

Mašinsko učenje na ivici upotrebom *Nvidia Jetson TX2* uređaja

Dušan Bućan

Sadržaj

- * Motivacija i definicija problema
- * Arhitektura sistema
 - * Podsystem za obradu izvornih podataka na ivici
 - * Podsystem kontrolne jedinice
 - * *MLOps* podsystem
- * Korišćeni skupovi podataka
 - * Skup podataka za prepoznavanje pola ljudi
- * Evaluacija rešenja i rezultati
 - * Korišćene metrike
 - * Evaluacija podsystema za obradu izvornih podataka na ivici

Motivacija i definicija problema

- * Mašinsko učenje na ivici predstavlja oblast primene mašinskog učenja na uređajima ograničenih performansi (mobilnim telefonima, embedde system-a...)
- * Prednosti mašinskog učenja na ivici su obrada velike količine podataka u realnom vremenu kao i povećana privatnost izvornih podataka u poređenju sa tradicionalnim sistemima koji koriste modele mašinskog učenja.

Motivacija i definicija problema

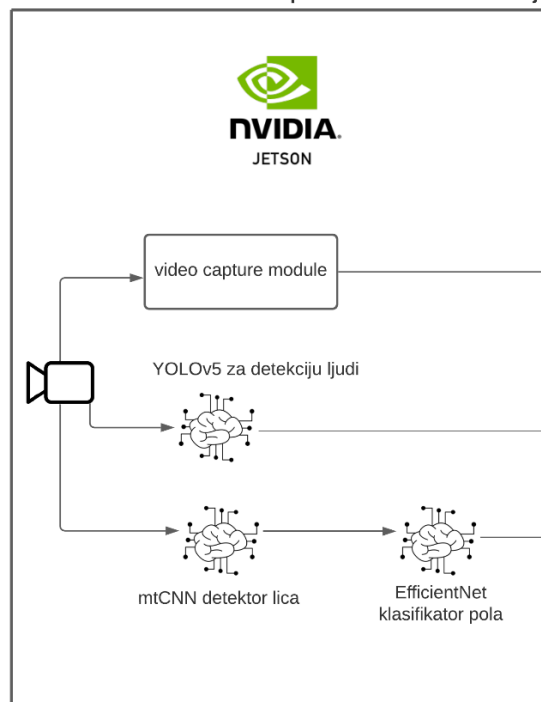
- * Cilj projekta je bio implementirati sistem koji radi demografsku analizu u realnom vremenu upotrebom računarske vizije na *Nvidia Jetson TX2* uređaju.

Arhitektura sistema

- * Tri glavne celine sistema su:
 - Podsystem za obradu izovrnih podataka na ivici
 - implementiran na *Nvidia Jetson TX2* uređaju
upotrebom *Python* programskog jezika
- * Podsystem kontrolne jedinice
 - * Implementiran upotrebom *Flink* i *Superset* alata
- * *MLOps* podsystem
 - * Implementiran upotrebom *MIFlow* platforme

Arhitektura sistema

Podsistem za obradu izvornih podataka na Jetson uređaju



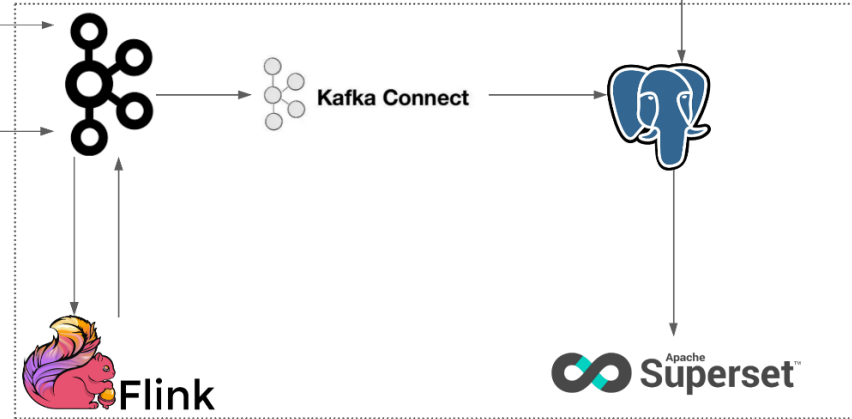
Prikupljanje video materijala

Poruke sa informacijom o detekciji ljudi

Poruke sa informacijom o klasifikaciji pola

MINIO

Infrastruktura na serveru



čuvanje istreniranih modela

Analitika toka podataka

Kafka Connect

Flink

Apache Superset™

Nvidia Jetson TX2 uređaj



Podsistem za obradu izvornih podataka na ivici

- * Osnovne komponente podsistema su:
 - * Komponente za učitavanje videa
 - * Komponente za čuvanje videa
 - * Komponente za slanje poruka
 - * Komponente za detekciju
 - * Komponente za klasifikaciju
 - * *Pipeline* komponente

