Fakultet Elektrotehnike, Računarstva i Informacijskih tehnologija Osijek

PRIMJENJENO STROJNO UČENJE

BROJANJE PRSTIJU RUKE KORISTEĆI STROJNO UČENJE

Mentor: Ratko Grbić

Izradili:

Mihael Španović

Sven Erić

SADRŽAJ

SADRŽAJ 1

UVOD 2

2. IZRADA CNN MODELA 3

2.1 Odabir dataset-a za izradu modela 3

2.2 Učitavanje podataka i konvolucijski slojevi 4

2.3 Učenje modela 6

2.4 Izrada programa za testiranje modela 9

2.5 Testiranje u stvarnim uvjetima 10

3. IZRADA MEDIAPIPE PROGRAMA 12

3.1 Mediapipe 12

3.2 Izrada Mediapipe programa 13

3.3 Testiranje u stvarnim uvjetima 14

4.USPOREDBA DVA PROGRAMA 15

5. ZAKLJUČAK 16

6.REFERENCE 17

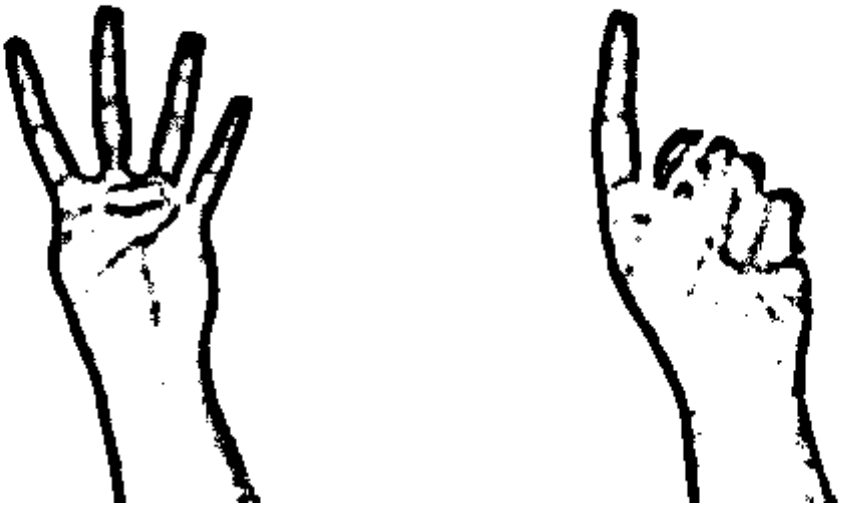
UVOD

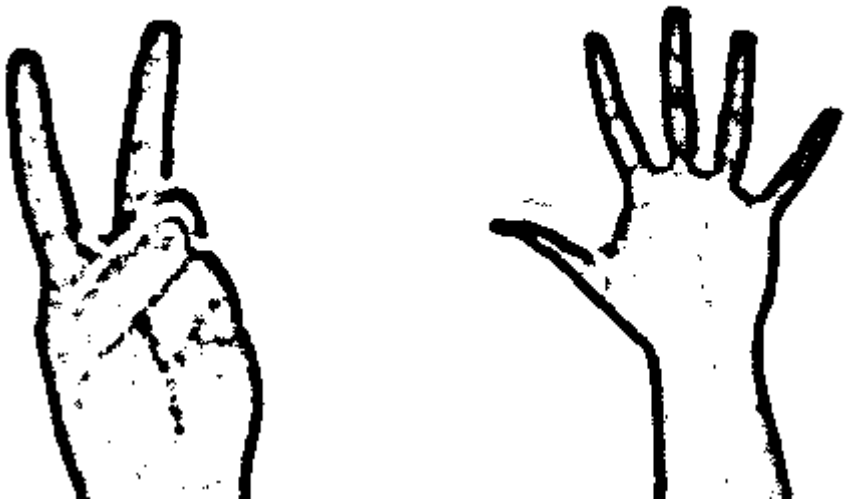
Kao uvjet uspješnog prolaza kolegija Primijenjeno strojno učenje bila je izrada završnog projekta koji bi koristio razne tehnologije koje su naučene kao dio izvođenja nastave. Neke od tehnologija su: rad sa Python programskim jezikom, korištenje machine learning library-a za izradu preciznih modela, primjena izrađenog modela u stvarnim/simuliranim okruženjima. Inicijalni cilj našeg projekta je bila izrada, učenje i testiranje CNN-a (eng. Convolutional Neural Network) za svrhu brojanja prstiju na ruci osobe. Kroz ostatak dokumentacije opisati ćemo postupke izrade, naša razmišljanja i zapažanja te pokazati rezultate završenog modela. Također kao dodatni cilj htjeli smo napraviti usporedbu između CNN modela i već treniranog modela iz Python biblioteke MediaPipe.

