()
cv2.destroyAllWindows()
```

3. Zagon

Zaženi aplikacijo:

```
python3 yolo_detect.py
```

Odpre se okno s kamero, kjer bodo zaznani predmeti obkroženi in označeni z imeni + verjetnostjo. (Izhod ustaviš s tipko **Q**.)

4. Nadgradnje

- Če želiš hitrejše delovanje → uporabi yolov8n ali yolov8s.
- Če želiš večjo natančnost → uporabi yolov8m ali yolov8l.
- Če želiš lastno prepoznavanje predmetov → kasneje lahko YOLOv8 natreniraš na svojem datasetu.
- Če želiš višjo hitrost na Jetsonu → izvozi model v **TensorRT**:

```
yolo export model=yolov8n.pt format=engine
```

To je res **minimalen, a delujoč projekt** za začetek. Ko ga boš zagnal, lahko postopoma dodajamo funkcionalnosti (štetje ljudi, logiranje, alarmi, sledenje …).

5. Modifikacije

Recimo, da želimo sliko zajemati iz filma 🎬 in ne kamere. V tem primeru moramo:

V kodi zamenjati vir slike

V tvojem skriptu *yolo_detect.py* zamenjaj to vrstico:

```
cap = cv2.VideoCapture(0)
```

s to vrstico:

```
cap = cv2.VideoCapture("demo1.mp4")
```

To bo kot vir uporabil video datoteko demo1.mp4, ki pa mora biti v isti mapi kot tvoj skript.
Če je drugje, navedi **celotno pot**:

```
cap = cv2.VideoCapture("/home/jetson/Videos/demo1.mp4")
```

Nastavi velikost videa

Če želiš preveriti, kakšna je ločljivost, ali jo omejiti:

```
width  = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
fps    = cap.get(cv2.CAP_PROP_FPS)

print(f"Video: {width}x{height} @ {fps} FPS")
Če želiš zmanjšati obremenitev GPU:
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)
```

(Pomembno) Po koncu videa — ustavi zanko

Ker video ima konec, dodaj to znotraj while zanke:

```
if not ret:
    print("Konec videa.")
    break
```

Končni primer

```
from ultralytics import YOLO
import cv2

model = YOLO("yolov8n.pt")

cap = cv2.VideoCapture("demo1.mp4")  # <-- tvoj video
if not cap.isOpened():
    print("Napaka pri odpiranju videa!")
    exit()

while True:
    ret, frame = cap.read()
    if not ret:
        print("Konec videa.")
        break

    results = model(frame)
    annotated = results[0].plot()

    cv2.imshow("YOLOv8 - Demo Video", annotated)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

Zdaj bomo tvojo aplikacijo razširili tako, da ne bo le prikazovala rezultatov, ampak bo zaznane predmete **shranila v nov video** — npr. output.mp4.

Končna različica yolo_video_save.py

```
from ultralytics import YOLO
import cv2

# Naloži model (majhen, hiter YOLOv8n)
model = YOLO("yolov8n.pt")

# Odpri vhodni video
input_path = "demo1.mp4"
cap = cv2.VideoCapture(input_path)
if not cap.isOpened():
    print("❌ Napaka: ne morem odpreti videa.")
    exit()

# Preberi osnovne informacije o vhodnem videu
fps = cap.get(cv2.CAP_PROP_FPS)
width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
frame_size = (width, height)

print(f"🎞️ Video: {width}x{height} @ {fps:.1f} FPS")

# Nastavi izhodni video (kodek MP4V)
output_path = "output.mp4"
fourcc = cv2.VideoWriter_fourcc(*'mp4v')
out = cv2.VideoWriter(output_path, fourcc, fps, frame_size)

# Obdeluj video po sličicah
while True:
    ret, frame = cap.read()
    if not ret:
        print("✅ Konec videa.")
        break
```

```
    # detekcija predmetov
    results = model(frame)
    annotated = results[0].plot()

    # shrani okvir v izhodni video
    out.write(annotated)

    # (neobvezno: prikaži med obdelavo)
    cv2.imshow("YOLOv8 Processing", annotated)
    if cv2.waitKey(1) & 0xFF == ord("q"):
        print("🔘 Uporabnik prekinil.")
        break

# Zapri vire
cap.release()
out.release()
cv2.destroyAllWindows()

print(f"💾 Rezultat shranjen kot: {output_path}")
```

Uporaba

Shrani kodo kot *yolo_video_save.py*

V isti mapi imej video demo1.mp4.

Zaženi:

```
python3 yolo_video_save.py
```

Ko konča, boš imel datoteko output.mp4, v kateri so **zaznani in označeni predmeti**.

Če se video obdeluje prepočasi, uporabi **manjši model**:

```
model = YOLO("yolov8n.pt")
```

(Namesto npr. yolov8s.pt ali yolov8m.pt.)

Kako YOLOv8 model optimizirati z TensorRT, da bo deloval hitreje na Orin Nano?

Če želiš, da gre hitreje in izkoristi Jetsonov TensorRT:

```
yolo export model=yolov8n.pt format=engine
```

Nato v kodi spremeni:

```
model = YOLO("yolov8n.engine")
```