Detekcija lica korištenjem modela umjetne inteligencije

Čarapina, Arian

Undergraduate thesis / Završni rad

2023

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Rijeka, Faculty of Engineering / Sveučilište u Rijeci, Tehnički fakultet**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:190:345197

Rights / Prava: Attribution 4.0 International/Imenovanje 4.0 međunarodna

Download date / Datum preuzimanja: **2024-05-17**

Repository / Repozitorij:

Repository of the University of Rijeka, Faculty of Engineering

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Prijediplomski sveučilišni studij elektrotehnike

Završni rad

Detekcija lica korištenjem modela umjetne inteligencije

Rijeka, rujan 2023.

Arian Čarapina

00690090620

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Prijediplomski sveučilišni studij elektrotehnike

Završni rad

Detekcija lica korištenjem modela umjetne inteligencije

Mentor: Prof. dr. sc. Zlatan Car

Rijeka, rujan 2023.

Arian Čarapina

00690090620

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
POVJERENSTVO ZA ZAVRŠNE ISPITE

Rijeka, 14. ožujka 2023.

Zavod: **Zavod za automatiku i elektroniku**
Predmet: **Automatizacija**
Grana: **2.03.06 automatizacija i robotika**

ZADATAK ZA ZAVRŠNI RAD

Pristupnik: **Arian Čarapina (0069090620)**
Studij: Sveučilišni prijediplomski studij elektrotehnike

Zadatak: **Detekcija lica korištenjem modela umjetne inteligencije**

Opis zadatka:

Predstaviti problem detekcije lica i komentirati primjene. Razviti model zasnovan na umjetnoj inteligenciji za prepoznavanje lica, temeljen na javno dostupnom setu podataka. Usporediti performanse modela sa sustavom za prepoznavanje lica koji nije temeljen na umjetnoj inteligenciji. Komentirati rezultate, prednosti i nedostatke ovakvog sustava.

Rad mora biti napisan prema Uputama za pisanje diplomskih / završnih radova koje su objavljene na mrežnim stranicama studija.

Zadatak uručen pristupniku: 20. ožujka 2023.

Mentor:

Prof. dr. sc. Zlatan Car

Predsjednik povjerenstva za završni ispit:

Prof. dr. sc. Dubravko Franković

IZJAVA O SAMOSTALNOJ IZRADI ZAVRŠNOG RADA

Sukladno članku 7. Pravilnika o završnom radu, završnom ispitu i završetku sveučilišnog preddiplomskog studija Tehničkog fakulteta Sveučilišta u Rijeci izjavljujem da sam izradio ovaj završni rad samostalno, koristeći vlastito znanje i navedenu literaturu u razdoblju od 14.03.2023. do 12.09.2023.

Rijeka, 02.09.2023.

Arian Čarapina

ZAHVALA

Ovim putem želim se zahvaliti svom mentoru prof. dr. sc. Zlatanu Caru te asist. Sandiju Baressi Šegoti na svom znanju i podršci koju su mi pružali tijekom mog školovanja i izrade završnog rada. Također se želim zahvaliti i svojoj obitelji i prijateljima na neprestanoj podršci i razumijevanju tijekom mog dosadašnjeg školovanja.

Sadržaj

1. UVOD 1
2. MATERIJALI I METODE 2
   2.1 Detekcija lica OpenCV 2
      2.1.1. Opis OpenCV koda 3
   2.2.YOLOv5 5
      2.2.1. Opis YOLOv5 koda 6
   2.3. WIDER Dataset 18
   2.4. Metrike 20
3. REZULTATI 23
   3.1. Analiza rezultantnih metrika 26
   3.2. Usporedba rezultata prepoznavanja deterministčkim algoritmom i algoritmom temeljenim na umjetnoj inteligenciji 30
4. ZAKLJUČAK 32
5. LITERATURA 33
6. POPIS SLIKA 34
7. SAŽETAK 35

1. Uvod

Pojam umjetne inteligencije po prvi put se javlja u 20. st. kada je Alan Mathison Turing osmislio Turingov test kojim je htio ispitati inteligenciju nekog stroja. Danas je ona postala sveprisutna te nezamjenjiv dio modernog svijeta koju možemo pronaći u gotovo svim granama znanosti, pa i u medicini, gdje uz pomoć umjetne inteligencije možemo lakše dijagnosticirati pojedine bolesti. Prema definiciji umjetna inteligencija je dio računalstva koji se bavi razvojem sposobnosti računala da obavlja zadaće za koje je potreban neki oblik inteligencije. Cilj ovog završnog rada je predstaviti problem detekcije lica i komentirati moguće primjene. Razvijeni model zasnovan je na umjetnoj inteligenciji, te je treniran na javnom dostupnom setu podataka. Također će biti uspoređene performanse modela sa sustavom za prepoznavanje lica koji nije temeljen na umjetnoj inteligenciji, te će naposljetku biti prikazani dobiveni rezultati, prednosti i nedostaci ovakvog sustava. Sve analize i usporedbe koje su se provodile tokom ovog rada su neophodne kako bi se znalo u kojim situacijama je pogodnije primijeniti sustav zasnovan na umjetnoj inteligenciji, a u kojim je pogodan sustav koji nije temeljen na umjetnoj inteligenciji. Za model prepoznavanja lica koji nije temeljen na umjetnoj inteligenciji koristila se knjižnica OpenCV, dok je za izradu modela temeljenog na umjetnoj inteligenciji korišten YOLOv5 model računalnog vida. Moguće primjene ovakvih sustava možemo naći svugdje u svijetu od medicine, npr. detekciju lica osoba u javnosti koja su zaražena COVID-19 virusom, zaštite, npr. detekcija lica traženih kriminalaca, pa čak i u školstvu, npr. zabilježiti prisutnost studenata. Jedan od aktualnijih primjena ove tehnologije pronalazimo u Kini. Prednost ove tehnologije prepoznala je i tamošnja vlast te je uvela sistem socijalnog kredita koji predstavlja ponašanje pojedinog građana te države. Budući da je Kina druga u svijetu po broju nadzornih kamera po osobi, kontrola i upravljanje populacije uz pomoć ove tehnologije postaje lakše. Umjetna inteligencija omogućava detekciju lica te prepoznavanje identiteta bilo kojeg građanina Kine, te ovisno o njihovom ponašanju povećava odnosno smanjuje njihov socijalni kredit.

2. Materijali i metode

Unutar ovog poglavlja opisani su kodovi koji su se koristili pri izradi modela za prepoznavanje lica. Prvi kod temeljen je na klasičnom programiranju dok je drugi kod temeljen na umjetnoj inteligenciji. Kroz ovo poglavlje obrađen je i javno dostupan skup podataka „WIDER dataset“, namijenjen za prepoznavanje lica. Unutar zadnjeg potpoglavlja predstavljene su određene metrike koje su potrebne za razumijevanje dobivenih rezultata.

2.1 Detekcija lica OpenCV

Za izradu sustava za detekciju lica koji nije temeljen na umjetnoj inteligenciji korištena je knjižnica OpenCV. OpenCV (eng. Open Source Computer Vision Library) je velika knjižnica slobodno dostupnog koda koja se koristi pri obradi slike, strojnom učenju te podržava programske jezike poput Python, C++, Java itd. Knjižnica se sastoji od 2500 algoritama te se primjenjuje u brojnim poznatim tvrtkama poput: Google, Yahoo, Microsoft, Intel, IBM i mnogim drugima. OpenCV knjižnica našla je široku primjenu u raznim područjima ljudske djelatnosti, od kojih je zanimljivo istaknuti sljedeće primjene: detekcija utapanja ljudi u bazenima, pomaganje navigaciji kretanja robota, kontrola postojanja krhotina na pistama i slično. Kako bi se potpuno realizirao potencijal ove knjižnice važno ju je spariti i s još knjižnica, poput Nupmy koja omogućava da se ubrzaju određeni matematički algoritmi, i time koja omogućava da se vrijeme u Python kodu predstavi kao objekt, broj ili niz znakova tj. string.

2.1.1 Opis OpenCV koda

```
import cv2
import numpy as np
import time
i = 0
```

U prvom dijelu koda potrebno je unijeti potrebne knjižnice koje će se koristiti tokom koda, a to se čini uz pomoć naredbe import gdje se unose knjižnice cv2, numpy i time. U drugom redu može se primijetiti da uz naredbu import nupmy stoji i naredba as np koja olakšava pozivanje knjižnice numpy tako da se umjesto numpy piše samo np. Također, stvorena je i varijabla i, kojoj je dodijeljena vrijednost nula.

```
faca_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'/haarcascade_frontalface_def.xml')
oci_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'/haarcascade_eye.xml')
cap = cv2.VideoCapture(0)
```

U drugom dijelu koda potrebno je definirati Haar Cascade za oči i lice. Haar Cascade jedan je od algoritama koji služi za prepoznavanje lica i očiju, no također se može koristiti i za prepoznavanje bilo kakvog drugog objekta npr. tablice od auta. Haar Cascade definiran je prvom i drugom linijom koda gdje je prva linija za facu, a druga za oči. Pri tome je važno omogućiti da kod ima izvor slike, a upravo to omogućava treća linija koda koja varijabli cap dodjeljuje izlaz iz računala 0 na kojem je bila spojena kamera. Valja napomenuti da promjenom broja kod naredbe cv2.VideoCapture() mijenja se izvor od kuda kod dobiva sliku na kojoj traži zadani objekt.