Komponente za učitavanje videa

- * Cilj komponenti za učitavanje videa je da pruže podršku za učitavanje videa frejm po frejm sa različitih izvora. Osnovna komponenta je predstavljena kao apstraktna klasa *VideoLoader* koju konkretne implementacije nasleđuju.
- * Implementirane su komponente za učitavanje videa sa:
 - * Web kamere – *VideoLoaderWebCamera* klasa
 - * Ugrađene video kamere na Jetson-u – *VideoLoaderJetson* klasa
 - * Fajl sistema – *VideoLoaderFileSystem* klasa

Komponente za čuvanje videa

- * Cilj komponenti za čuvanje videa je da pruže podršku za čuvanje videa na različitim mestima. Razlog čuvanja videa je naknadno treniranje modela. Osnovna komponenta je apstraktna klasa *VideoCapture* koju konkretne implementacije nasleđuju.
- * Implementirane komponente za čuvanje videa su:
 - * Čuvanje videa na lokalnom fajl sistemu
 - * Čuvanje videa na *MinIO* serveru

Komponente za slanje poruka

- * Cilj komponente za slanje poruka je slanje isprocesiranih ulaznih podataka na *Message Broker*-e.
- * Implementirane su komponente za slanje poruka na:
 - * *Kafka*
 - * *RabbitMQ*
- * U trenutnoj verziji sistema se koristi *Kafka* kao *message broker*

Komponente za detekciju

- * Postoje dve grupe komponenti za detekciju:
 - * Komponente za detekciju ljudi – baziraju se na YOLOv5
 - * Komponente za detekciju lica – koriste *mtCNN* detektor
 - * *Joint Face Detection and Alignment using Multi-task Cascade Convolutional Networks*

Komponente za detekciju ljudi

- * Osnovna komponenta je *Detector* koju nasleđuju:
 - * Komponenta za lokalnu upotrebu na računaru – *DetectorYolov5Local*
 - * Komponenta za upotrebu na Jetson-u – *DetectorYolov5Jetson*
- * Obe komponente koriste YoloV5 za detekciju ljudi

YOLOv5

- * Osnovna implementacija YOLOv5 detektora je PyTorch <https://github.com/ultralytics/yolov5>
- * YOLOv5 detektor dolazi u više veličina:
 - * YOLOv5s
 - * YOLOv5m
 - * YOLOv5l
 - * YOLOv5x

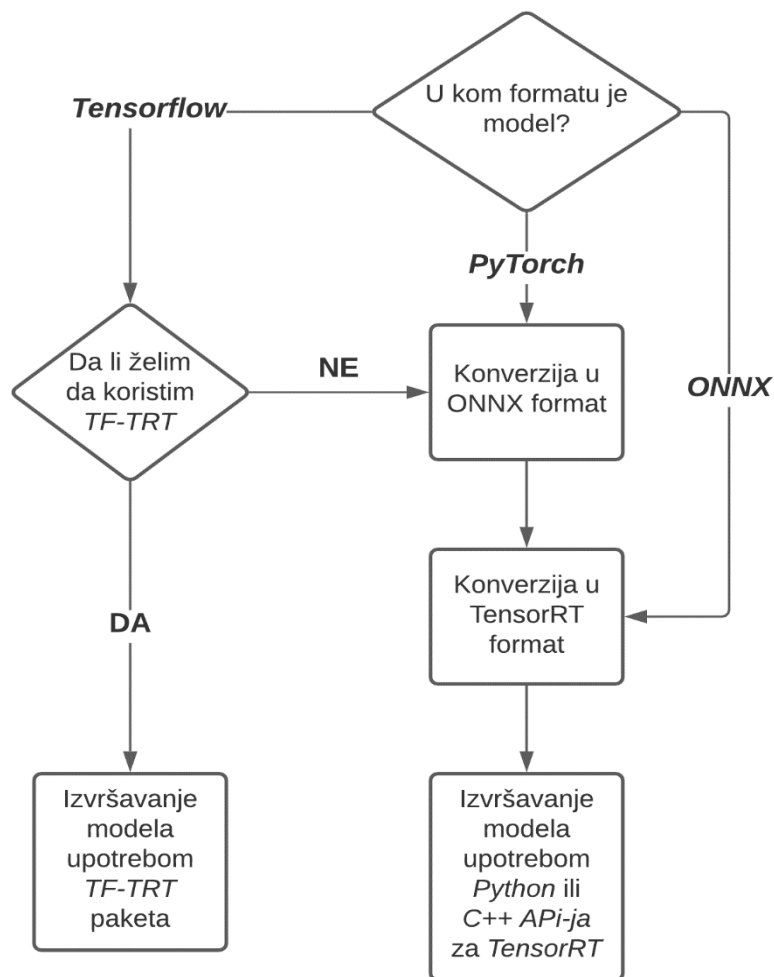
YOLOv5

- * Za implementaciju komponenti za detekciju ljudi je odabran YOLOv5m
- * YOLOv5m je na ručno kreiranom skupu podataka pokazao isti *mAP* kao YOLOv4 detektor uz manji vreme zaključivanja (*inference time*) dok je od YOLOv4Tiny modela postizao veći *mAP*.