2. IZRADA CNN MODELA

2.1 Odabir dataset-a za izradu modela

Prije izrade samog modela potrebno je prikupiti podatke koje ćemo kasnije koristiti za treniranje neuronske mreže. Naš cilj je bio pronaći dataset slika ljudske šake sa različitim brojem prstiju „u zraku“ te sa realističnim osvjetljenjem i pozadinom. Zbog manjka dostupnih dataset-ova odlučili smo se koristiti model bez pozadine te kasnije prilagoditi naš program kako bi input odgovarao treniranom.





Slika 1. Primjer slika za treniranje modela

Direktorij se sastoji od dvije glave mape. Mape „train“ i „test“ podijeljene su na pod mape sa nazivima „ONE, TWO“ itd. Unutar mapa se nalaze specifične slike koje odgovaraju imenu mape.

2.2 Učitavanje podataka i konvolucijski slojevi

Tijekom učitavanja dataset-a u program potrebno je postaviti zajedničku veličinu koju će pratiti svaka slika, izabrati color_mode, odrediti batch size i za kraj class_mode.

```
train_gen = train_datagen.flow_from_directory(
        filepath_train,
        target_size=(300, 300),
        color_mode='grayscale',
        batch_size=nbatch,
        classes=['NONE','ONE','TWO','THREE','FOUR','FIVE'],
        class_mode='categorical'
    )
```

Slika 2. Učitavanje podataka u programu

Od navedenih parametara mislimo da je najbitnije objasniti class_mode. Class mode ima više opcije od kojih su neke: categorical, binary, None.

Categorical = 2D encoded podaci koji su ekskluzivni, npr : Pas ne može biti mačka, čovjek nije pas.

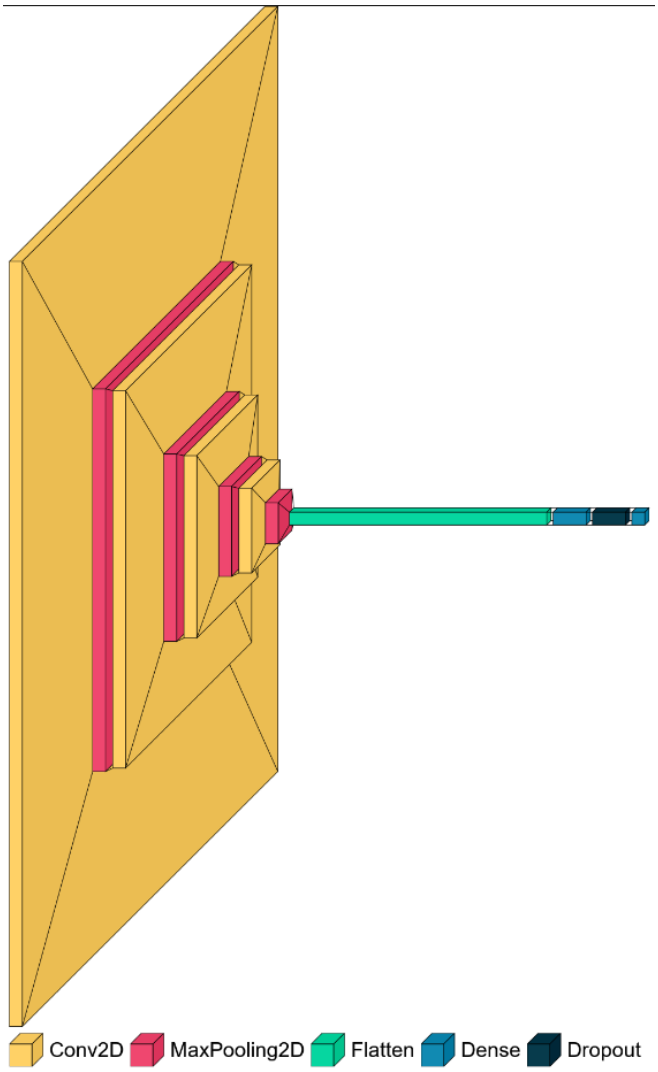
Binary = 1D podatci zapisani kao 1 broj.

Class_mode nam služi za određivanje vrste povratnih nizova koji sadrže jednu od naših klasa.

Nakon učitavanja podataka potrebno je postaviti slojeve konvolucijske mreže.

```
model = Sequential()
model.add(Conv2D(32, (3,3), activation='relu', input_shape=(300,300,1)))
model.add(MaxPooling2D((2,2)))
model.add(Conv2D(64, (3,3), activation='relu'))
model.add(MaxPooling2D((2,2)))
model.add(Conv2D(128, (3,3), activation='relu'))
model.add(MaxPooling2D((2,2)))
model.add(Conv2D(128, (3,3), activation='relu'))
model.add(MaxPooling2D((2,2)))
model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(6, activation='softmax'))
```