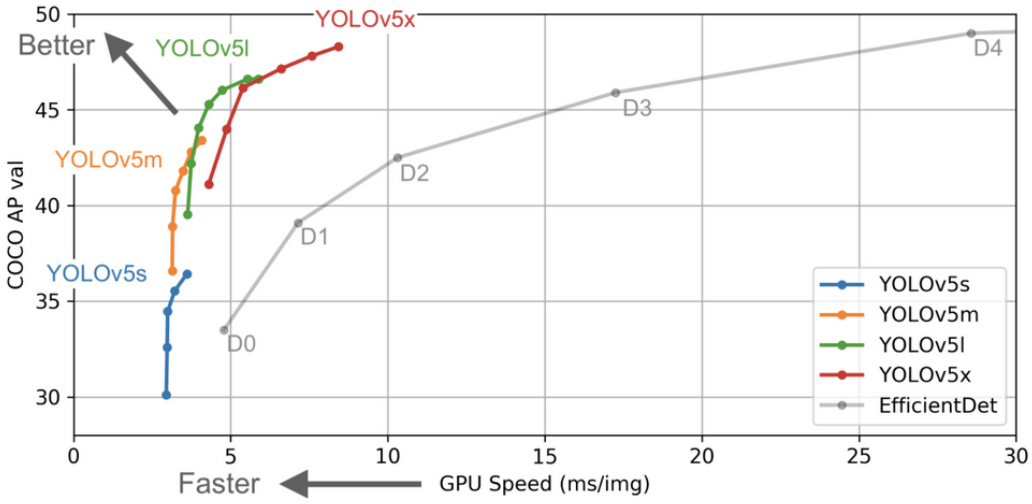
```
while True:
          ret, img = cap, cap
          gray=cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
          face = faca_cascade.detectMultiScale(gray, 1.3, 5)
          for (x,y,w,h) in face:
                    cv2.rectangle(img, (x,y), (x+w, y+h), (255,0,0), 2)
                    roi_gray = gray[y:y+h,x:x+w]
                    roi_color = img[y:y+h,x:x+w]
                    oci = oci_cascade.detectMultiScale(roi_gray)
                    for (ex,ey,ew,eh)    in oci:
                              cv2.rectangle(roi_color, (ex,ey), (ex+ew,
ey+eh), (0,255,0), 2)
          cv2.imshow('img', img)
          k = cv2.waitKey(30)& 0xff
          if k ==27:
              break


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

Sada kada su uvedene sve potrebne knjižnice i definirane sve potrebne varijable, prelazi se na treći i najvažniji dio koda. On započinje while petljom (prvi red koda) koja traje sve dok se ne prekine pritiskom na tipku ESC na tipkovnici. Unutar petlje definira se varijabla img te joj se dodjeljuje vrijednost ulaza kamere s funkcijom read(). Varijabla Gray predstavlja sliku u crno-bijeloj boji te ona pridonosi lakšoj obradi slike. Varijabla Gray postiže se na način da se prijašnja slika, koja je definirana s img i koja je u boji, pretvori u crno-bijelu sliku uz pomoć funkcije cv2.cvtColor(img,cv2.COLOR_BGR2GRAY). Funkcija face_cascade.detectMultiScale() služi za detekciju lica te unutar sebe ima određene parametre koji se podešavaju ovisno o vjerojatnosti pronalaska lica. Za crtanje pravokutnika na licima i očima koristimo funkciju cv2.rectangle(img, (x,y), (x+w, y+h)) u kojoj je (x,y) koordinata 1. točke pravokutnika, a (x+w, y+h) 2. točke pravokutnika, dok ostali parametri služe samo za definiranje boje i debljine linije pravokutnika. Također, može se primijetiti da se u kodu nalaze i roi funkcije uz pomoć kojih se definira prostor na kojem program može pronaći sliku tj. isključuje se mogućnost da program pronalazi oči van granica lica. Zatim treba ponoviti iste funkcije koje su korištene za detekciju lica, samo uz modifikaciju da detektiraju oči. Naravno, potreban je i prikaz slike pa se za to koristi funkcija cv2.imshow().

2.2 YOLOv5

Za izradu sustava za prepoznavanje lica temeljen na umjetnoj inteligenciji korišten je YOLOv5 model računalnog vida koji je peta verzija YOLO (eng. You Only Look Once) algoritma. YOLOv5 se obično koristi za detekciju objekta na slikama te dolazi u četiri glavne verzije: small(s), medium(m), large(l) i extra large (x). Verzije se međusobno razlikuju u preciznosti prepoznavanja te svaki od navedenih modela ima drugačije vrijeme treniranja. Sa slike 2.1. može se uočiti da što je verzija „veća“ to je potrebno duže vrijeme treniranja, no zauzvrat se dobije precizniji model detekcije. Važno je odabrati model koji će zadovoljiti potrebe ovog zadatka pa je u ovom radu korišten YOLOv5s model zbog malog vremena treniranja te dovoljne preciznosti. Razlozi odabira YOLOv5 modela su: jednostavna instalacija, brzo treniranje, lakša manipulacija pojedinih slika, skupina slika te intuitivni i lagan način navigiranja datotekama kod treniranja.



Slika 2.1. Prikaz usporedbe veličine i brzine treniranja različitih YOLOv5 modela

2.2.1 Opis YOLOv5 koda

```
from google.colab import drive
drive.mount("/content/gdrive")
```

Prije početka izrade modela detekcije lica na bazi umjetne inteligencije potrebno je osigurati mjesto gdje će se spremati datoteke i podaci. Za izradu ovog modela kao mjesto za spremanje podataka koristio se Google Drive radi lakše korekcije grešaka, kao i radi mogućnosti korištenja Google programa za pisanje koda Collab. Funkcijom from iz knjižnice google colab unosi se u kod funkcija drive koja se u sljedećem redu poziva. Funkciju drive.mount() koristi se radi spremanja svih datoteka i podataka s kojima će se raditi tokom izrade sustava detekcije na google drive disk.

```
%cd /content/gdrive/MyDrive
!pwd
```

Funkcijom %cd definira se put spremanja podataka, te je u ovom slučaju to Google drive s imenom MyDrive. !pwd naredba služi za provjeru adrese spremanja podataka te ona izbacuje adresu /content/gdrive/MyDrive što odgovara željenoj adresi spremanja podataka

```
import os

if not os.path.isdir("Projekt"):
  os.makedirs("Projekt")
%cd Projekt
```

Zatim se unosi knjižnica os u kod te se sa if naredbom kreira datoteka pod imenom Projekt u kojoj će biti spremljen model detekcije lica. Ponovno se koristi naredba %cd kako bi se prilagodila adresa spremanja podataka koja će sadržavati datoteku Projekt.

```
!wget
http://mmlab.ie.cuhk.edu.hk/projects/WIDERFace/support/bbx_annotation/w
ider_face_split.zip  # pomoću funkcije "wget" skidamo potrebne datoteke
za naš program
!unzip -qx /content/gdrive/MyDrive/Projekt/archiveee.zip    #
raspakiravamo Wider skup podataka
```

U ovom dijelu koda koristi se funkcija !wget kako bi se sa željene adrese skinule datoteke koje će biti potrebne pri izradi ovoga modela. Također, na disku MyDrive nalazi se datoteka archivee.zip koja je preuzeta sa stranice Kaggle, a u njoj se nalazi WIDER skup podataka tj. sve slike lica na kojima će se model trenirati. Kako bi se koristila navedena datoteka, potrebno ju je prvo raspakirati te se to čini uz pomoć funkcije !unzip.

```
import os
import numpy as np
import matplotlib.pyplot as plt
from glob import glob as g
import cv2
from tqdm.notebook import tqdm
from shutil import copy, move
from google.colab.files import download
```

Kako bi se olakšali neki postupci u ovom kodu, potrebno je unijeti određene knjižnice. Unošenje knjižnica u projekt vrši se uz pomoć naredbe import. Neke od knjižnica koje su uvedene su: numpy, radi olakšavanja izvođenja određenih matematičkih algoritama, matplotlib.pyplot, koja služi za prikazivanje 2D grafova, cv2, koja olakšava obradu slika itd.

```
# Val Dataset
new_imgs_dir = '/content/gdrive/MyDrive/Projekt/newDataset/images/val'
new_lbls_dir = '/content/gdrive/MyDrive/Projekt/newDataset/labels/val'
label_text_name =
'/content/gdrive/MyDrive/Projekt/wider_face_split/wider_face_split/wide
r_face_val_bbx_gt.txt'
imgs_address =
"/content/gdrive/MyDrive/Projekt/WIDER_val/WIDER_val/images"

os.makedirs(new_imgs_dir,exist_ok = True)
os.makedirs(new_lbls_dir,exist_ok = True)
annots = open(label_text_name)
lines = annots.readlines()
names =   [x for x in lines if 'jpg' in x]
indices = [lines.index(x) for x in names]
```

Definiranje puta datoteka, kao što je to prikazano u gornjem dijelu koda omogućuje manipuliranje slikama i vrijednostima koje su potrebne.

```
for n in tqdm(range(len(names
    i = indices[n]
    name = lines[i].rstrip()
    old_img_path = os.path.join(imgs_address , name)
    name = name.split('/')[-1]
    label_path = os.path.join(new_lbls_dir , name.split('.')[0] +
'.txt')
    img_path = os.path.join(new_imgs_dir , name)

    num_objs = int(lines[i+1].rstrip
    bboxs = lines[i+2 : i+2+num_objs]
    bboxs = list(map(lambda x:x.rstrip() , bboxs
    bboxs = list(map(lambda x:x.split()[:4], bboxs
```

Započinje se for petlja koja traje sve dok se sve slike i oznake za validacijske datoteke ne obrade. Također, koristi se funkcija za prikaz napretka programa.

```
img = cv2.imread(old_img_path
    img_h,img_w,_ = img.shape
    img_h,img_w,_ = img.shape
    f = open(label_path, 'w')
    count = 0
    for bbx in bboxs
        x1 = int(bbx[0])
        y1 = int(bbx[1])
        w = int(bbx[2])
        h = int(bbx[3])
        x = (x1 + w//2) / img_w
        y = (y1 + h//2) / img_h
        w = w / img_w
        h = h / img_h
        if w * h * 100 > 2:
            yolo_line = f'{0} {x} {y} {w} {h}\n'
            f.write(yolo_line)
            count += 1
    f.close()
    if count > 0:
        copy(old_img_path , img_path)
    else:
        os.remove(label_path)
```

Kako bi model mogao raditi, nužno mu je zadati potrebne slike, odnosno materijale na kojima će on trenirati. Navedeni materijali se dijele na validacijske i za treniranje. Validacijske slike koriste se kako bi se testiralo koliko je model uspješan u prepoznavanju zadanog objekta, dok slike za treniranje služe kako bi model usavršio prepoznavanje zadanih objekta, u ovom slučaju lica. U ovom dijelu koda kreiraju se validacijske datoteke. Validacijske datoteke osim što imaju slike iz kojih model uči, također moraju imati i oznaku (eng. label) koja se sastoji od: vrste slike koju program prepoznaje(class), x i y koordinate centra slike te visine i širine slike. Spomenute se vrijednosti kreću od 0-1 pa primjer oznake određene slike može izgledati kao sa slike 2.2.

