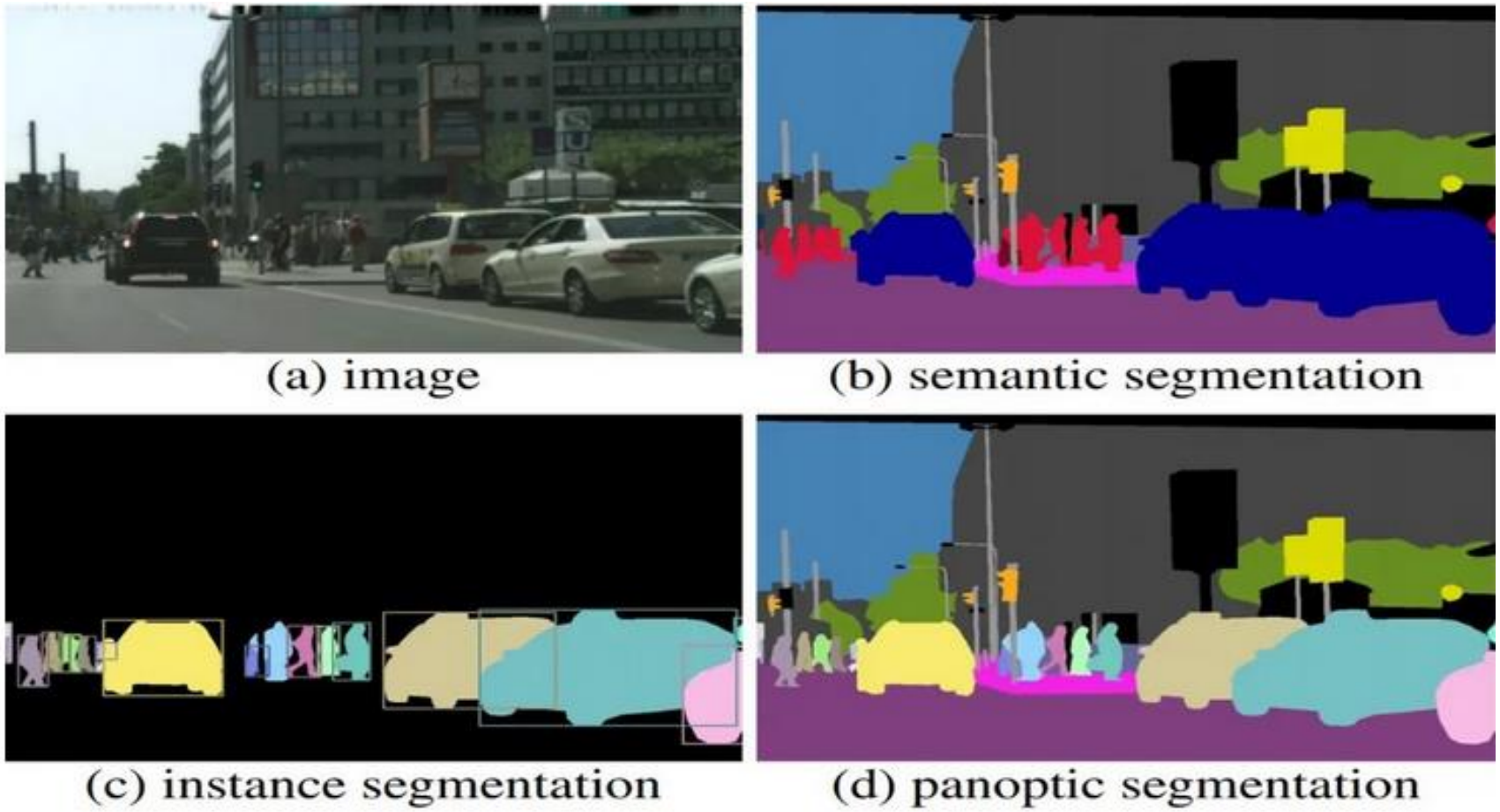
Instance segmentation

Marko Nikitović 123/2020

Kristijan Petronijević 357/2022

Sta je instance semgentation i slicni problemi:



Object Detection

DOG, DOG, CAT

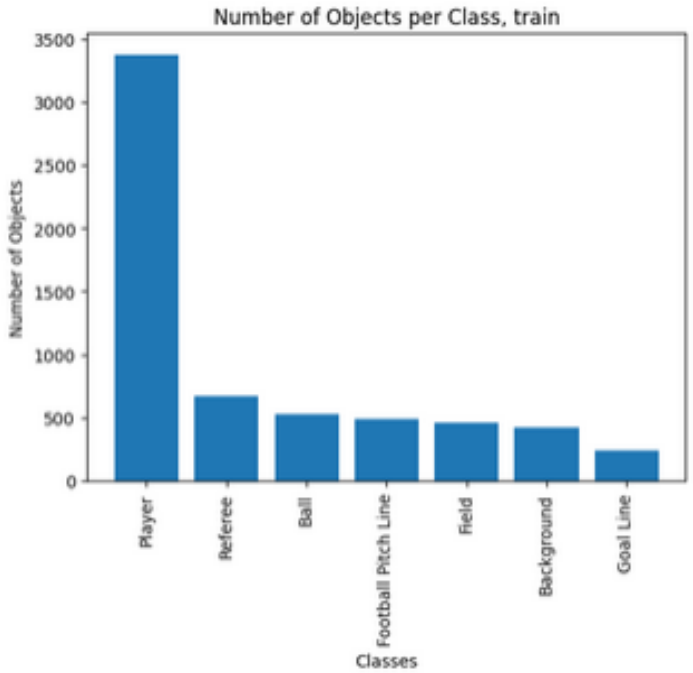
Instance Segmentation

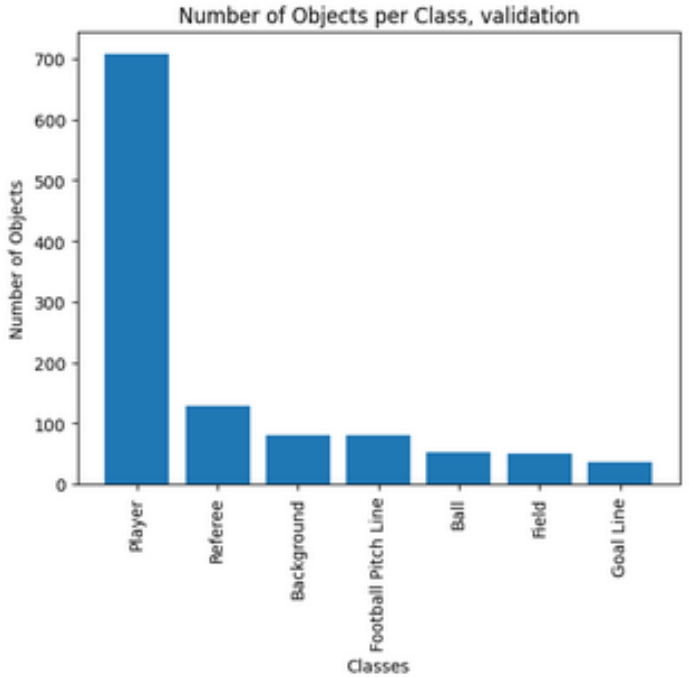
DOG, DOG, CAT

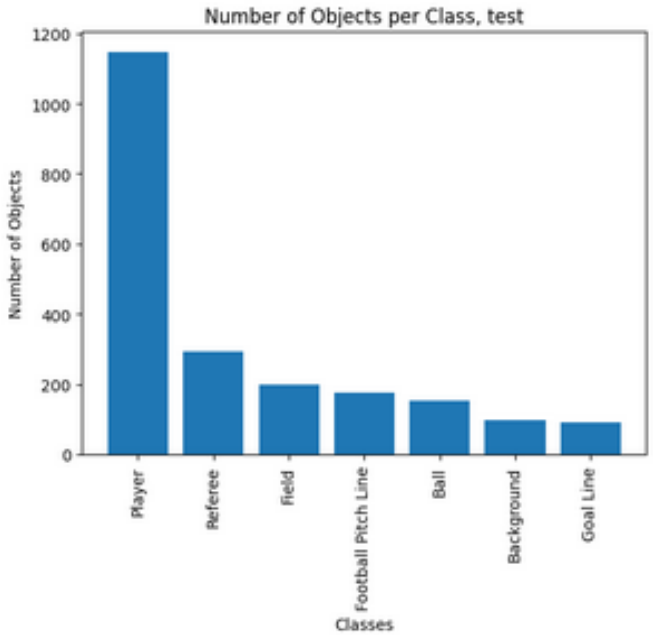
Podaci:

- Tri batch-a od kojih je svaki snimak, velika varijacija izmedju njihovog sadrzaja (ne moze podela da se svakom skupu dodeli po jedan batch)
- Na koji nacin podeliti podatke
- Kako da se izbegne data leakage?
- Delimo po scenama, I scene onda rasporedjujemo u razlicite skupove