Komponenta za lokalnu upotrebu pri detekciji ljudi

- * Pri implementaciji ove komponente je urađena konverzija *PyTorch YOLOv5* u *Tensorflow SavedModel*.

Tok konverzije modela



Komponenta upotrebu na Jetson-u pri detekciji ljudi

- * Pri implementaciji ove komponente je urađena konverzija *PyTorch YOLOv5* u *ONNX* format koji je zatim konvertovan u *TensorRT* format sa *fp16*

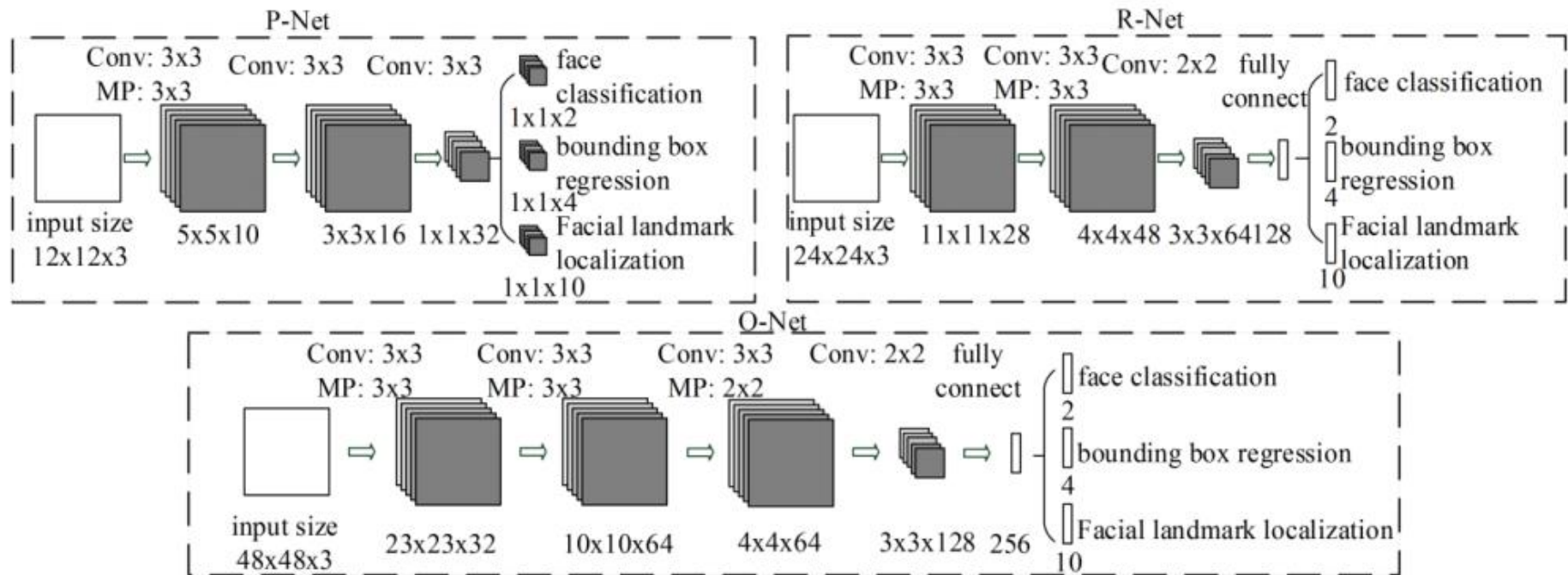
Komponente za detekciju lica

- * Pri odabiru detektora lica isprobani su:
 - * *Face detection using Haar cascade built in OpenCV*
 - * *mtCNN* detektor lica
 - * *DNN module for OpenCV*
- * Za implementaciju komponenti za detekciju lica je odabran *mtCNN* detektor jer postiže bolje rezultate od *Face detection using Haar cascade built in OpenCV* i lakši je za implementaciju na Jetson-u u poređenju sa *DNN module for OpenCV*.
 - * Detaljnije poređenje detektora lica je dato u <https://towardsdatascience.com/face-detection-models-which-to-use-and-why-d263e82c302c>

Komponente za detekciju lica

- * Osnovna komponenta je *FaceDetector* koju nasleđuju:
 - * Komponenta za lokalnu upotrebu na računaru – *FaceDetectorMtcnnLocal*
 - * Implementirana je upotrebom *mtcnn* pip paketa
 - * Komponenta za upotrebu na Jetson-u – *FaceDetectorMtcnnJetson*

Arhitektura konvolucionih mreža koje čine *mtcnn* detektor



Komponenta za upotrebu na Jetson-u pri detekciji lica

- * Komponenta je implementirana kao kombinacija *Cython utility* paketa i *TensorRT* formata *mtcnn* detektora.