```
0 0.916015625 0.2936857562408223 0.1240234375 0.20411160058737152
```

Slika 2.2. Primjer oznake slike

```
# Train Dataset
new_imgs_dir =
'/content/gdrive/MyDrive/Projekt/newDataset/images/train'
new_lbls_dir =
'/content/gdrive/MyDrive/Projekt/newDataset/labels/train'
label_text_name =
'/content/gdrive/MyDrive/Projekt/wider_face_split/wider_face_split/wide
r_face_train_bbx_gt.txt'
imgs_address =
'/content/gdrive/MyDrive/Projekt/WIDER_train/WIDER_train/images'

os.makedirs(new_imgs_dir,exist_ok = True)
os.makedirs(new_lbls_dir,exist_ok = True)
annots = open(label_text_name)
lines = annots.readlines()
names =   [x for x in lines if 'jpg' in x]
indices = [lines.index(x) for x in names]
```

Povezuju se adrese datoteka s novo definiranim varijablama što omogućuje njihovo daljnje korištenje u kodu.

```
for n in tqdm(range(len(names[:]))):
    i = indices[n]
    name = lines[i].rstrip()
    old_img_path = os.path.join(imgs_address , name)
    name = name.split('/')[-1]
    label_path = os.path.join(new_lbls_dir , name.split('.')[0] +
'.txt')
    img_path = os.path.join(new_imgs_dir , name)
    num_objs = int(lines[i+1].rstrip())
    bboxs = lines[i+2 : i+2+num_objs]
    bboxs = list(map(lambda x:x.rstrip() , bboxs))
    bboxs = list(map(lambda x:x.split()[:4], bboxs))
    # if len(bboxs) > 5:
    #     continue
    img = cv2.imread(old_img_path)
    img_h, img_w, _ = img.shape
    f = open(label_path, 'w')
    count = 0 # Num of bounding box
```

For petljom se započinju obrađivati slike, te kao i u kodu za validacijske datoteke pozivamo funkciju kako bi se pratio napredak koda.

```
 for bbx in bboxs:
        x1 = int(bbx[0])
        y1 = int(bbx[1])
        w = int(bbx[2])
        h = int(bbx[3])
        x = (x1 + w//2) / img_w
        y = (y1 + h//2) / img_h
        w = w / img_w
        h = h / img_h
        if w * h * 100 > 2:
            yolo_line = f'{0} {x} {y} {w} {h}\n'
            f.write(yolo_line)
            count += 1
f.close()
    if count > 0:
        copy(old_img_path , img_path)
    else:
        os.remove(label_path)
```

Kao i u prijašnjem dijelu koda gdje su kreirane validacijske datoteke za slike i oznake, tako se i u ovom dijelu koda kreiraju datoteke slika i oznaka za treniranje modela. Važno je napomenuti da se ne nalazi isti broj slika u validacijskoj datoteci i datoteci za treniranje te omjer tih slika ne mora uvijek biti isti, već varira kakvi se rezultati žele postići.

```
def resize_img(input_name , output_name, target_width = 640):
    im = cv2.imread(input_name)
    h,w,_  = im.shape
    target_height = int(h / w * target_width)
    im = cv2.resize(im , (target_width , target_height), interpolation =
cv2.INTER_AREA
    cv2.imwrite(output_name , im
def resize_all_imgs(imgs_dir):
    names = g(os.path.join(imgs_dir , '*'))
    for img in tqdm(names):
        resize_img(img, img)
```

Kako bi YOLOv5 model mogao koristiti slike koje su mu ponuđene, potrebno ih je prebaciti u odgovarajući format tj. potrebno je da širina slike iznosi 640px. Prema mjeri širine potrebno je skalirati slike. Prva funkcija u ovom dijelu koda bavi se skaliranjem slike resize_img(), dok druga funkcija resize_all_imgs() poziva prijašnju funkciju da se izvrši na svim slikama u svim datotekama.

```
names = g('/content/gdrive/MyDrive/Projekt/newDataset/labels/*/*')
print(f'Threre are {len(names)}  images')
```

Zatim se kreira varijabla names kako bi se provjerilo s kojom količinom slika model raspolaže, a to je u ovom slučaju 5253 slike.

```
resize_all_imgs('/content/gdrive/MyDrive/Projekt/newDataset/images/*'
```

Poziva se funkcija za skaliranje slika `resize_all_imgs`() koja slike pretvara u željeni format.

```
n = np.random.randint(0, len(names
f = open(names[n

lines = f.readlines()
```

Nakon završenog skaliranja, provjeravaju se dimenzije oznaka slika uz pomoć kreiranja varijable lines.

```
lines
```

Nakon definiranja varijable lines, poziva se na način prikazan u gornjem dijelu koda. Kao rezultat je dobivena slika 2.3. koja odgovara željenom formatu.

```
['0 0.4013671875 0.37734375 0.40234375 0.46640625\n']
```

Slika 2.3. Primjer oznake slike

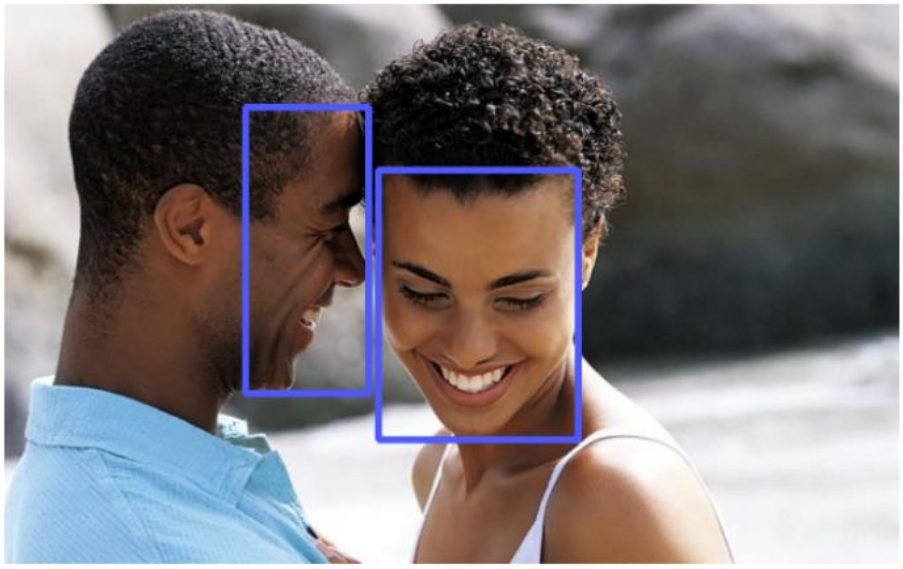
```
n = np.random.randint(0, len(names))
f = open(names[n])

lines = f.readlines()
classes = list(map(lambda x: int(x[0]), lines))
lines = list(map(lambda x:x.rstrip()[2:], lines))
objects = list(map(lambda x:(x.split()), lines))

img = cv2.imread(names[n].replace('txt','jpg').replace('labels',
'images'))
for c, bbox in zip(classes, objects):
  bbox = list(map(lambda x:float(x), bbox))
  x,y,w,h = bbox
  img_h = img.shape[0]
  img_w = img.shape[1]
  x = int(x * img_w)
  w = int(w * img_w)
  y = int(y * img_h)
  h = int(h * img_h)
  color = (255,100,50)
  cv2.rectangle(img , (int(x-w/2), int(y-h/2)), (int(x+w/2),
int(y+h/2)), color , 4)
plt.figure(figsize = (8,8))
plt.imshow(img[:,:,::-1]); plt.axis('off')
print(f'number of bounding boxes : {len(classes)}')
print(f'Shape on the image : {img.shape}')
```

Prije početka treniranja nužno je proučiti kako YOLOv5 model označava prepoznata lica. Na slici 2.4. dobiven je izlaz za slučajno odabranu sliku. Kao rezultat, program na ovoj slici detektira dva lica, te istodobno ispisuje dimenzije slike.

```
number of bounding boxes : 2
Shape on the image : (400, 640, 3)
```



Slika 2.4. Primjer označavanje slike iz datoteke Dataset

```
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1 import ImageGrid
import numpy as np

images = []
for _ in range(25):
    n = np.random.randint(0, len(names))
    f = open(names[n])

    lines = f.readlines()
    classes = list(map(lambda x: int(x[0]), lines))
    lines = list(map(lambda x:x.rstrip()[2:], lines))
    objects = list(map(lambda x:(x.split()), lines))
```

Uvode se knjižnice kako bi se olakšalo izvođenje određenih dijelova koda te se dodjeljuju vrijednosti varijablama classes, lines, objects.

```
img = cv2.imread(names[n].replace('txt','jpg').replace('labels',
'images'))
    for c, bbox in zip(classes, objects):
        bbox = list(map(lambda x:float(x), bbox))
        x,y,w,h = bbox
        img_h = img.shape[0]
        img_w = img.shape[1]
        x = int(x * img_w)
        w = int(w * img_w)
        y = int(y * img_h)
        h = int(h * img_h)
        color = (255,100,50)
        cv2.rectangle(img , (int(x-w/2), int(y-h/2)), (int(x+w/2),
int(y+h/2)), color , 6)
    images.append(img[:,:,::-1])
fig = plt.figure(figsize=(16., 16.))
grid = ImageGrid(fig, 111,
                 nrows_ncols=(5 ,5),
                 axes_pad=0.1,  .
                 )

for ax, im in zip(grid, images):
    ax.imshow(im)
    ax.axis('off')

plt.show()
```

Kako bi se dobio veći uzorak slika na kojima se testira rad prepoznavanja, potrebno je napraviti mrežu slika na kojoj su prikazane sve slike. (Slika 2.5.) Ova mreža se sastoji od pet redova i pet stupaca.

Slika 2.5. Primjer označene mreže slika

```
fig.savefig('grid_output.png')
download('grid_output.png')
```

Gore prikazane funkcije služe za spremanje i skidanje dobivene mreže slika.