Zastupljenost klasa u trening, validacionom I test skupu:





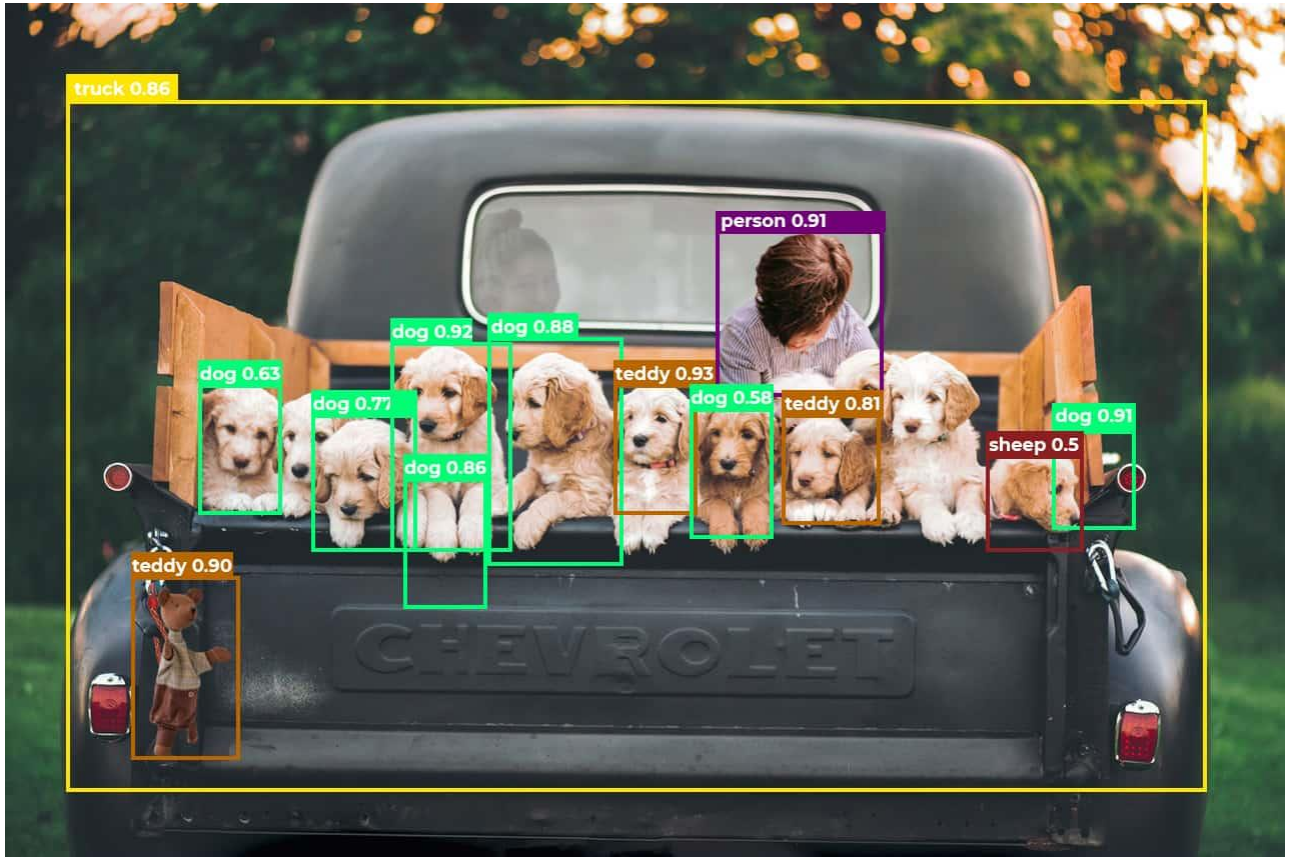


Metrike:

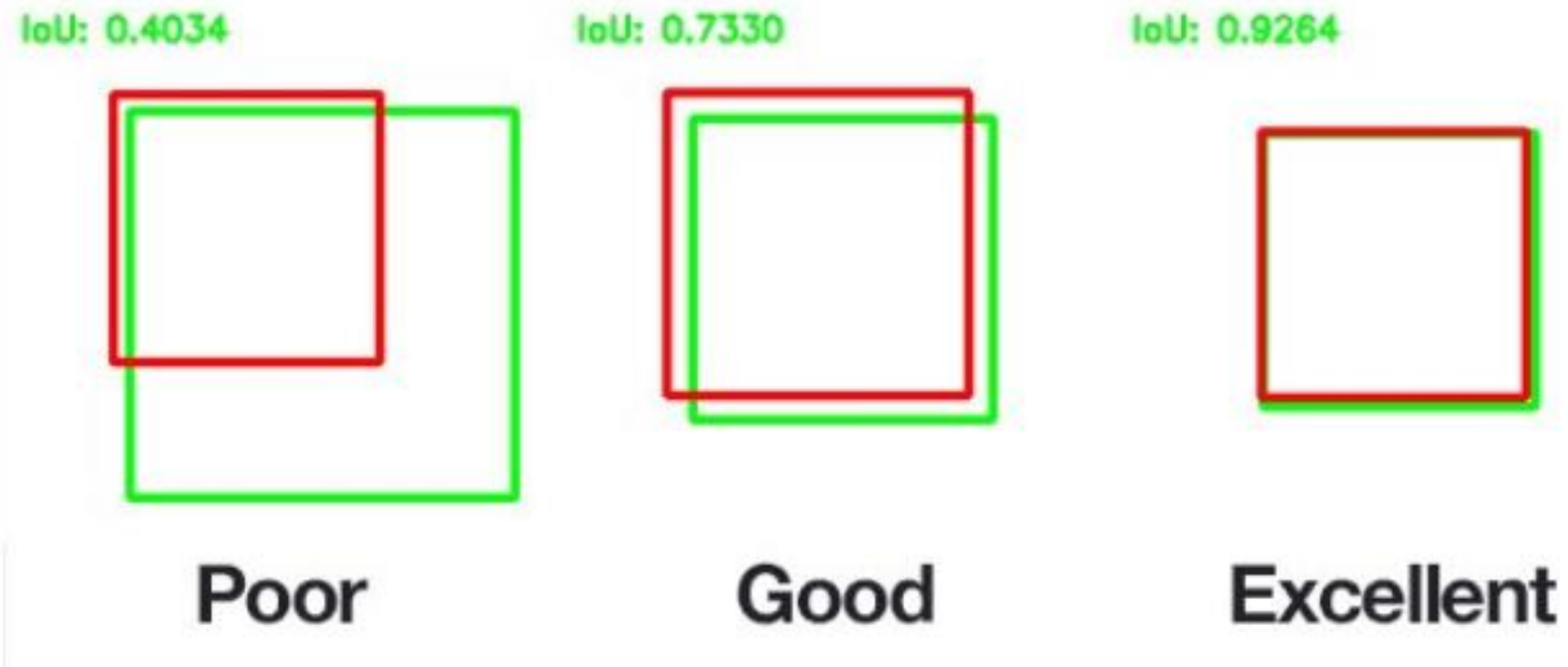
- mAP (Mean average precision)
- IoU (Intersection over union)
- mAR (Mean average recall)

mAP

Ova statistika se racuna kao prosecna vrednost **AP** (eng. Average precision) vrednosti za svaku od klasa, gde se AP izracunava kao povrsina ispod **PR** (eng. Precision - recall) krive.