Komponente za određivanje pola

- * Komponente za određivanje pola za svaki od regiona u kojima je detektovano lice (dimenzija određuje pol osobe).
- * Model za klasifikaciju pola u osnovi koristi *EfficientNet B7* model kod kojeg je potpuno povezani sloj zamenjen ručno kreiranom arhitekturom.

Komponente za određivanje pola

- * Komponenta za određivanje pola je specificirana kao apstraktna klasa *GenderClassifier* koju nasleđuju:
 - * *GenderClassifierJetson*
 - * *GenderClassifierLocal*
- * Klase naslednice koriste ručno kreirani model opisan ranije

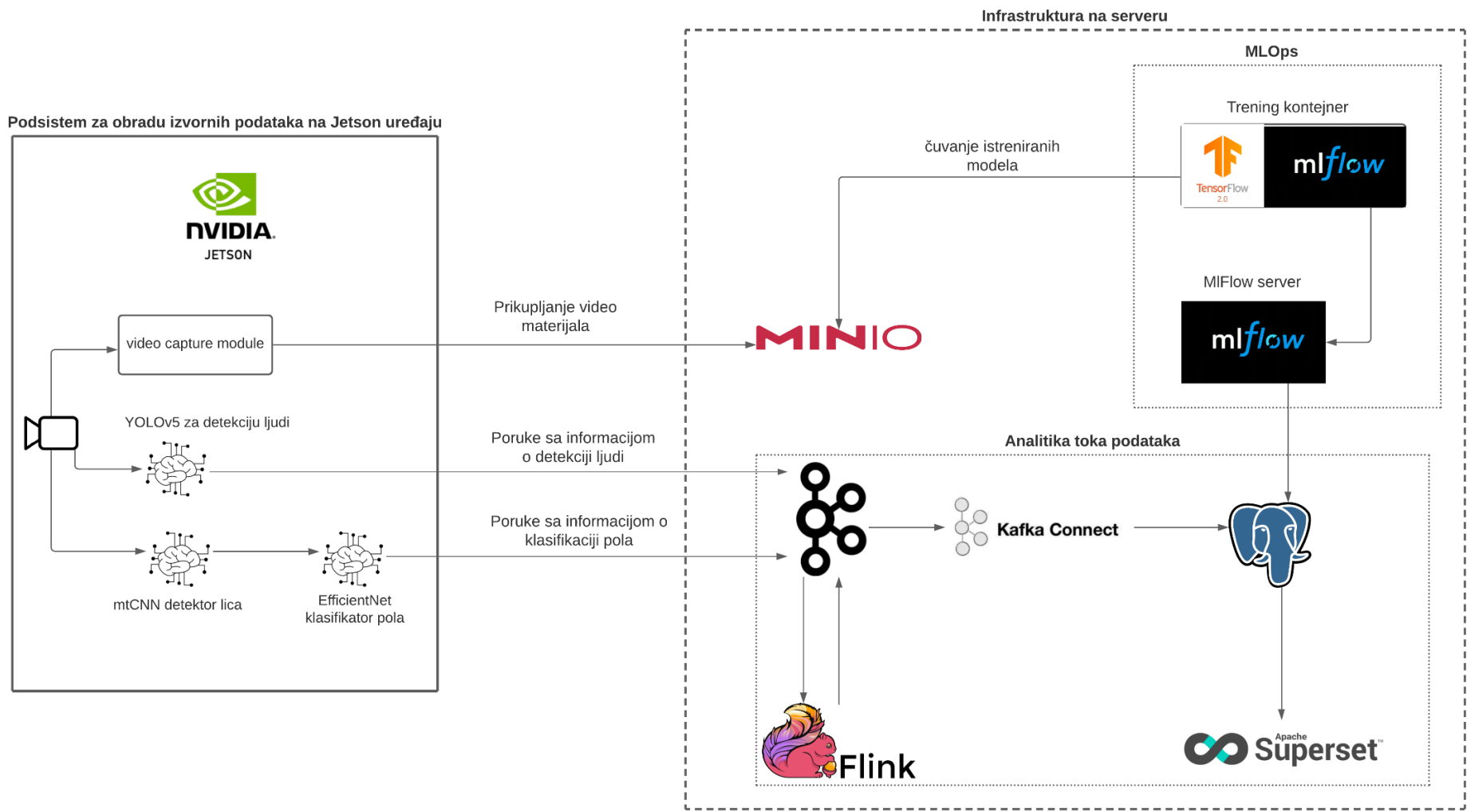
Komponenta za određivanje pola na *Jetson* uređaju

- * Komponenta koristi *TensorRT* format ručno kreiranog klasifikacionog modela koji je ranije opisan.
- * Konverzija u *TensorRT* format se sastoji iz dva koraka:
 - * Konverzija u *ONNX* format
 - * Konverzija se odvija van *Jetson*-a jer je za nju potrebno više resursa nego što *Jetson* poseduje.
 - * Konverzija se obavlja unutar Trening kontejner-a, a rezultat konverzije se postavlja na minIO server odakle se prenosi na *Jetson*
 - * Konverzija iz *ONNX* formata u *TensorRT* format
 - * Izvršava se na *Jetson*-u