```
os.makedirs('/content/gdrive/MyDrive/Projekt/Yolo/images', exist_ok=
True)
os.makedirs('/content/gdrive/MyDrive/Projekt/Yolo/labels', exist_ok=
True)
os.makedirs('/content/gdrive/MyDrive/Projekt/Yolo/images/train',
exist_ok= True)
os.makedirs('/content/gdrive/MyDrive/Projekt/Yolo/images/val',
exist_ok= True)
os.makedirs('/content/gdrive/MyDrive/Projekt/Yolo/labels/train',
exist_ok= True)
os.makedirs('/content/gdrive/MyDrive/Projekt/Yolo/labels/val',
exist_ok= True)
```

Nakon što se dobije uvid u način na koji će model označiti lice kada ga prepozna, potrebno je slike za treniranje i validaciju smjestiti u odgovarajuće datoteke od YOLOv5 modela. Gore prikazane linije koda služe kako bi se kreirale potrebne datoteke za spremanje slika i za skidanje prethodno dobivene mreže slika.

```
import torch
from IPython.display import clear_output
torch.cuda.get_device_name()
```

Zatim je potrebno u program uvesti knjižnicu torch koja pruža određene alate potrebne za treniranje modela.

```
!git clone https://github.com/ultralytics/yolov5.git
!pip install -qr /content/yolov5/requirements.txt
%cd yolov5
clear_output()
f = open('/content/yolov5/data/dataset.yaml', 'w')
f.write('train:
/content/gdrive/MyDrive/Projekt/newDataset/images/train')
f.write('\nval:
/content/gdrive/MyDrive/Projekt/newDataset/images/val')
f.write('\nnc: {}'.format(1))
f.write("\nnames: ['Face']")
```

Prije početka rada na YOLOv5 modelu potrebno ga je prvo preuzeti s interneta što se čini uz pomoć naredbe git clone. Nakon preuzimanja potrebno je postaviti novu adresu spremanja podataka unutar yolov5 datoteke te se to vrši naredbom %cd. Kako bi model znao gdje se nalaze datoteke s kojima će raditi, potrebno je unijeti potrebne adrese datoteka.