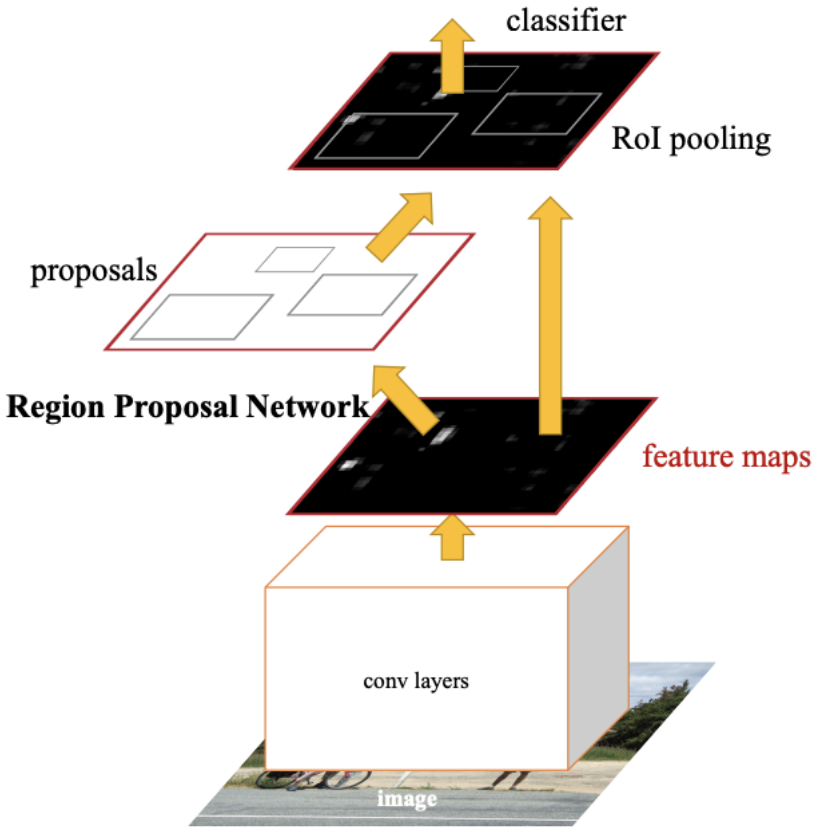
IoU

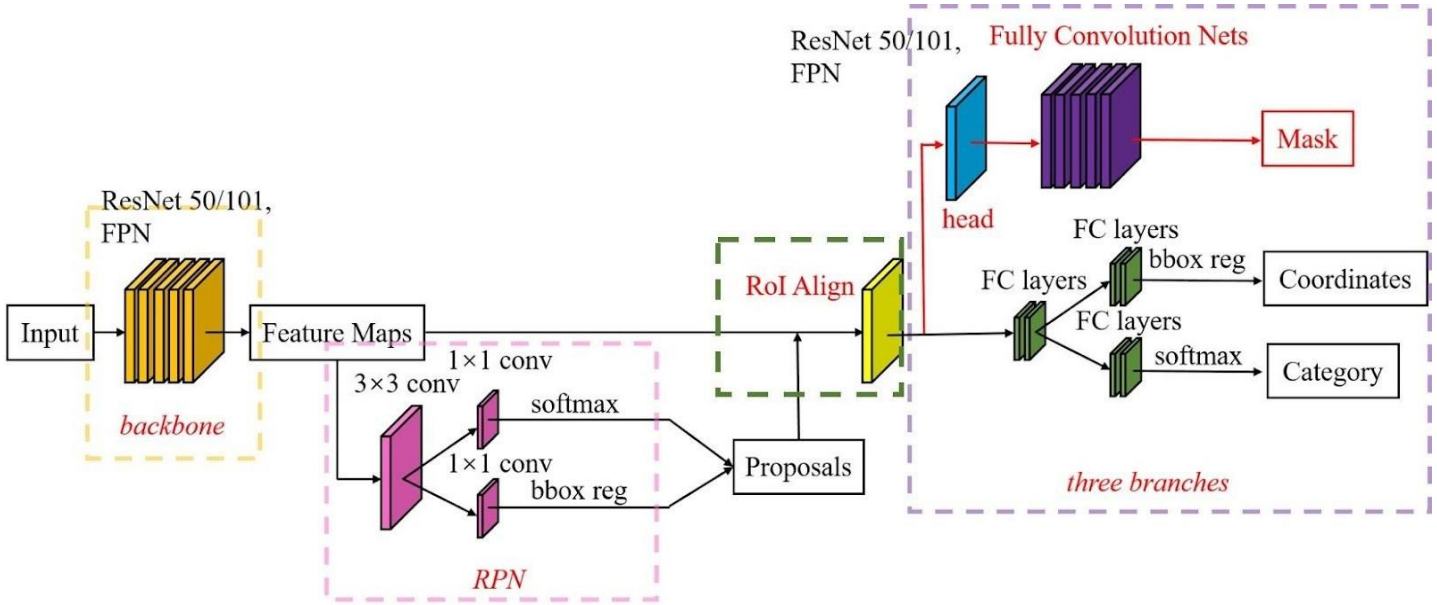


Modeli

- Mask RCNN (Mask Region-based Convolutional Neural Network)
- Cascade Mask RCNN (Detectron library)
- YOLOv8 (nije zavrsen)

Mask R-CNN


classifier
RoI pooling
proposals
Region Proposal Network
feature maps
conv layers
image


ResNet 50/101,
FPN
Fully Convolution Nets
Mask
head
FC layers
bbox reg
Coordinates
FC layers
FC layers
softmax
Category
RoI Align
Input
Feature Maps
backbone
1×1 conv
3×3 conv
softmax
1×1 conv
bbox reg
Proposals
RPN
three branches

Pocetna arhitektura

```python
def get_instance_segmentation_model(num_classes):
    backbone = torchvision.models.resnet50(weights=ResNet50_Weights.IMAGENET1K_V1)
    backbone = torch.nn.Sequential(*(list(backbone.children())[:-2]))
    backbone.out_channels = 2048

    anchor_generator = AnchorGenerator(
        sizes=((32, 64, 128, 256, 512),)
        aspect_ratios=((0.5, 1.0, 2.0),) * 5
    )

    roi_pooler = torchvision.ops.MultiScaleRoIAlign(
        featmap_names=['0'], output_size=7, sampling_ratio=2
    )

    model = MaskRCNN(backbone,
                     num_classes=num_classes,
                     rpn_anchor_generator=anchor_generator,
                     box_roi_pool=roi_pooler)

    return model
```

- **ResNet50**
- Jedan nivo feature mapa
- **RoI pooler**

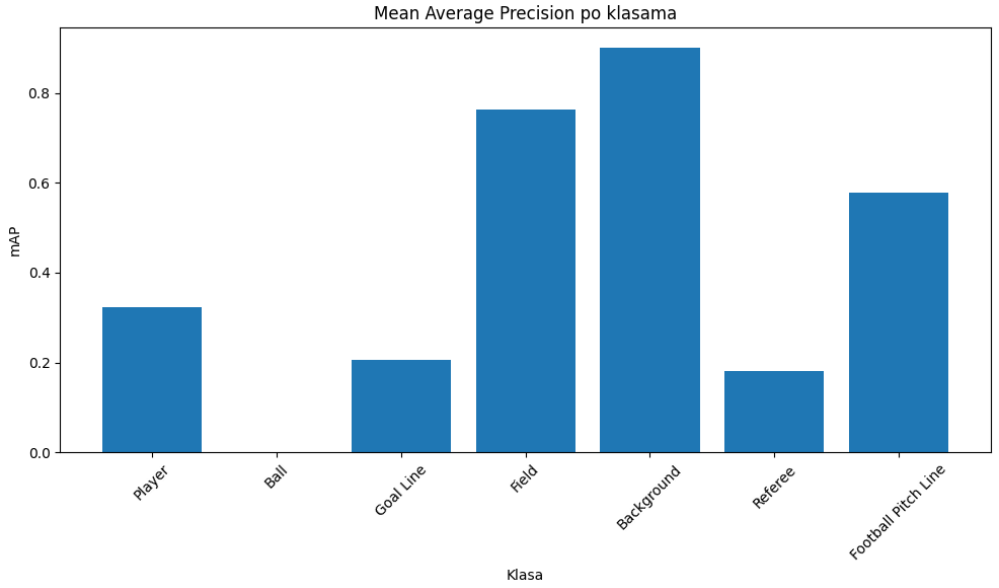
Optimizator

```
optimizer = torch.optim.SGD(params, lr=0.005, momentum=0.9, weight_decay=0.0005)
lr_scheduler = torch.optim.lr_scheduler.StepLR(optimizer, step_size=3, gamma=0.1)
num_epochs = 7
```

- Stohasticki gradijentni spust
- Za update koristimo lr_scheduler na svake tri epohe, za faktor smanjenja brzine ucenja smo uzeli 0.1