Pipeline komponenta

- * Pipeline komponenta na ulazu prima slika a na izlazu šalje skup poruka na *Kafka message broker*.
- * Pipeline komponenta se sastoji od drugi tipova komponenti koji su ranije prikazani i ima za cilj njihovu koordinaciju.
- * U zavisnosti od komponenti koji čine pipeline razlikujemo:
 - * *PeopleCounterPipeline* – na izlazu kreira poruke o detekciji ljudi
 - * *GenderClassificationPipeline* – na izlazu kreira poruke o klasifikaciji pola
 - * *AllPipeline* – na izlazu kreira poruke o detekciji ljudi i klasifikaciji pola

Arhitektura sistema



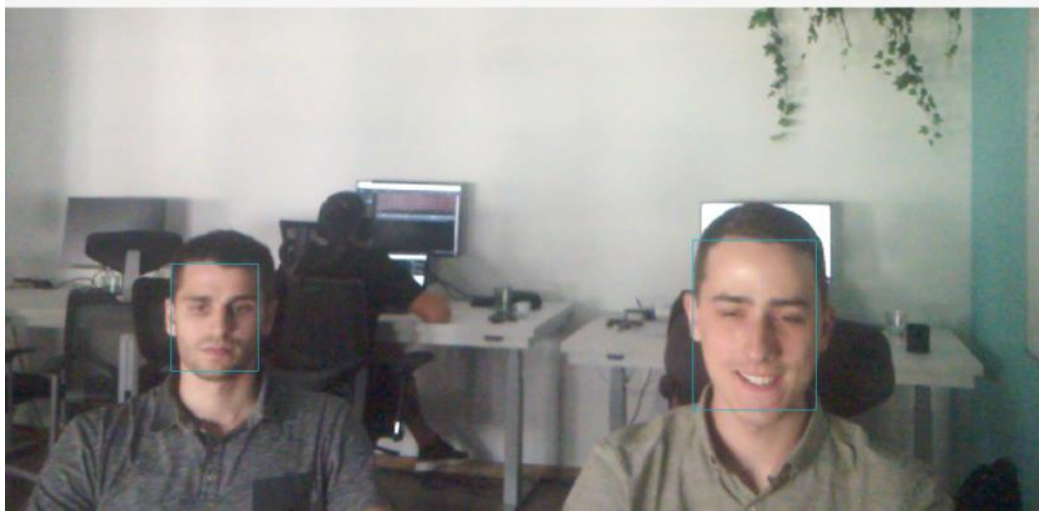
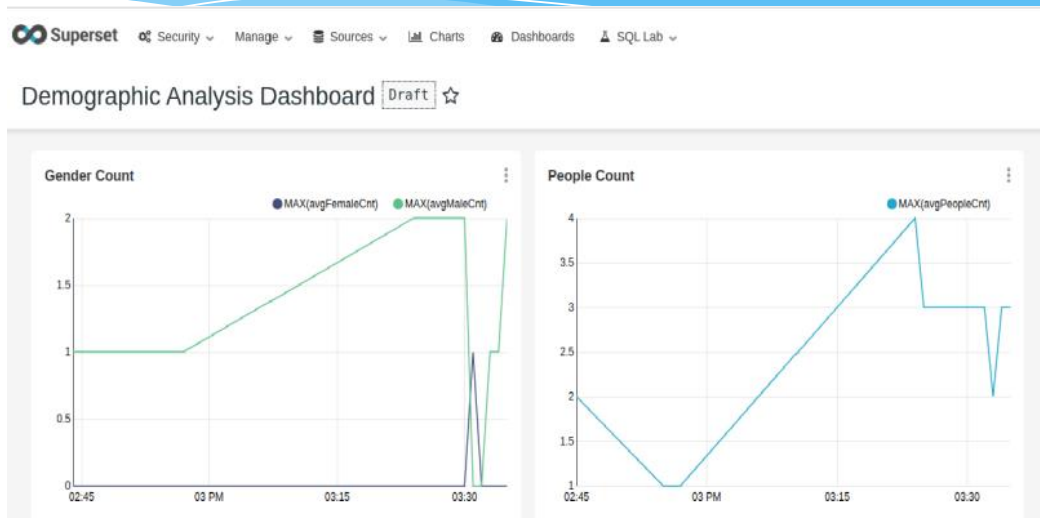
Podsistem kontrolne jedinice

- * Za potrebe obrade toka podataka korišćen je *Flink* alat.
- * Implementirana su dva *Job*-a:
 - * *PeopleCounterScowlJob*
 - * *GenderCounterScowlJob*
- * Izlaz implementiranih *Job*-ova su poruke koje nose informacije o prosečnom broju ljudi/ broju muškaraca i žena u okviru jednog minuta.