```
!python train.py --img 640 --batch 64 --workers 8 --epochs 40\
  --weights yolov5s.pt\
  --cfg /content/yolov5/models/yolov5s.yaml\
  --data /content/yolov5/data/dataset.yaml\
  --weights yolov5s\
```

Kada su završene sve pripreme za početak treniranja YOLOv5 modela potrebno je započeti treniranje. Naredbom python train započinje se treniranje modela te mu je potrebno zadati parametre prema kojima će on trenirati. Prvi parametar je img koji prikazuje dimenzije slika na kojima će model trenirati, u ovom slučaju širina slike iznosi 640px. Drugi parametar je batch tj. veličina grupe slika s kojom model radi. Treći parametar je workers koji govori koliko CPU-a će model koristiti. Četvrti parametar epochs definira broj epoha trajanja treniranja. Parametar weights definira vrijednosti pojedinih signala. Parametar cfg prikazuje arhitekturu modela koja može biti: yolov5s.yml, yolov5m.yml, yolov5l.yml, yolov5x.yml, no u ovom je slučaju odabran yolov5s.yml. data parametar koji sadrži informacije o skupu podataka (adresa slika, oznaka). Također, postoji i hyp parametar koji sadrži hiperparametre za koje su uzete standardne vrijednosti.

2.3. WIDER Dataset

WIDER FACE Dataset je skup podataka (eng. dataset) koji služi za treniranje, validaciju i testiranje modela temeljenog na umjetnoj inteligenciji. Slike koje se nalaze u WIDER FACE skupu podataka su uzete iz javno dostupnog WIDER skupa podataka. Sastoji se od 32,203 slika na kojima je označeno 393,703 lica. Ovisno o situacijama i događajima na slikama skup podataka je podijeljen u 61 kategoriju, a neke od njih su: ulične tučnjave, ljudi kada glasaju na izborima, ljudi koji spašavaju, policijska akcija, ljudi koji treniranju, ljudi koji se fotografiraju itd.

Slika 2.6. Primjer slike iz kategorije: Raid

Slika 2.7. Primjer slike iz kategorije: Sports_Coach_Trainer

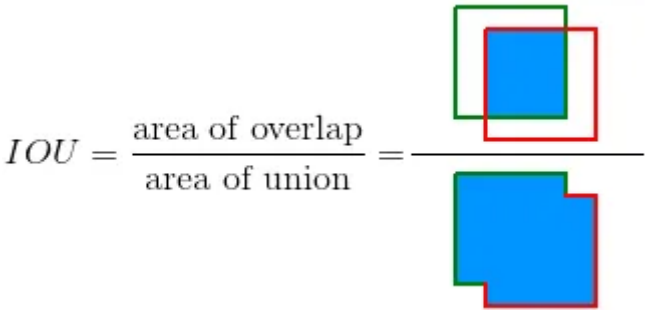
Slika 2.8. Primjer slike iz kategorije: Angler_peoplefishing

Zanimljivo je za primijetiti da se ovaj skup podataka sastoji od lica u raznim situacijama tj. nećemo uvijek imati čisti prikaz lica. Na slici 2.6. mogu se primijetiti policajci čije je lice pokriveno maskama, dok su na slici 2.7. prikazani igrači nogometnog kluba koji slave te kojim ništa ne maskira lica. Također, može se primijetiti da lica neće uvijek gledati direktno u kameru već će ljudi na slikama u većini slučajeva gledati u nekom drugom smjeru. Time se dobivaju slike lica u profilu koje će naš model također naučiti prepoznavati. Doduše, slika 2.8. prikazuje idealnu sliku na kojoj model može trenirati budući da čovjek na slici gleda direktno u kameru. Omjer slika za treniranje i validaciju ne mora biti 50/50 već će u većini slučajeva biti 80/20, 70/30 gdje se dobiva veći broj slika na kojima model trenira. Međutim, omjer slika je proizvoljan te se bira ovisno o rezultatu koji se želi postići. U ovom slučaju omjer slika je otprilike 90/10 gdje 90% čine slike za treniranje a 10% slika za validaciju.

2.4. Metrike

Najvažnija metrika s kojom opisujemo kvalitetu modela baziranog na umjetnoj inteligenciji je mAP (eng. mean Average Precision). Kako bi se što jasnije objasnilo njegovo značenje potrebno je proučiti određene metrike o kojima mAP ovisi.

IOU (eng. Intersection Over Union) definira koliko se tzv. točna površina detektiranog objekta podudara s predviđenom površinom koji je model označio. Formula za izračunavanje ove metrike je jednaka površini preklapanja podijeljenom s ukupnom predviđenom i točnom površinom. (Slika 2.9.)



Slika 2.9. Formula i grafički prikaz izračuna „IOU“ metrike

Metrika precision označava koliko su predikcije dane modelom točne, odnosno broj točno detektiranih objekata od strane modela.

Recall vrijednost govori koliki postotak od ukupnog broja objekata je model prepoznao, npr. ako je na slici 100 lica i model je prepoznao 80, recall parametar će iznositi 80% odnosno 0.8.

Kako bi se pojednostavio izračun tih metrika mogu se definirati još neki pojmovi.

- TP (eng. True positive) označavat će sve objekte koje je model dobro prepoznao.
- TN (eng. True negative) označavat će sve objekte koje je model prepoznao da ne treba označiti.
- FP (eng. False positive) označavat će sve objekte koje je model krivo prepoznao.
- FN (eng. False negative) označavat će sve objekte koje je model zanemario, a trebao je prepoznat.

Nakon definiranja pojmova, izrazi za preciznost i recall metrike glase:

$$Preciznost = \frac{TP}{TP+FP} = \frac{TP}{sve\ detekcije} \tag{2.1}$$

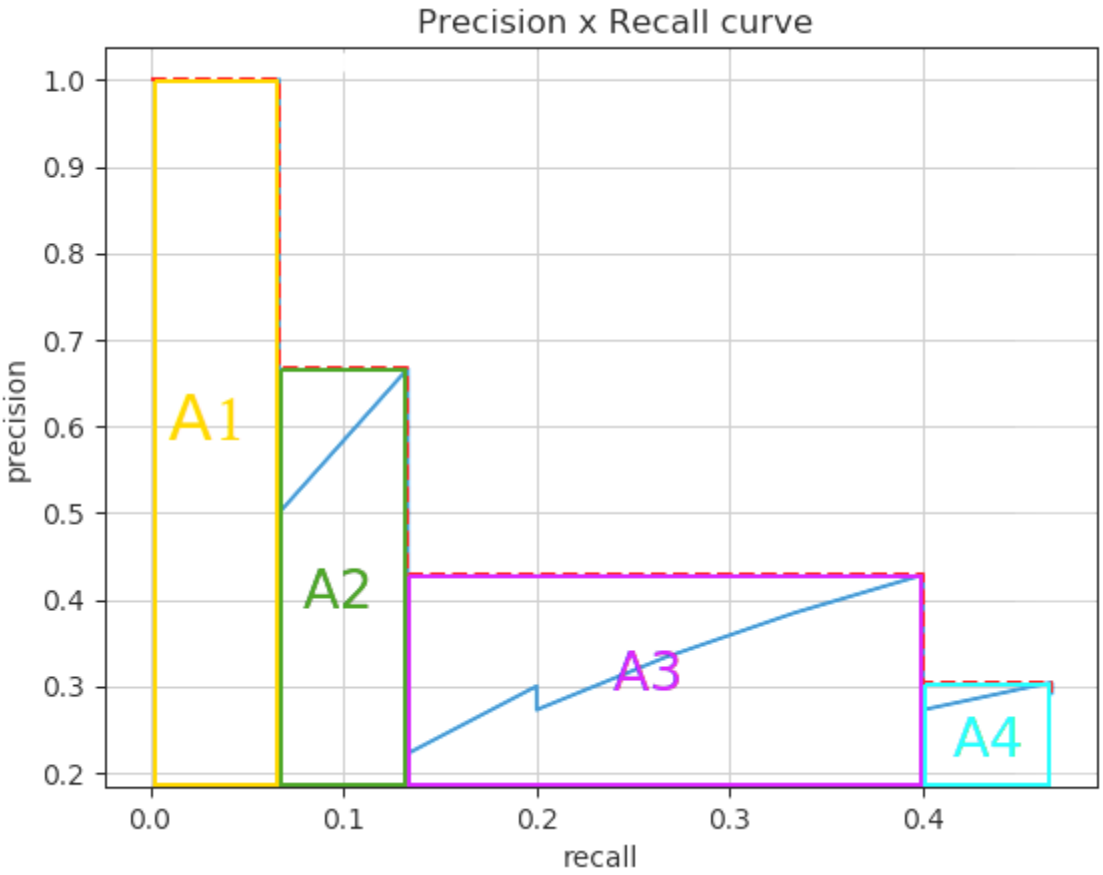
$$Recall = \frac{TP}{TP+FN} = \frac{TP}{sve\ točne\ detekcije} \tag{2.2}$$

Također, bitna je i metrika pouzdanosti (eng. confidence) ovoga modela koji govori koliko je model siguran da je prepoznao točan objekt. Ta metrika se može postaviti kada npr. želimo biti sigurni da će model točno prepoznati objekt.

Slika 2.10. Primjer slike sa metrikom pouzdanosti

Sa slike 2.10. može se primijetiti primjer gdje model prepoznaje lice. Lice je označeno crvenim kvadratom te se uočava da ima i broj koji je dan uz lice. Upravo taj broj definira pouzdanost ovoga modela.