Pocetni rezultati

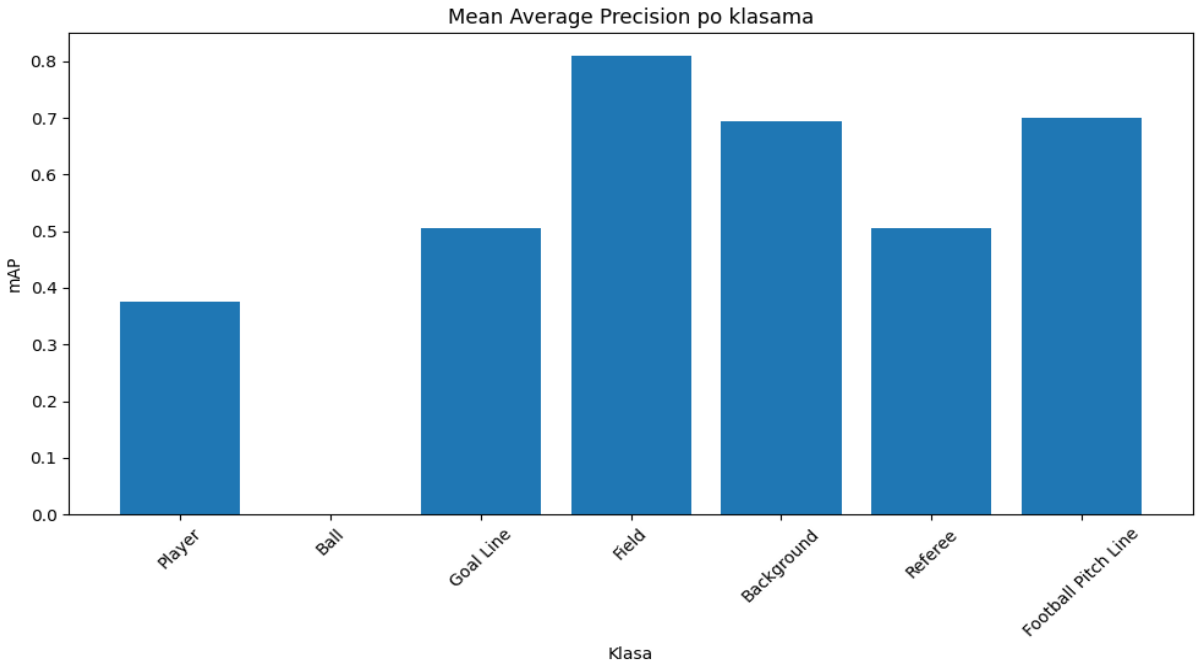
Metrička vrednost	Vrednost
map	tensor(0.4215)
map_50	tensor(0.6788)
map_75	tensor(0.3982)
map_small	tensor(0.)
map_medium	tensor(0.1148)
map_large	tensor(0.5269)
mar_1	tensor(0.3714)
mar_10	tensor(0.4822)
mar_100	tensor(0.4853)
mar_small	tensor(0.)
mar_medium	tensor(0.1584)
mar_large	tensor(0.5966)



- Model ne prepoznaje loptu uopste, kao ni male objekte. Na vecim daje solidne rezultate.

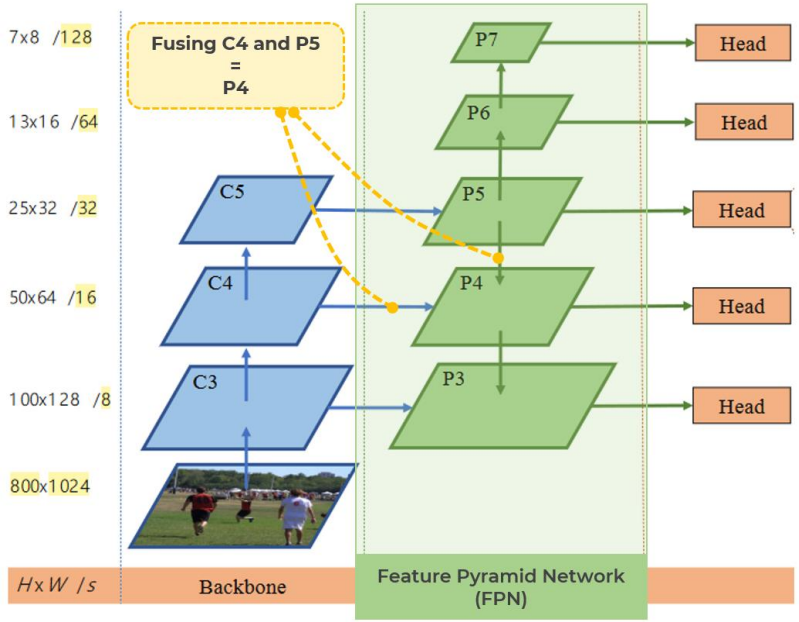
- Lopta na slikama je uglavnom manja od 20px, zato u sizes dodajemo 16.
- Dobijamo nesto bolje rezultate

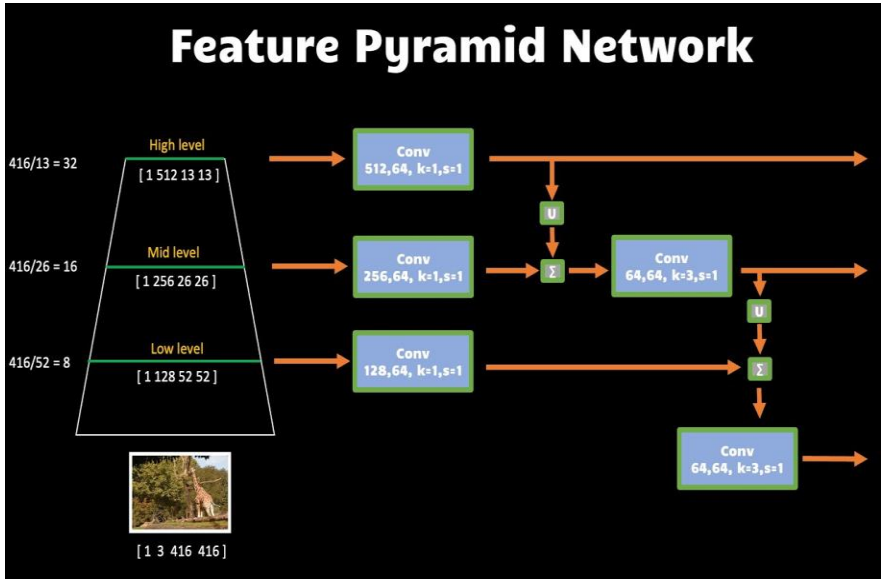
Metrička vrednost	Vrednost
map	tensor(0.5128)
map_50	tensor(0.7426)
map_75	tensor(0.5557)
map_small	tensor(0.)
map_medium	tensor(0.2955)
map_large	tensor(0.6338)
mar_1	tensor(0.4262)
mar_10	tensor(0.5666)
mar_100	tensor(0.5757)
mar_small	tensor(0.)
mar_medium	tensor(0.3293)
mar_large	tensor(0.7024)



Menjanje arhitekture

- Kombinujemo informacije sa razlictih nivoa apstraktnosti
- Sto je nivo apstraktniji koristimo vece velicine kutija