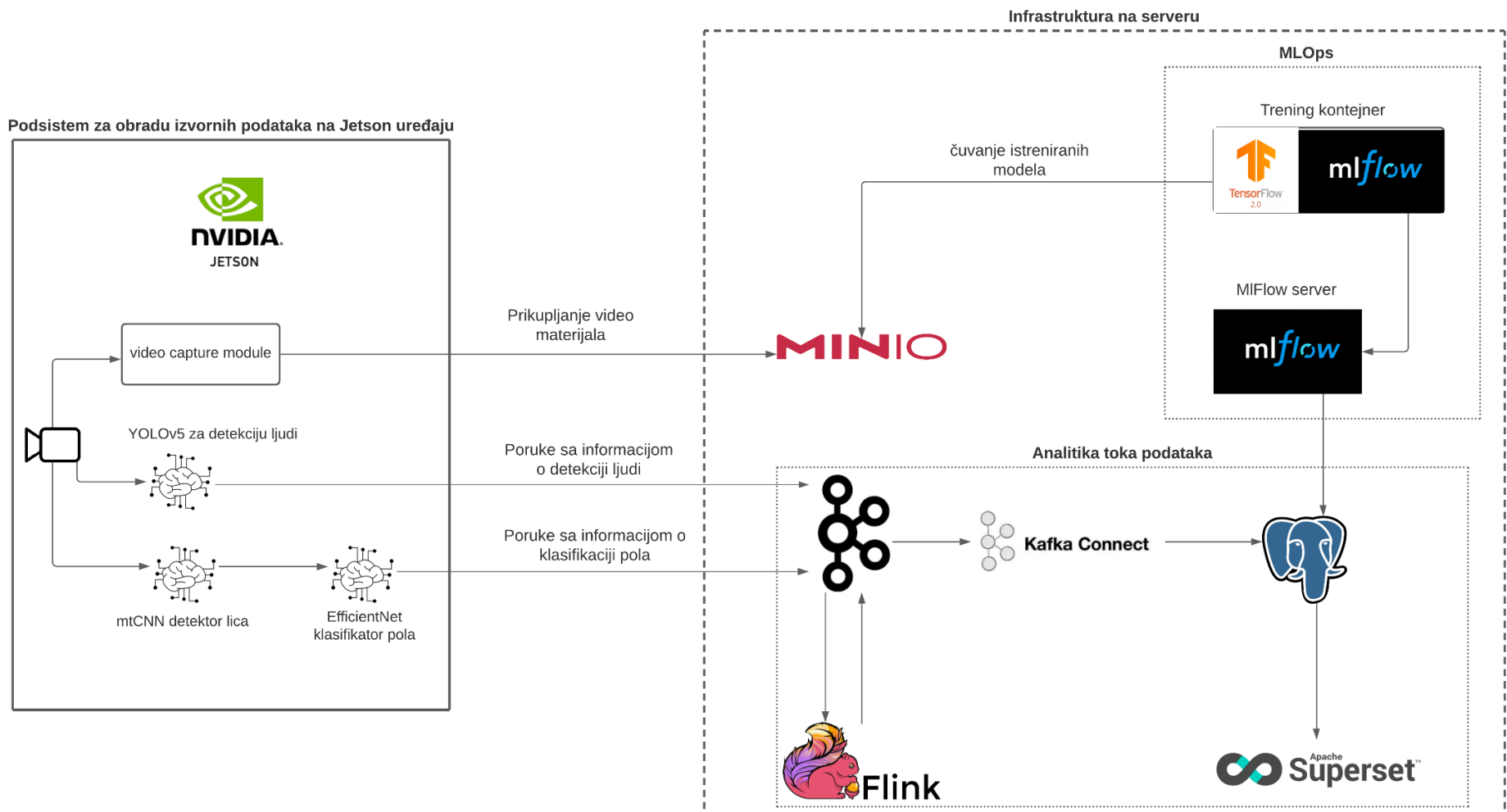
Podsistem kontrolne jedinice

- * Nakon upisa izlaznih poruka *Job*-ova u *Kafka Topic* koristi se *Kafka Connect* za upis tih poruka u *Postgres* bazu podataka.
- * Kontrolna tabla je implementirana upotrebom *Superset* alata koji se oslanja na podatke sačuvane u *PostgreSQL* bazi podataka.

Podsistem kontrolne jedinice



Arhitektura sistema



MLOps

- * Pri implementaciji *MLOps*-a korišćen je *MLFlow* u dva slučaja:
 - * *MLFlow* server - *docker* kontejner
 - * U okviru trening kontejner-a kao *MLFlow* klijent za logovanje metrika na *MLFlow* server
- * *MLFlow* server koristi *PostgreSQL* bazu podataka za čuvanje metrika koje pristižu iz trening kontejnera
- * *minIO* server se koristi čuvanje artefakta (obučenih modela u *.onnx* formatu i *.pb* formatu)

Korišćeni skupovi podataka

- * YOLOv5 detektor je pretreniran na COCO skupu podataka
- * mtCNN je pretreniran na *WIDER FACE* i *CelebA* skupovima podataka
- * Osnova modela za prepoznavanje pola je pretrenirana na *ImageNet* skupu podataka, dok su novi slojevi modela trenirani na *UTKFace* skupu podataka