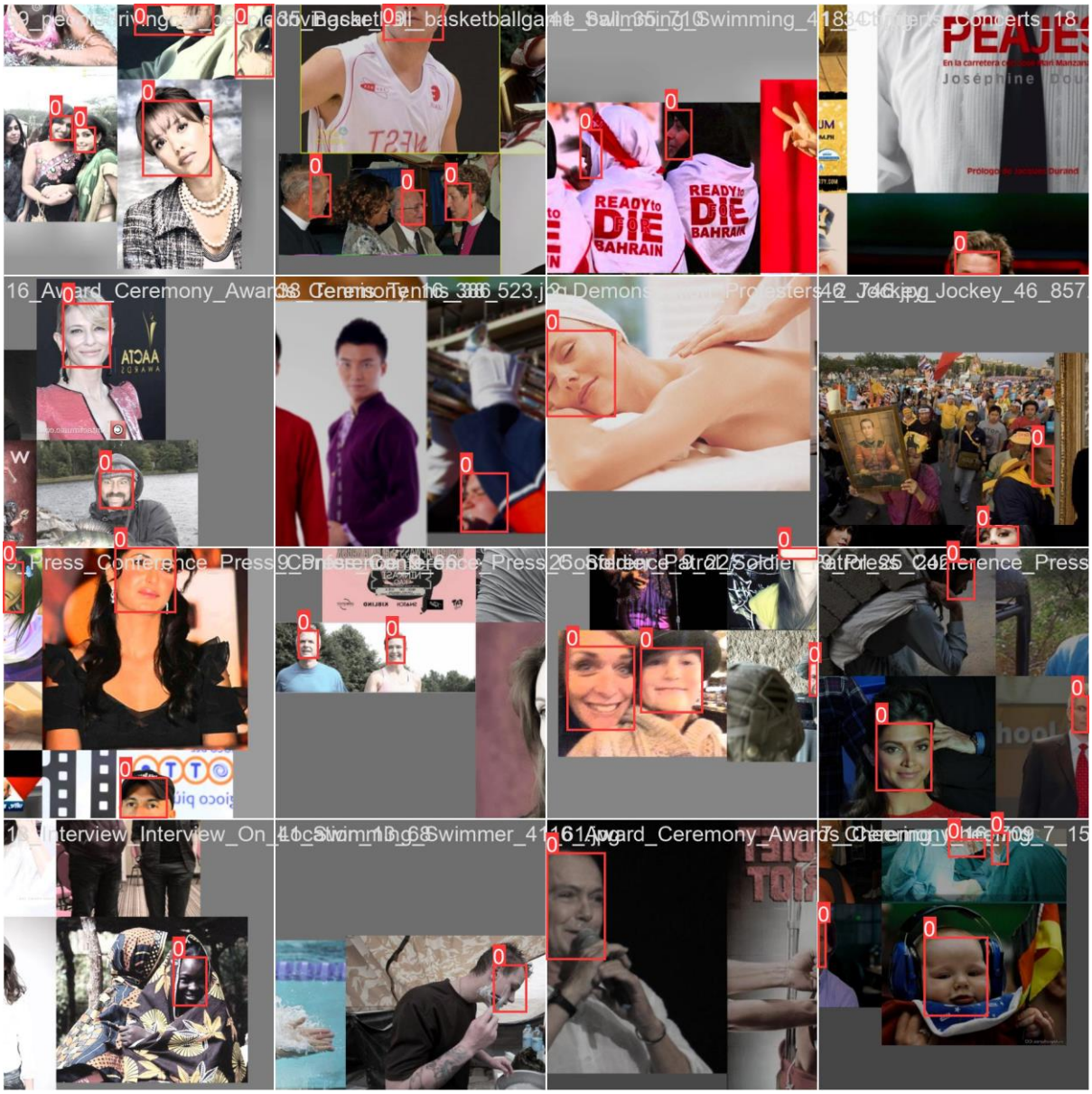
Zadnja metrika koja je važna prije definiranja same mAP metrike je AP. AP (eng. Average precision) metrika prikazuje srednju vrijednost preciznosti modela. Vrijednost AP metrike je jednaka površini ispod „Precision Recall krivulje.mAP“, dok mAP metrika predstavlja srednju vrijednosti AP parametara te je ona glavna metrika koja se promatra kada se žele usporediti dva različita modela.



Slika 2.11. Prikaz izračuna AP metrike pomoću grafa Precision x Recall curve

3. Rezultati

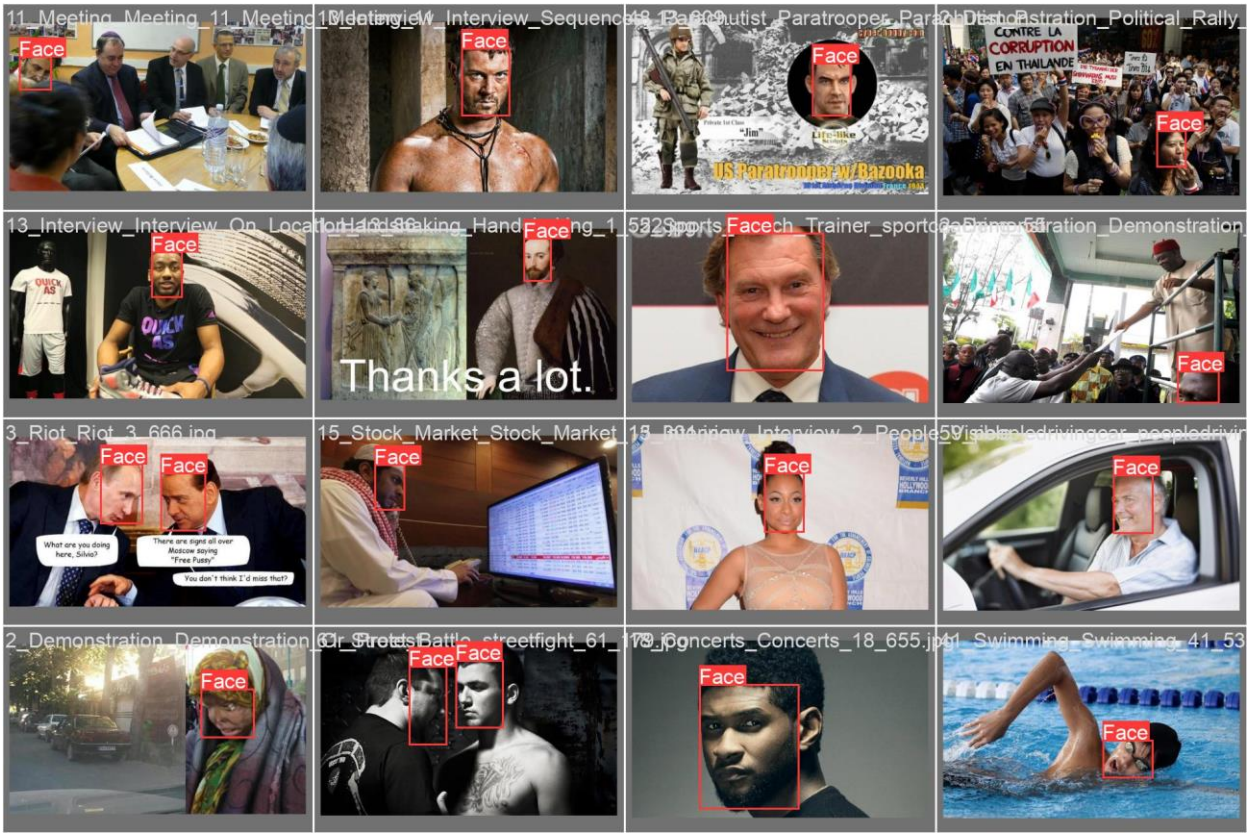
Nakon pokretanja programa model je završio treniranje koje je ukupno trajalo 2:30 sata, nakon čega je dao rezultate treniranja. Prvo što se može vidjeti u rezultatima su slučajne slike na kojima je model trenirao. (Slika 3.1.)



Slika 3.1. Primjer slika na kojima je model trenirao

Na Slici 3.1. može se vidjeti kako je model trenirao, odnosno koliko je lica na slikama prepoznao. Također, mogu se vidjeti i greške našega modela kada nije uspio prepoznati lice.

Nakon treniranja slijedi faza validacije, te se dobiva mreža slika na kojima se testirao model. Prva validacijska slika (Slika 3.2.) predstavlja rješenja sva lica koja je program trebao prepoznati dok druga validacijska slika (Slika 3.3.) predstavlja rezultat koji je naš model detektirao.



Slika 3.2. Primjer validacijskih slika

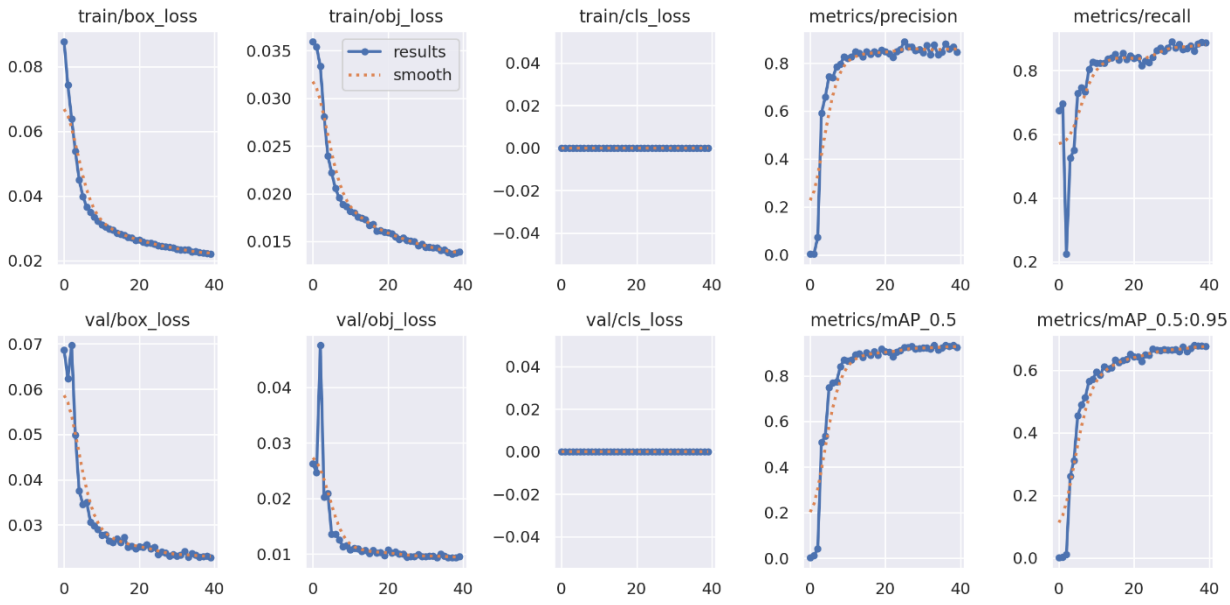
Slika 3.3. Rezultat prepoznavanja validacijskih slika

Važno je uočiti nekoliko grešaka kod ovog dijela testiranja. Prva greška koja se javlja su validacijske slike koje nisu dobro označene, što konkretnije znači da model može dobro prepoznati lice na određenoj slici dok će rezultati ukazati da je model pogriješio. Takva grešku se može ispraviti ako sami radimo svoj skup podataka na kojima će model učiti, no budući da je ovo javno dostupna baza podataka sa slikama, ovakve greške se mogu javiti. Druga greška koja može nastati tokom testiranja je mogućnost detektiranja maske umjesto lica kao što je prikazano na slici 3.4., i tada program može označiti da na slici nema lica, dok će validacijska slika reći da je model zaboravio prepoznati sliku.

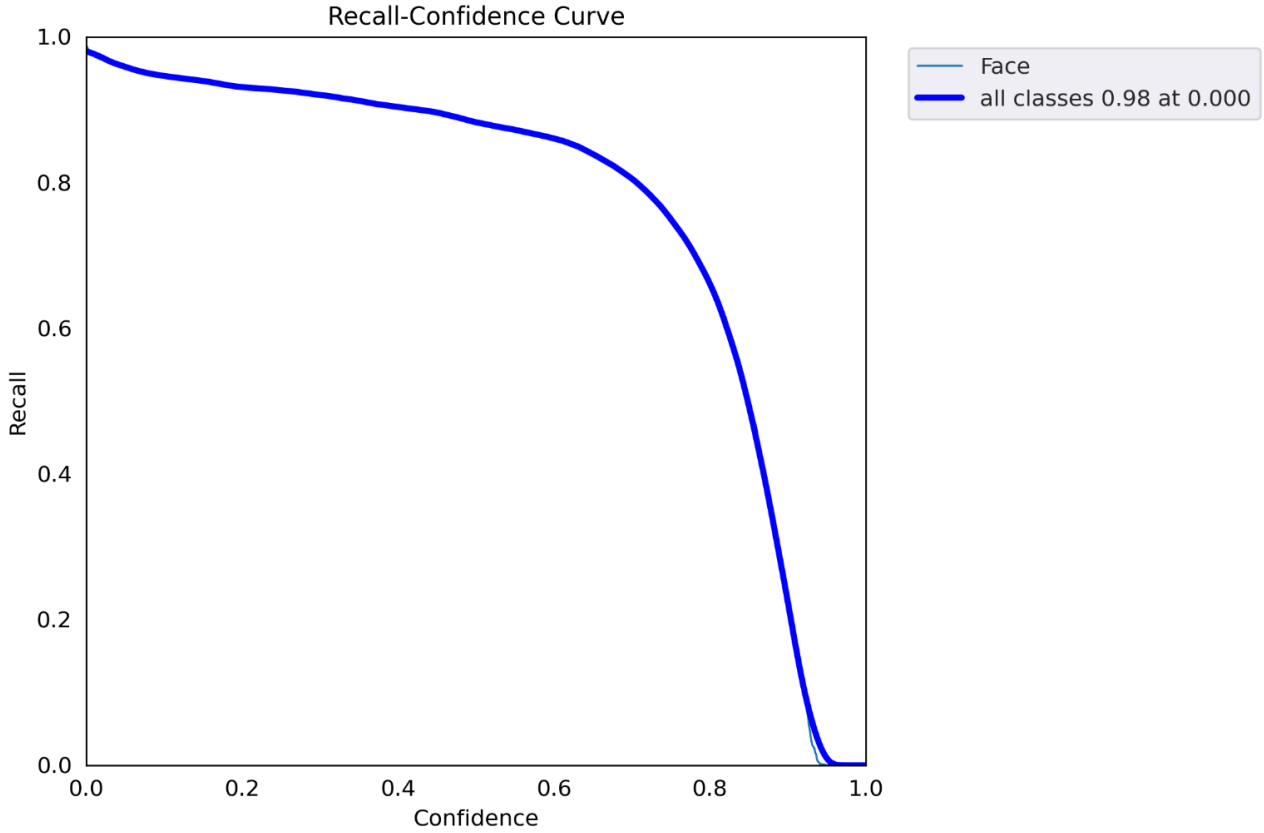
Slika 3.4. Pogreška pri označavanju slike

3.1 Analiza rezultantnih metrika

Sagledavajući sve navedene greške, mogu se početi analizirati dobiveni podaci o preciznosti ovoga modela. Prvi stupac predstavlja regresijske gubitke graničnog okvira (srednja kvadratna pogreška) tj. koliko dobro ovaj model postavi kvadar u kojemu se lice nalazi, te su u donjem redu vrijednosti dobivene za validacijske slike, dok su u gornjem redu vrijednosti dobivene za slike na kojima je model trenirao. Sljedeći stupac govori o pogrešci pretpostavke da se objekt nalazi na određenom području, tj. govori o tome koliko je ovaj model sposoban zaključiti da se na određenom dijelu slike nalazi lice. Treći stupac bi se koristio u slučaju kada bi model prepoznavao više od jednog objekta, te bi pokazivao pogrešku u prepoznavanju klase objekata, ali je u ovom slučaju pogreška jednaka nuli iz razloga što je ovaj model treniran da prepoznaje samo jednu klasu objekata. Graf metrics/precision ispituje koliki postotak predviđenih pretpostavki modela je točan, dok graf metrics/recall govori koliko dobro ovaj model na slici pronalazi sva lica na slici. Grafovi metrics/mAP prikazuju kvalitetu dobivenog modela, a mAP je metrika s kojom se ovaj model može uspoređivati s drugim modelima koji su bazirani na drugačijim objektno baziranim modelima poput R_CNN, Mask R-CNN…

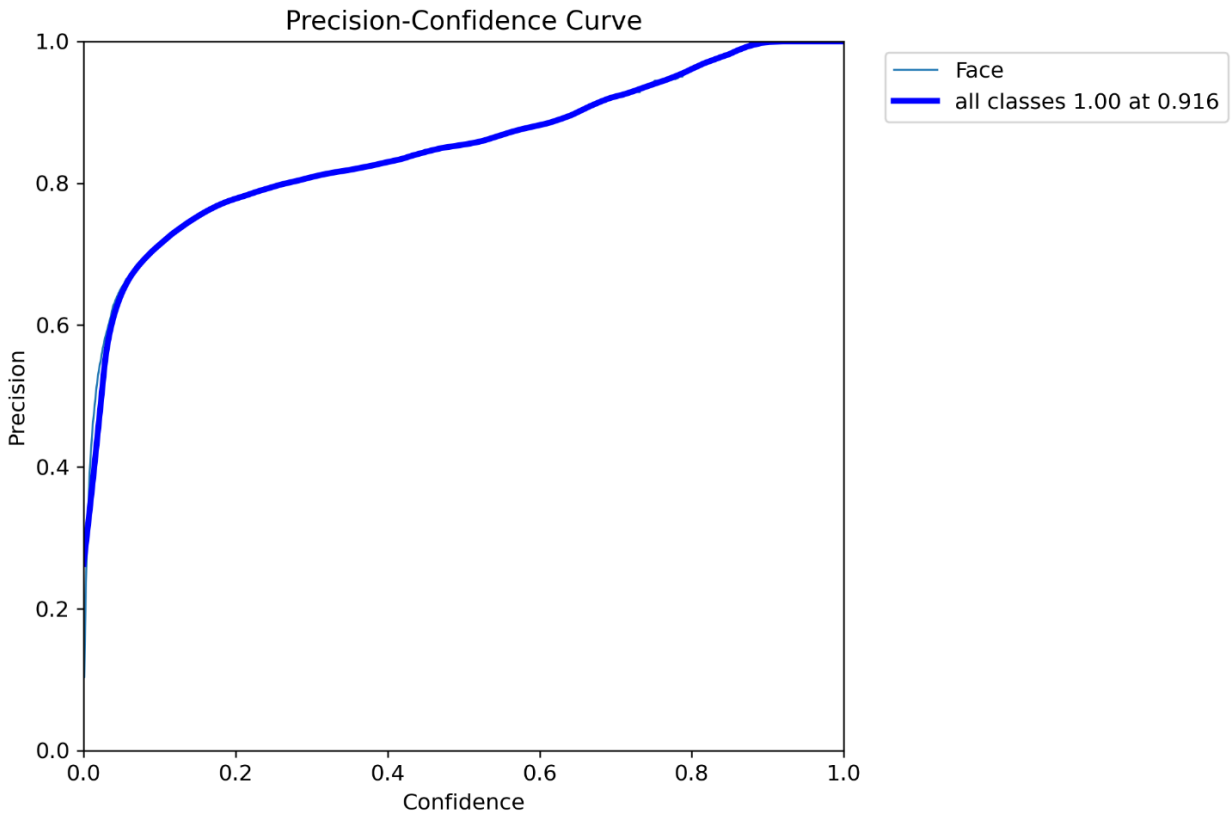


Slika 3.5. Dobiveni grafovi



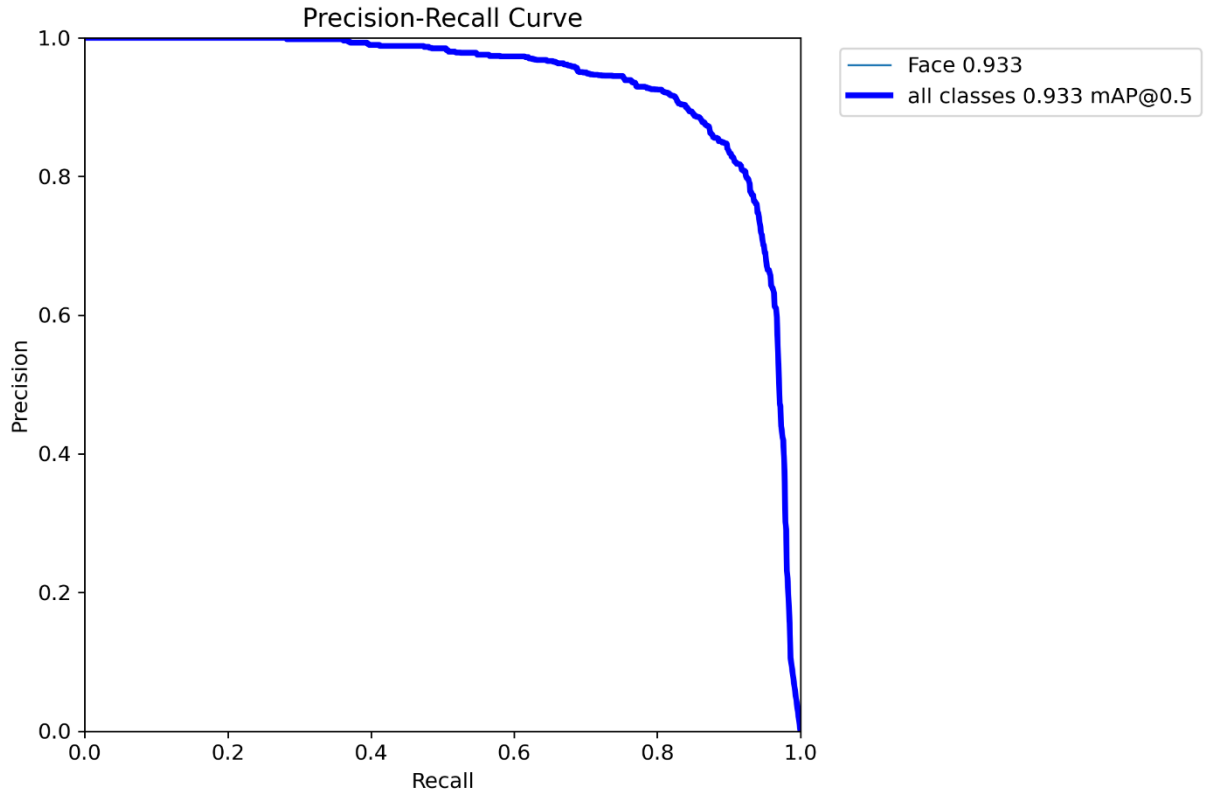
Slika 3.6. Recall/Confidence graf

Dobiveni graf sa slike 3.6. prikazuje ovisnost metrike recall o pouzdanju (eng. confidence) ovoga modela. Može se primijetiti da s većim pouzdanjem modela pada recall metrika. Dolazi se do zaključka da ako se postavi da model ima veću pouzdanost, to će ukupan broj prepoznatih lica pasti. Ako bi se postavilo da je pouzdanost jednaka 1.0, onda model skoro nikada ne bi prepoznao lice budući da nikada ne bi bio dovoljno siguran da je to lice, a također vrijedi i obrnuto, ako postavimo da je pouzdanost modela jednaka nuli, onda će model sve prepoznati kao lice te iz toga slijedi da nije propustio niti jedno lice. (Sjecište sa y osi)



Slika 3.7. Precision/Confidence graf

Dobiveni graf sa slike 3.7. prikazuje ovisnost metrike preciznosti o metrici pouzdanosti. Budući da su preciznost i pouzdanost metrike prema svojoj teoriji slični, ponašanje grafa će biti slično. Porastom pouzdanosti raste i preciznost grafa zbog činjenice da što je model sigurniji da je objekt lice, to je veća vjerojatnost da je i to točna pretpostavka (preciznost raste). U ekstremnim slučajevima kada se postavi da je pouzdanost jednaka 1.0, tada je i preciznost jednaka 1.0 odnosno 100%, a kod recall/precision grafa će detaljnije biti prezentirana problematika ovoga slučaja. U slučaju kada je pouzdanost jednaka nuli, tada je preciznost skoro jednaka nuli iz razloga što model svaki detektirani objekt na slici označava kao lice te preciznost drastično pada. U slučaju da nema lica na slici preciznost je jednaka nuli.