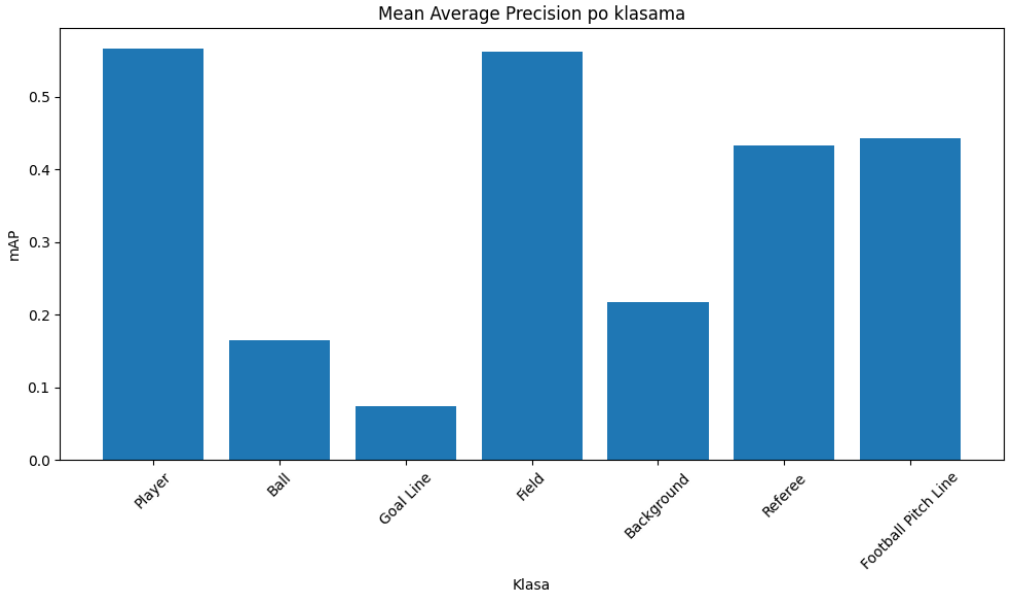


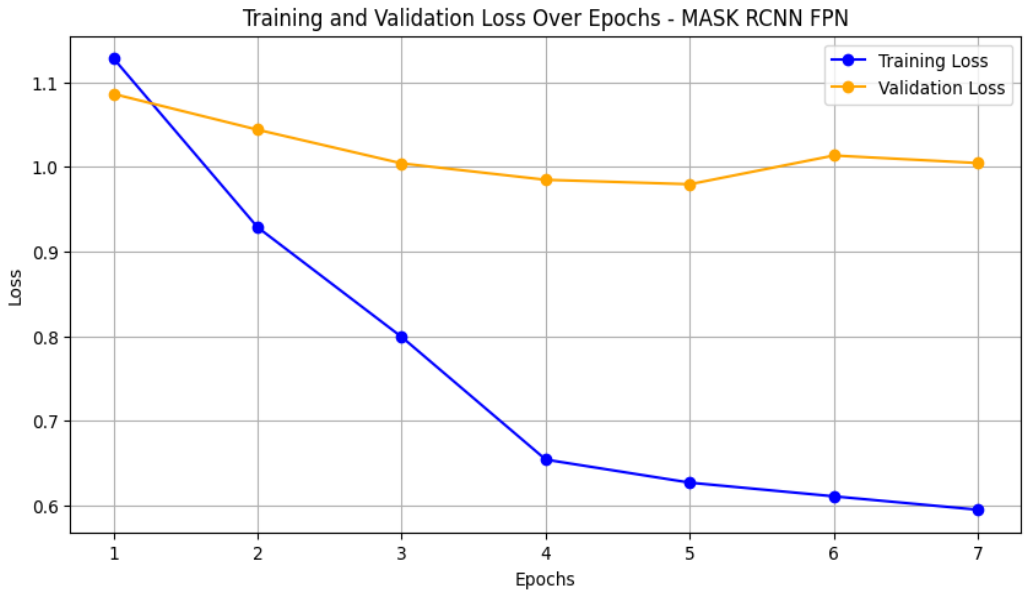


U kodu:

```python
def get_instance_segmentation_model(num_classes):
    backbone = resnet_fpn_backbone('resnet50', weights=ResNet50_Weights.IMAGENET1K_V1)

    anchor_generator = AnchorGenerator(
        sizes=(
            (4, 8, 16),    # P2 - High-resolution feature map for small objects
            (8, 16, 32),   # P4
            (16, 32, 64),  # P5
            (32, 64, 128),  # P6
            (128, 256, 512)
        ),
        aspect_ratios=((0.5, 1.0, 2.0),) * 5
    )

    model = MaskRCNN(
        backbone,
        num_classes=num_classes,
        rpn_anchor_generator=anchor_generator,
        rpn_pre_nms_top_n_train=4000,  # Increased from 2000
        rpn_post_nms_top_n_train=2000, # Increased from 1000
        rpn_pre_nms_top_n_test=2000,   # Increased from 1000
        rpn_post_nms_top_n_test=1000   # Increased from 500
    )

    return model
```

- parametre koji kontrolišu broj predloga regiona (eng. **region proposals**) koji RPN generiše pre i posle **NMS**-a (Non-Maximum Suppression) tokom treniranja i testiranja smo duplo povecali

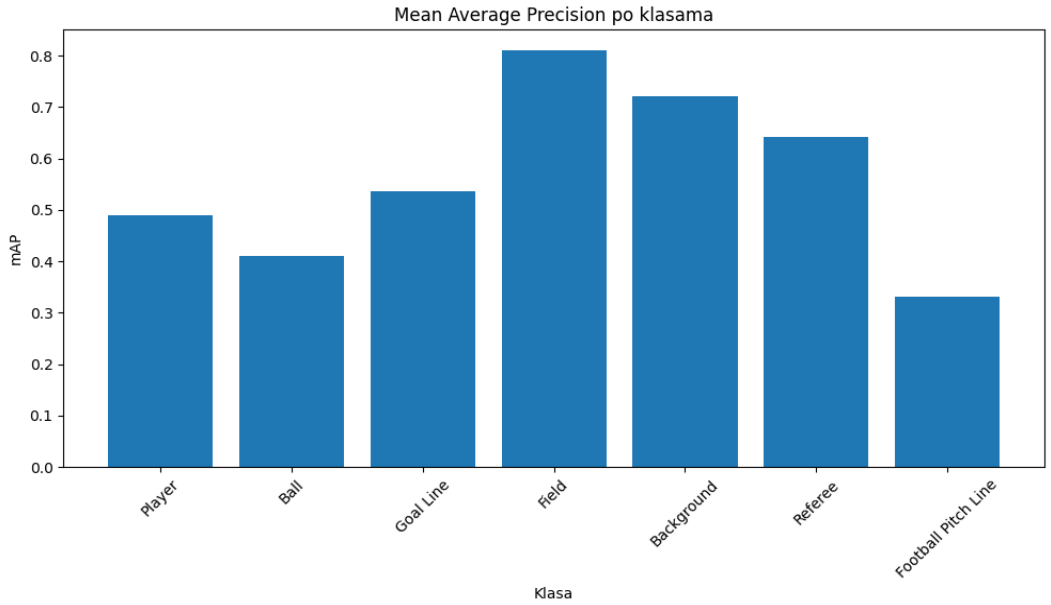




- Model je poceo da prepoznaje loptu
- Poveca se kvalitet modela za igrace
- smanjio kvalitet predikcije za gol liniju i pozadinu koji zauzimaju veći deo slike.
- Deluje kao da model preuranjeno konvergira

- Broj epoha povecavamo na 10
- Step_size stavljamo na 6

```
Ukupne metrike:
Metrička vrednost      Vrednost
                map tensor(0.5631)
             map_50 tensor(0.8625)
             map_75 tensor(0.5900)
          map_small tensor(0.2070)
         map_medium tensor(0.3982)
          map_large tensor(0.6000)
              mar_1 tensor(0.4848)
             mar_10 tensor(0.6331)
            mar_100 tensor(0.6451)
          mar_small tensor(0.2240)
         mar_medium tensor(0.4430)
          mar_large tensor(0.6887)
```

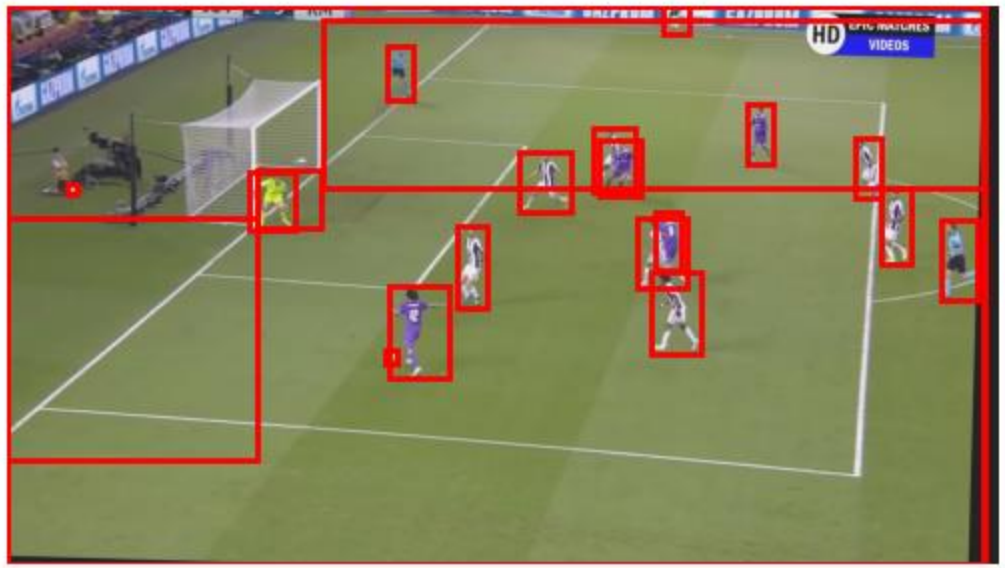


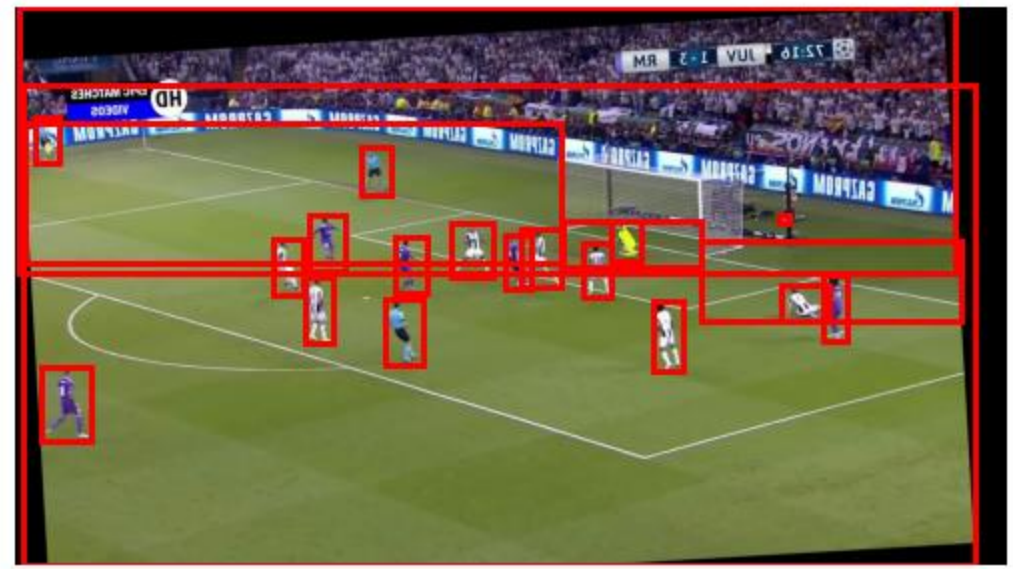
Proširivanje skupa podataka

- Trenutna velicina trening skupa: 567
- Uvecavamo taj broj duplo
- Secenje, horizontalno okretanje, promena osvetljenja,...

```
geom_transforms =  [A.ShiftScaleRotate(shift_limit=0.0625, scale_limit=0.1, rotate_limit=5,
                        interpolation=cv2.INTER_LINEAR, border_mode=cv2.BORDER_CONSTANT, value=0, mask_value=0, p=0.5),

                    A.HorizontalFlip(p=0.5),
                    A.RandomSizedCrop((800, 1070), height, width, w2h_ratio=1920/1080,
                                      interpolation=cv2.INTER_CUBIC, always_apply=False, p=0.5),
                    A.RandomBrightnessContrast(brightness_limit=0.2, contrast_limit=0.2, p=0.5)
                    ]
```