UTKFace skup podataka

- * *UTKFace* skup podataka sadrži 23567 slika
- * Uzorci unutar skupa su raznovrsni po količini osvetljenja slike, rezolucije slike, starosnim grupama ljudi na slici
- * Za trening je korišćeno 17048 slika, dok validacioni skup čini oko 1894 slika
- * Test skup sadrži preostalih 4735 slika

Evaluacija rešenja i rezultati

Korišćene metrike

- * Metrike korišćene pri evaluaciji podsistema za obradu izvornih podataka na ivici su:
 - * *Mean Average Precision (mAP)*
 - * *F1 score* metrika
- * Korišćene metrike se zasnivaju na metrikama:
 - * Preciznost
 - * Odziv
 - * *Intersection over Union (IoU)*

Korišćene metrike

- * Formula metrike preciznosti:

- * $\frac{TP}{TP+FP}$

- * Preciznost kao metrika odgovara na pitanje koliko model dobro prepoznaje stvarno pozitivne uzorke, odnosno koliko je njegov izlaz relevantan

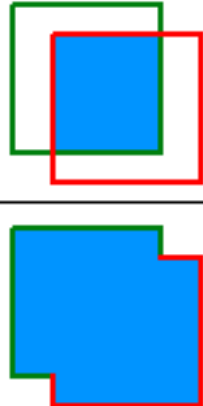
- * Metrika odziva:

- * formula: $\frac{TP}{TP+FN}$

- * Odziv kao metrika odgovara na pitanje koliki je procenat stvarno pozitivnih uzoraka prepoznao model

Korišćene metrike

- * *Intersection over Union (IoU)* metrika:
 - * Uz pomoć *IoU* metrike moguće je izračunati procenat preklapanja dve površine

$$IOU = \frac{\text{area of overlap}}{\text{area of union}} = \frac{\text{area of overlap}}{\text{area of union}}$$


F1 score metrika

- * *F1 score* metrika je jedna od najčešće korišćenih metrika u detekciji objekata
- * *F1 score* metrika se može koristiti na neizbalansiranim skupovima podataka
- * *F1 score* metrika ukazuje na odnos preciznosti i odziva, odnosno kako bi model postigao visoku vrednost *F1 score* metrike potrebno je da ima visoku preciznost i odziv.
- * Formula *F1 score*:
 - *
$$F1\ score = 2 \times \frac{(preciznost \times odziv)}{(preciznost + odziv)}$$

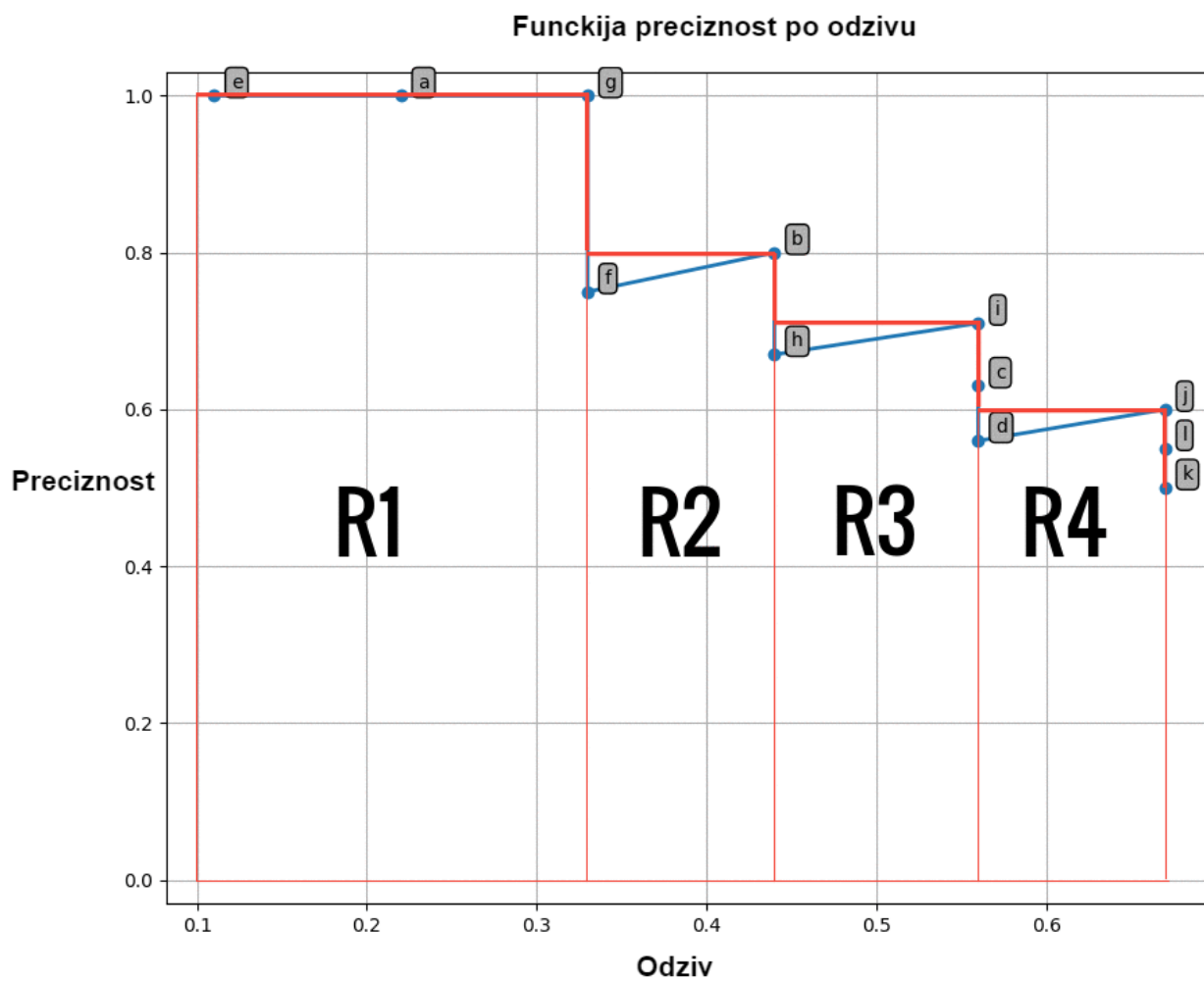
Mean Average Precision (mAP)