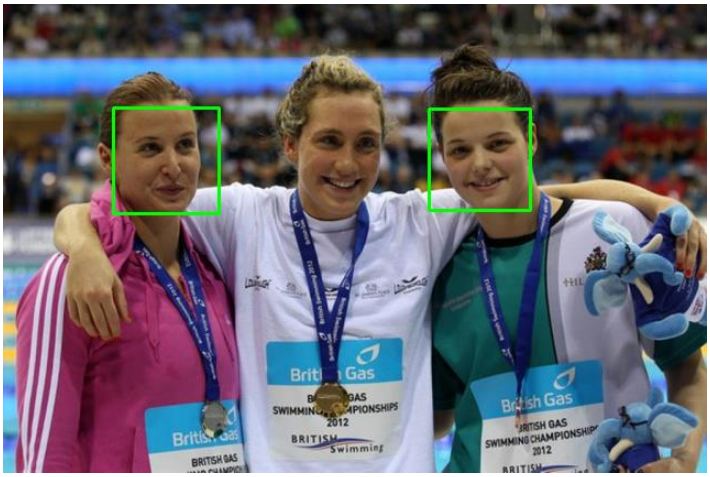


Slika 3.8. Precision/Recall graf

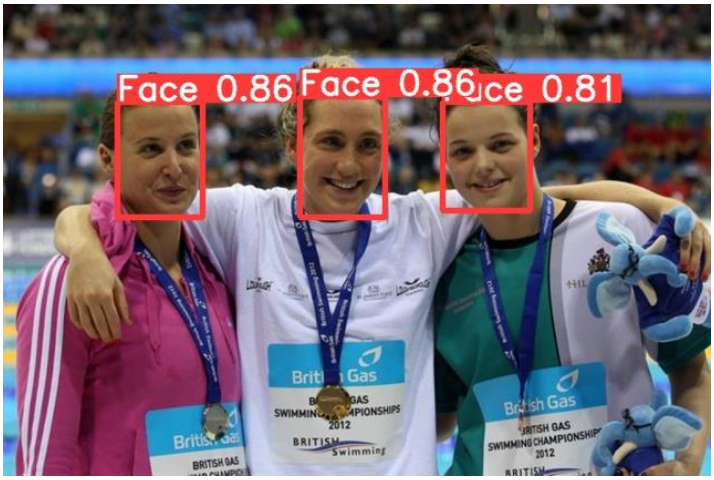
Na slici 3.8. je prikazan graf ovisnosti preciznosti o metrici recall. Evidentno je da rastom recall metrike preciznost opada. Takvo ponašanje može se objasniti nastojanjem da model prepozna sva lica na slici, za što je potrebno smanjiti pouzdanost, no poznato je da smanjenjem pouzdanosti se i preciznost smanjuje. U slučaju kada je metrika recall jednaka 1.0, tada je preciznosti jednaka 0 jer se u tom slučaju detektiraju sva lica na slici, ali uz to postoji puno FP jer je smanjena pouzdanost modela. U drugom slučaju kada je preciznost jednaka 1.0, vrijednosti metrike recall jednaka je 0 jer je povećana pouzdanost do te granice da skoro ništa na slici model neće označiti kao lice što naposljetku znači da neće biti FP, te prema formuli znači da je preciznost jednaka 1.0.

3.2 Usporedba rezultata prepoznavanja sa determinističim algoritmom i algoritmom temeljenim na umjetnoj inteligenciji

Na Slici 3.9. je prikazan rezultat prepoznavanja lica dobiven uz pomoć modela koji nije temeljen na umjetnoj inteligenciji, dok je na slici 3.10. prikazan rezultat dobiven uz pomoć modela koji je temeljen na umjetnoj inteligenciji. Kao prva slika uzeta je ona s natjecanja na kojoj su jasno prepoznatljiva lica, kako bi se testirale mogućnosti modela. Prije svega, jasno je vidljivo da je model baziran na umjetnoj inteligenciji ostvario bolje rezultate budući da je uspješno prepoznao tri od tri lica koja su prikazana na slici, dok model koji nije baziran na umjetnoj inteligenciji uspio prepoznati dva od tri lica. Važno je uočiti razliku označavanja modela. Model koji je baziran na umjetnoj inteligenciji preciznije označava lice, prikazuje koju je klasu objekta prepoznao te postotak sigurnosti da je prepoznao lice. Model koji nije baziran na umjetnoj inteligenciji je lošije označio kvadratom lice tj. obuhvatio je i prostor koji ne predstavlja lice.



Slika 3.9. Slika detektirana korištenjem modela koji nije temeljen na umjetnoj inteligenciji



Slika 3.10. Slika detektirana korištenjem modela temeljenog na umjetnoj inteligenciji

Analizirajući sada drugu sliku na kojoj je teže prepoznati lica, može se vidjeti da se razlika u kvaliteti prepoznavanja lica drastično povećala. Na Slici 3.11. je prikazan rezultat prepoznavanja lica koji nije temeljen na umjetnoj inteligenciji, dok na slici 3.12. je prikazan rezultat prepoznavanja lica koji je temeljen na umjetnoj inteligenciji. Model koji nije baziran na umjetnoj inteligenciji nije uspio prepoznati lice na slici jer osobe ne gledaju direktno u kameru te imaju maske na licu, dok model koji je bio temeljen na umjetnoj inteligenciji je uspio prepoznati sva lica koja su se pojavila na slici, uz malo manji postotak sigurnosti nego na prošloj slici.

Slika 3.11. Slika detektirana korištenjem modela koji nije temeljen na umjetnoj inteligenciji

Slika 3.12. Slika detektirana korištenjem modela temeljenog na umjetnoj inteligenciji

4. ZAKLJUČAK

Ovisno o izradi modela za prepoznavanje objekta pri izvedbi se nude dvije opcije. Prva je opcija da model ne bude temeljen na umjetnoj inteligenciji, dok je druga opcija da model bude temeljen na umjetnoj inteligenciji.

Tijekom ovog rada opisani su materijali prema kojima su izrađeni modeli, te analizirani dobiveni rezultati. U prvom slučaju predstavljen je model koji nije bio temeljen na umjetnoj inteligenciji te se moglo uočiti da je kod potreban za izradu takvog modela jednostavniji te zahtijeva manje vremena kako bi se izradio. Također, ne zahtijeva treniranje modela za koje su potrebni određeni resursi poput vremena i računalne snage. No, unatoč svojoj jednostavnoj izvedbi i primjeni, mana ovog pristupa je manjak preciznosti pri prepoznavanju lica. Iz poglavlja 3. primjećuje se kako je model koji nije bio baziran na umjetnoj inteligenciji ostvario lošije rezultate nego model koji je bio baziran na umjetnoj inteligencije.

U drugom slučaju kreirao se model koji je bio temeljen na umjetnoj inteligenciji te se izabrao yoloV5s model računalnog vida zbog svoje pristupačnije i jednostavnije izrade modela za detekciju lica. Kod ovog modela je puno kompleksniji od koda za model koji nije temeljen na umjetnoj inteligenciji te češće dolazi do situacije kada kod ne radi. Treniranje je također bitna stavka ovog modela te zahtijeva više uloženih resursa i vremena od modela koji nije temeljen na umjetnoj inteligenciji. Prednost ovog modela je znatno veća preciznost od prethodnog modela što je jasno vidljivo u 3. poglavlju gdje je model baziran na umjetnoj inteligenciji ostvario puno bolje rezultate od modela koji nije bio temeljen na umjetnoj inteligenciji.