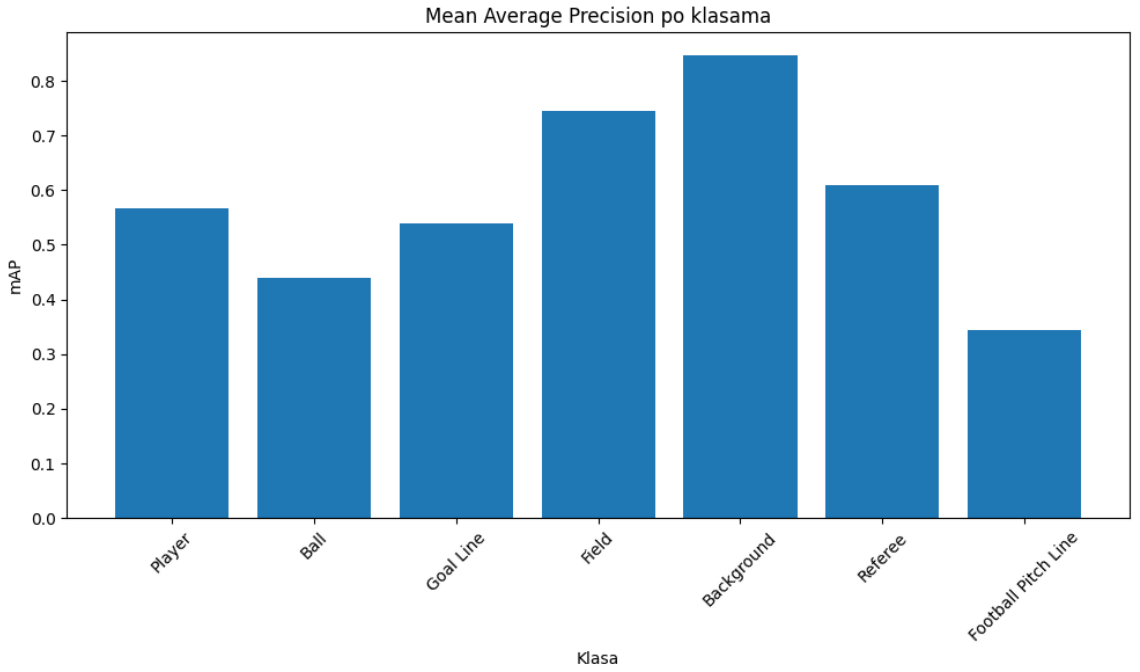
- Primeri primene tranformacija:

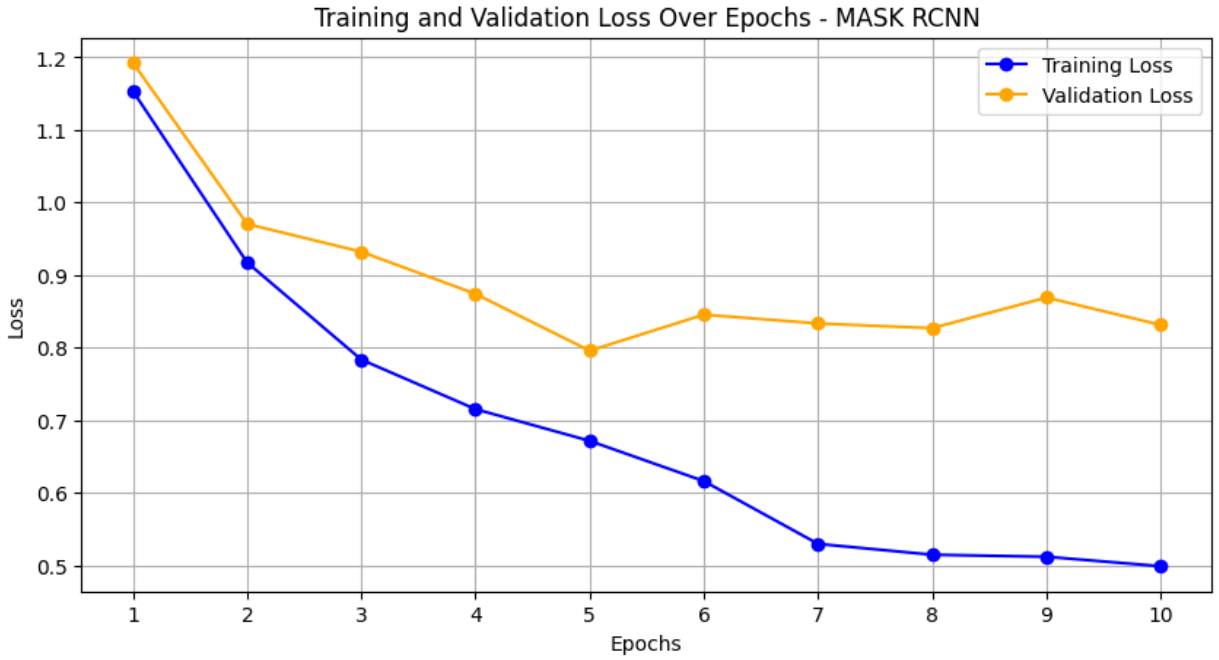




- Model se poboljsao

```
Ukupne metrike:
Metrička vrednost      Vrednost
                map tensor(0.5843)
             map_50 tensor(0.8870)
             map_75 tensor(0.5841)
          map_small tensor(0.2204)
         map_medium tensor(0.4023)
          map_large tensor(0.6295)
              mar_1 tensor(0.4625)
             mar_10 tensor(0.6376)
            mar_100 tensor(0.6541)
          mar_small tensor(0.2510)
         mar_medium tensor(0.4678)
          mar_large tensor(0.6917)
```



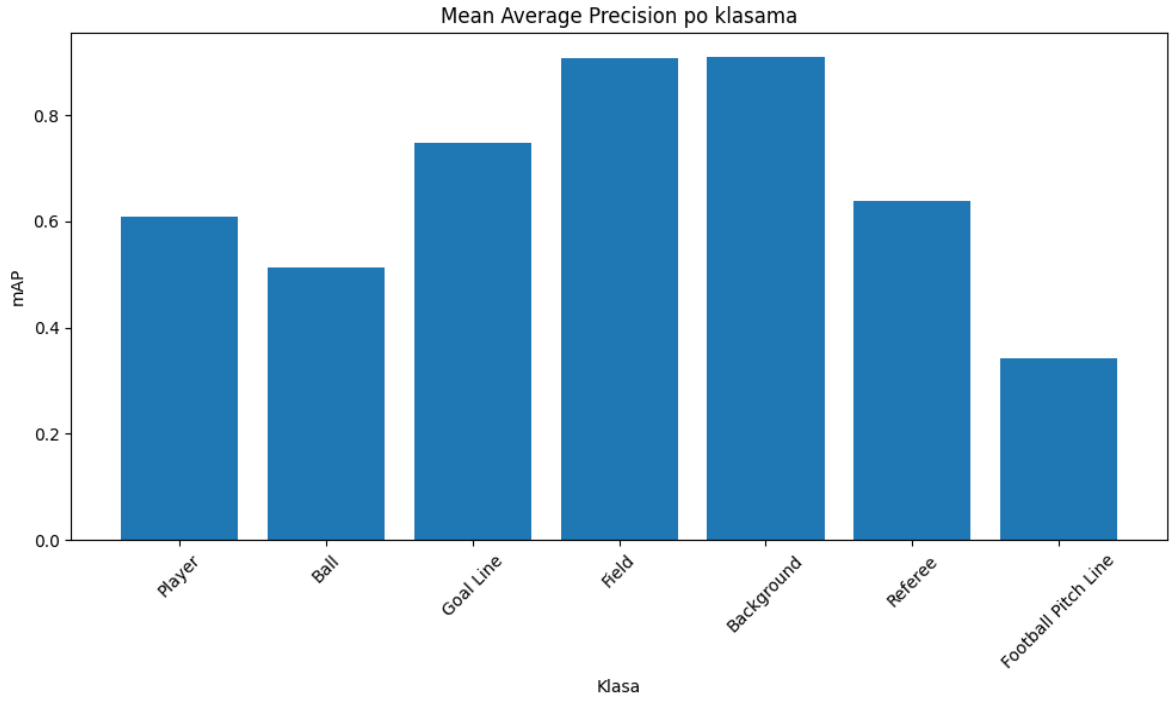


Epoha	1	2	3	4	5	6	7	8	9	10
mAP	0.2287	0.3537	0.5099	0.5160	0.5312	0.5707	0.5710	0.5832	0.5840	0.5843