- * Zasniva se na metrikama preciznosti, odziva i na *Intersection over Union (IoU)* metrici
 - * Za svaki od predloženih graničnih regiona se računa *IoU* sa odgovarajućim pravim graničnim okvirom
 - * U slučaju da postoji više predloženih regiona samo onaj sa najvećim preklapanjem sa pravim graničnim odgovorom se smatra *TP*, dok se ostali smatraju *FP*.

Mean Average Precision (mAP)

pred. okvir	sigurnost	TP	FP	cumTP	cumFP	sve predikcije	preciznost	odziv
e	99	1	0	1	0	1	1	0.11
a	98	1	0	2	0	2	1	0.22
g	97	1	0	3	0	3	1	0.33
f	96	0	1	3	1	4	0.75	0.33
b	95	1	0	4	1	5	0.8	0.44
h	92	0	1	4	2	6	0.67	0.4
i	89	1	0	5	2	7	0.71	0.56
c	86	0	1	5	3	8	0.63	0.56
d	84	0	1	5	4	9	0.56	0.56
j	73	1	0	6	4	10	0.6	0.67
l	62	0	1	6	5	11	0.55	0.67
k	53	0	1	6	6	12	0.5	0.67

Mean Average Precision (mAP)



$$R1 = 0.33 \times 1$$

$$R2 = (0.44 - 0.33) \times 0.8$$

$$R3 = (0.56 - 0.44) \times 0.71$$

$$R4 = (0.67 - 0.56) \times 0.6$$

$$AP = R1 + R2 + R3 + R4$$

$$mAP = \frac{1}{n} \sum_{i=1}^n AP_i$$

Evaluacija i rezultati YOLOv5m detektora ljudi

- * Pretrenirani YOLOv5m detektor postiže na COCO skupu podataka 63,1% *mAP*
- * Na ručno kreiranim test skupu podataka pretrenirani YOLOv5m postiže 0,845% *mAP*
 - * Zbog visoke vrednosti *mAP* pri detekciju ljudi YOLOv5m nije dodatno treniran nad ručno kreiranim skupom podataka

Evaluacija i rezultati *mtCNN* detektora lica

- * Pretrenirani *mtCNN* detektor lica se pokazao kao jedan od najboljih detektora lica na *CelebA* i *Wider Face* skupovima podataka
- * Zbog činjenice da slike prikupljene sa *Jetson* kamere predstavljaju podskup slika iz *Wider Face* skupa podataka dodatno treniranje *mtCNN* detektora nije bilo potrebno

Evaluacija i rezultati testiranja klasifikatora pola

- * Model kreiran za klasifikaciju pola je evaluiran upotrebom *F1 score* metrike
- * Na izdvojenom test skupu *UTKFace* skupa podataka model za klasifikaciju pola postiže:
 - * 0.855 *F1 score* pri određivanju ženskog pola
 - * 0.856 *F1 score* pri određivanju muškog pola

Sažetak

- * Motivacija i definicija problema
- * Arhitektura sistema
 - * Podsystem za obradu izvornih podataka na ivici
 - * Podsystem kontrolne jedinice
 - * *MLOps* podsystem
- * Korišćeni skupovi podataka
 - * Skup podataka za prepoznavanje pola ljudi
- * Evaluacija rešenja i rezultati
 - * Korišćene metrike
 - * Evaluacija podsystema za obradu izvornih podataka na ivici



Hvala na pažnji!