Uzimajući u obzir sve prednosti i mane ovih dvaju modela, zaključujem da je u većini slučajeva bolje napraviti model koji je temeljen na umjetnoj inteligenciji, jer iako takav tip modela zahtijeva više uloženog vremena i resursa, vjerujem da to nije dovoljan argument za korištenje model koji nije baziran na umjetnoj inteligenciji kod kojeg preciznost i efikasnost prepoznavanja lica znatno opadaju.

Neki od načina na koji su se mogli unaprijediti dobiveni rezultati modela temeljenog na umjetnoj inteligenciji jesu povećanje vremena treniranja, te odabir boljeg izvora slika na kojima je model mogao trenirati i validirati.

5. LITERATURA

[1] Kathuria A.: “ How to train YOLO v5 on a Custom dataset“, s Interneta,

https://blog.paperspace.com/train-yolov5-custom-data/, 2021

[2] Arie L.: „The practical guide for Object Detection with YOLOv5 algorithm“, s Interneta,

https://towardsdatascience.com/the-practical-guide-for-object-detection-with-yolov5-algorithm-74c04aac4843, 14.03.2022

[3] Solawetz J.: „What is YOLOv5? A Guide for Beginners.“, s Interneta,

https://blog.roboflow.com/yolov5-improvements-and-evaluation/, 29.06.2020

[4] Cochard D.: „mAP : Evaluation metric for object detection models“, s Interneta,

https://medium.com/axinc-ai/map-evaluation-metric-of-object-detection-model-dd20e2dc2472, 09.06.2021

[5] Solawetz J.: „How to Train A Custom Object Detection Model with YOLO v5“, s Interneta, https://towardsdatascience.com/how-to-train-a-custom-object-detection-model-with-yolo-v5-917e9ce13208, 15.06.2020

[6] „Umjetna inteligencija“, s Interneta, https://www.enciklopedija.hr/natuknica.aspx?ID=63150 , 03.08.2023

[7] Basu K., Sinha R., Ong A., Basu T.: “ Artificial Intelligence: How is It Changing Medical Sciences and Its Future?“,s Interneta, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7640807/, 05.2020

6. POPIS SLIKA

Slika 2.1. Prikaz usporedbe veličine i brzine treniranja različitih YOLOv5 modela [3]

Slika 2.2. Primjer oznake slike

Slika 2.3. Primjer oznake slike

Slika 2.4. Primjer označavanje slike iz datoteke Dataset

Slika 2.5. Primjer označene mreže slika

Slika 2.6. Primjer slike iz kategorije: Raid

Slika 2.7. Primjer slike iz kategorije: Angler_peoplefishing

Slika 2.8. Primjer slike iz kategorije: Sports_Coach_Trainer

Slika 2.9. Formula i grafički prikaz izračuna IOU parametra [4]

Slika 2.10. Primjer slike sa parametrom pouzdanosti

Slika 2.11. Prikaz izračuna AP parametra pomoću grafa Precision x Recall curve [4]

Slika 3.1. Primjer slika na kojima je model trenirao

Slika 3.2. Primjer validacijskih slika

Slika 3.3. Rezultat prepoznavanja validacijskih slika

Slika 3.4. Pogreška pri označavanju slike

Slika 3.5. Dobiveni grafovi

Slika 3.6. Recall/Confidence graf

Slika 3.7. Precision/Confidence graf

Slika 3.8. Precision/Recall graf

Slika 3.9. Slika detektirana korištenjem modela koji nije temeljen na umjetnoj inteligenciji

Slika 3.10. Slika detektirana korištenjem modela temeljenog na umjetnoj inteligenciji

Slika 3.11. Slika detektirana korištenjem modela koji nije temeljen na umjetnoj inteligenciji

Slika 3.12. Slika detektirana korištenjem modela temeljenog na umjetnoj inteligenciji

7. SAŽETAK

U ovo radu izvršena je usporedba modela za prepoznavanje lica, i to prvi model napravljen pomoću klasičnog programiranja dok je drugi model bio temeljen na umjetnoj inteligenciji te arhitekturi YOLOv5. Nakon provedene usporedbe može se zaključiti da je model koji je bio temeljen na umjetnoj inteligenciji postigao bolje rezultate pri prepoznavanju lica, no također je zahtijevao više uloženih resursa i vremena. Model kreiran klasičnim programiranjem je ostvario lošije rezultate pri prepoznavanju lica, no pri njegovoj izvedbi je utrošeno znatno manje vremena i resursa. Zaključujem da je u većini slučajeva pogodnije koristiti model temeljen na umjetnoj inteligenciji jer uloženo vrijeme i potrebni resursi opravdavaju bolje rezultate kod prepoznavanja lica. U ekstremnim slučajevima model napravljen klasičnim programiranjem može biti koristan, npr. u situacijama kada smo vremenski ograničeni ili nemamo dovoljno resursa kako bi istrenirali model.

Ključne riječi: umjetna inteligencija, prepoznavanje lica, YOLOv5, programiranje

ABSTRACT

In this paper we compared two models for face recognition. First model was built using "normal" programing, while the second model was built using artificial inteligence based on the YOLOv5 architecture. After running the comparison, I've came to conclusion that the second model that was based on artificial inteligence performed better while recognising faces but it also consumed more time and processing power to realise. Model that was built using „normal“ programing performed worse but needed less time and processing power to create. In conclusion in most cases it is better to use the model based on artificial inteligence because invested time and computing power are justified with better results in face recognition. In extreme cases, model that was based on "normal" programing can be usefull, for example when we don't have enough time or we don't have enough processing power.

Key words: artificial inteligence, face recognition, YOLOv5, programming