- Deluje da moze bolje I da model prerano konvergira
- Prelazimo na **Adam**-a (lr=0.0001, weight_decay=1e-5

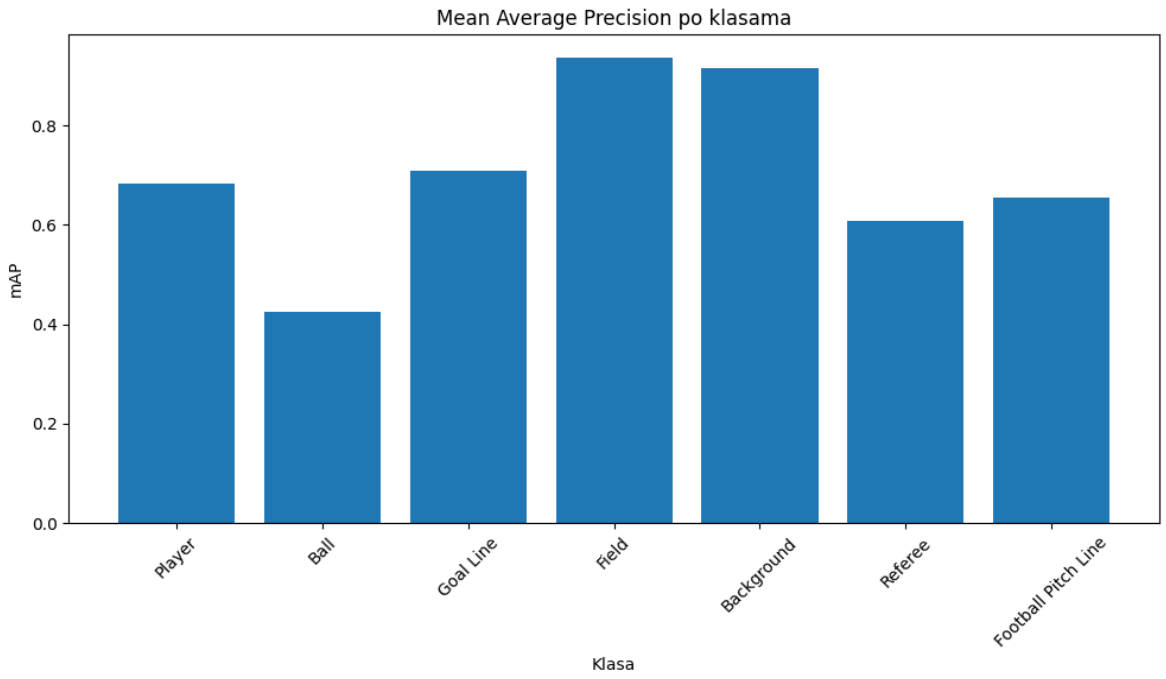
- Drasticne promene kvaliteta

Metrička vrednost	Vrednost
map	tensor(0.6669)
map_50	tensor(0.9028)
map_75	tensor(0.7390)
map_small	tensor(0.2595)
map_medium	tensor(0.4288)
map_large	tensor(0.7137)
mar_1	tensor(0.5398)
mar_10	tensor(0.7091)
mar_100	tensor(0.7278)
mar_small	tensor(0.2819)
mar_medium	tensor(0.4872)
mar_large	tensor(0.7648)



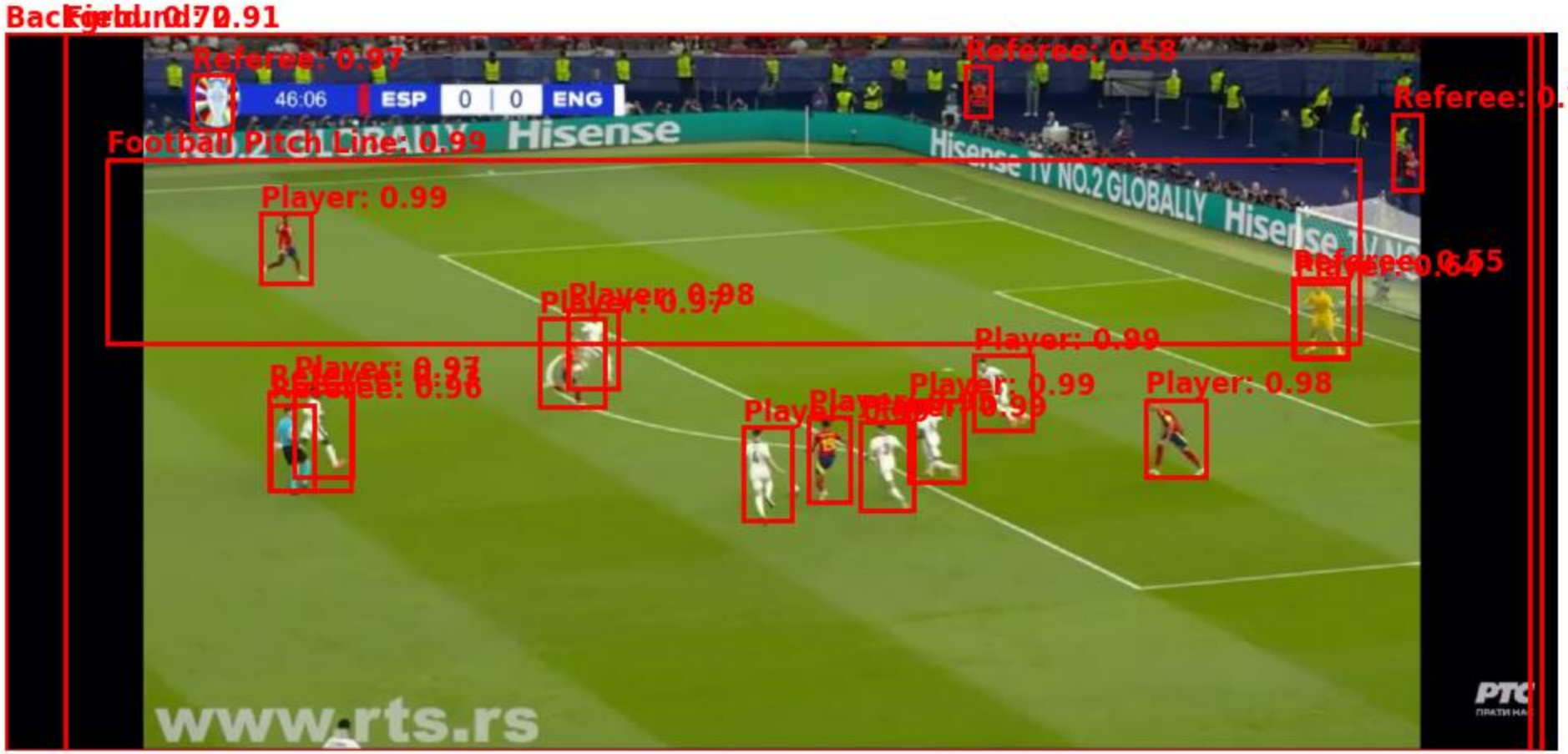
Testiranje modela

Metrička vrednost	Vrednost
map	tensor(0.7041)
map_50	tensor(0.9371)
map_75	tensor(0.8293)
map_small	tensor(0.1416)
map_medium	tensor(0.4662)
map_large	tensor(0.7834)
mar_1	tensor(0.5303)
mar_10	tensor(0.7589)
mar_100	tensor(0.7640)
mar_small	tensor(0.1732)
mar_medium	tensor(0.5073)
mar_large	tensor(0.8244)



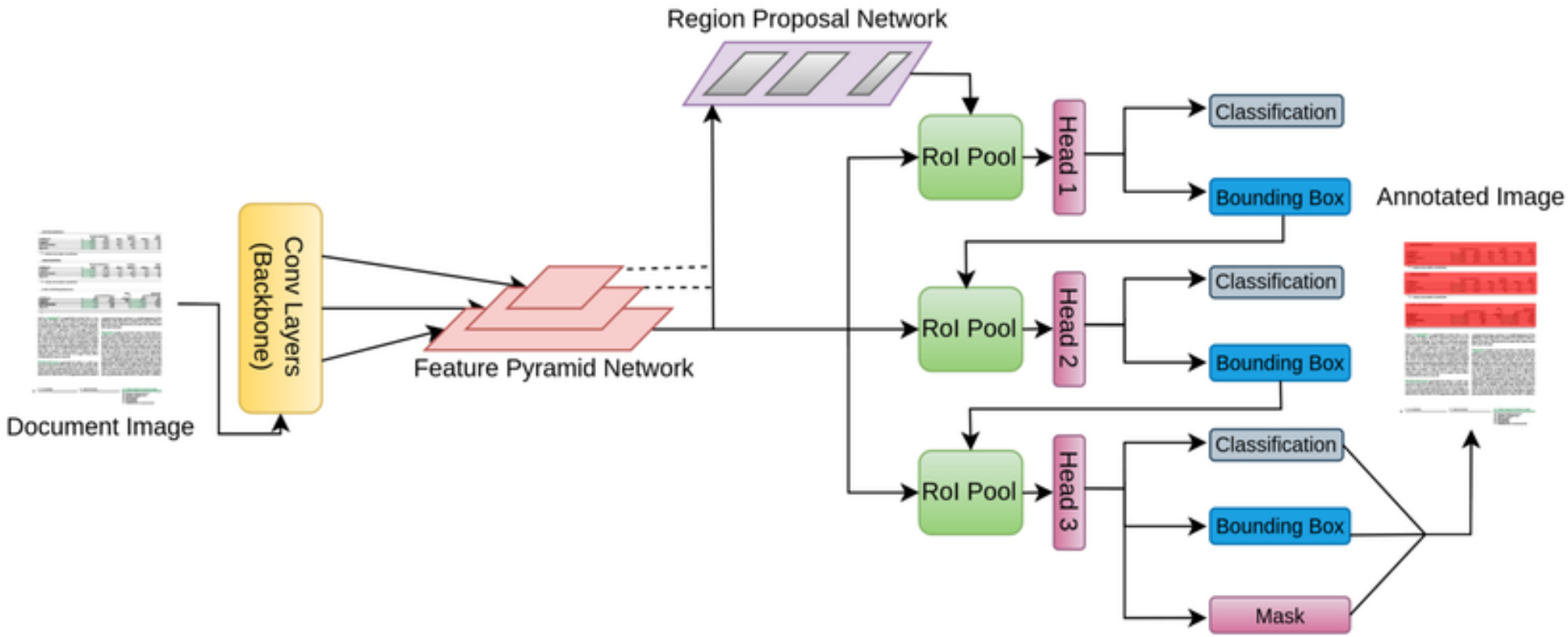
Klasa	map	mar_100
Player	0.6825	0.7499
Ball	0.4247	0.5196
Goal Line	0.7081	0.7484
Field	0.9358	0.9470
Background	0.9149	0.9444
Referee	0.6076	0.7027
Football Pitch Line	0.6549	0.7358

- Na nepoznatim instancama:



- Možemo primetiti da za dosta objekata koji nisu klase sudija, predvidja da to jesu što nam govori da model ima prostora za poboljšanje.

Kaskadni Mask RCNN (Detectron)



Detectron2

- **Detectron** je open-source softverski paket razvijen od strane Facebook AI Research tima, namenjen za implementaciju algoritama za detekciju objekata, segmentaciju slika i druge zadatke iz oblasti kompjuterskog vida.

Optuna

- Prvo smo preko **Optuna** biblioteke pronašli najbolje parametre na trening skupu testirajući ih na validacionom skupu.
- Optuna radi na način da se zada veliki broj hiperpametara i broj pokušaja. Ona u svakom pokusaju pokušava da predvidi najbolje parametre preko odredjenih heuristika i rezultata iz prethodnog treniranja.

Greske

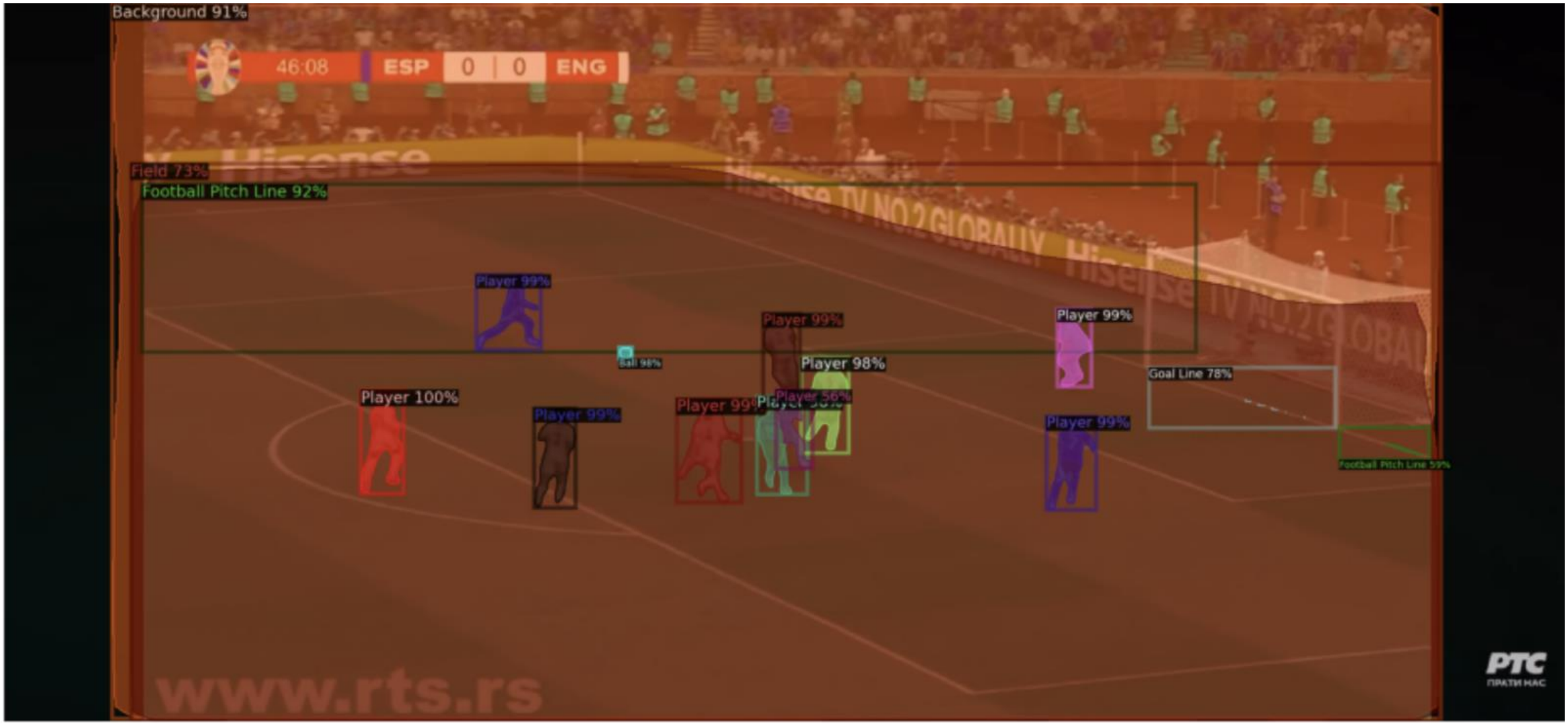
- Bounding box (bbox):
- Ukupni AP: 76.35
- AP50: 91.44
- AP75: 84.44
- AP za male objekte (APs): 43.40
- AP za srednje objekte (APm): 83.25
- AP za velike objekte (APl): 76.74

- Specifčne klase:
- Igrač (AP-Player): 80.07
- Lopta (AP-Ball): 41.16
- Gol-linija (AP-Goal Line): 78.59
- Teren (AP-Field): 100.0
- Pozadina (AP-Background): 79.14
- Sudija (AP-Referee): 72.31
- Linija terena (AP-Football Pitch Line): 83.15

- Segmentacija (segm):

- Ukupni AP: 48.53
- AP50: 64.38
- AP75: 52.37
- AP za male objekte (APs): 12.02
- AP za srednje objekte (APm): 30.35
- AP za velike objekte (APl): 69.28

- Specifične klase:
- Igrač (AP-Player): 66.28
- Lopta (AP-Ball): 41.28
- Gol-linija (AP-Goal Line): 0.0
- Teren (AP-Field): 100.0
- Pozadina (AP-Background): 77.14
- Sudija (AP-Referee): 55.01
- Linija terena (AP-Football Pitch Line): 0.0


Background 91%
46:08
ESP
0
0
ENG
Field 73%
Football Pitch Line 92%
Player 99%
Ball 98%
Player 99%
Player 98%
Player 99%
Goal Line 78%
Player 100%
Player 99%
Player 99%
Player 99%
Football Pitch Line 59%
www.rts.rs
РТС
ПРАТИ НАС

YOLOv8