Slika 3. Slojevi konvolucijske mreže



Slika 4. 3D prikaz slojeva mreže

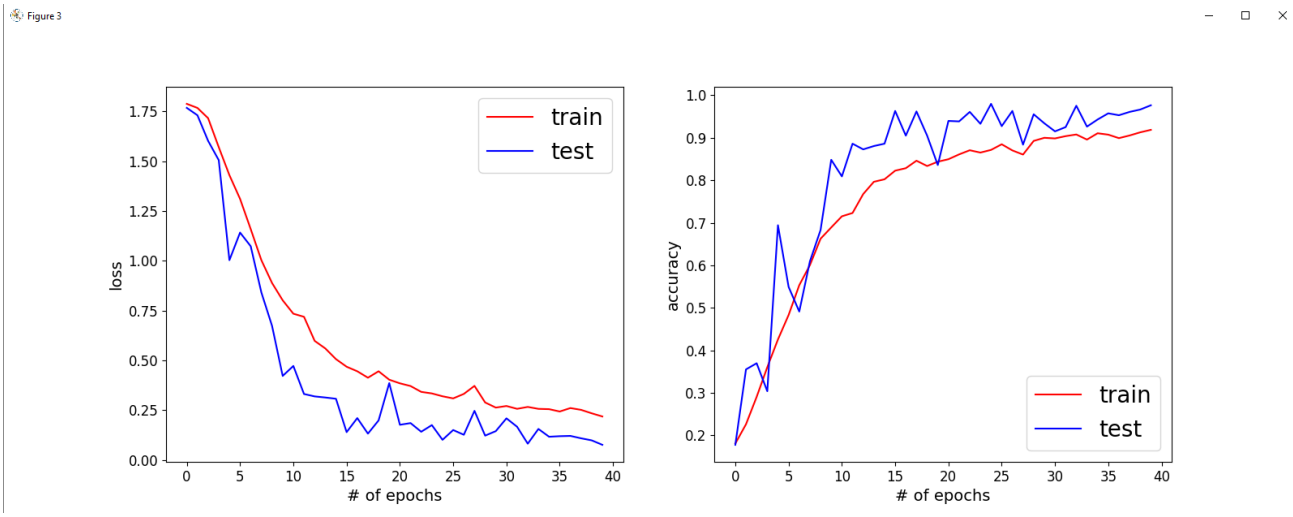
Konvolucijski model se sastoji od 4 konvolucijska sloja od kojih svaki sadrži 3x3 kernele, RELU aktivaciju i inkremente od 32 filtera. Između svakog konvolucijskog sloja nalazi se MaxPooling koji smanji dimenzije modela. Nakon flatten funkcije model šaljemo na Dense sloj te softmax na kraju gdje se definira svaka od klasa.

2.3 Učenje modela

Učenje modela je bilo izvršavano uz pomoć tensorflow i NVIDIA cuDNN biblioteke koja omogućava korištenje grafičke kartice za ubrzano učenje modela.

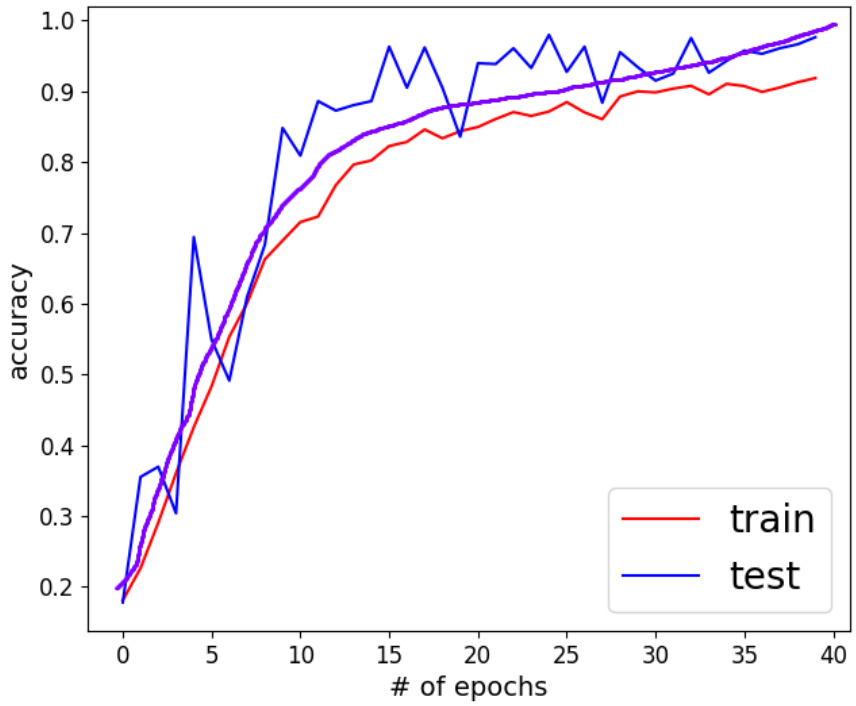
Batch size iznosi 64, a učenje se odvijalo kroz 40 epoha od kojih je svaka imala 80 koraka.

Učenje je trajalo oko 35 minuta, a na kraju dobivena točnost iznosi 0.9672 sa gubitkom od 0.0778.



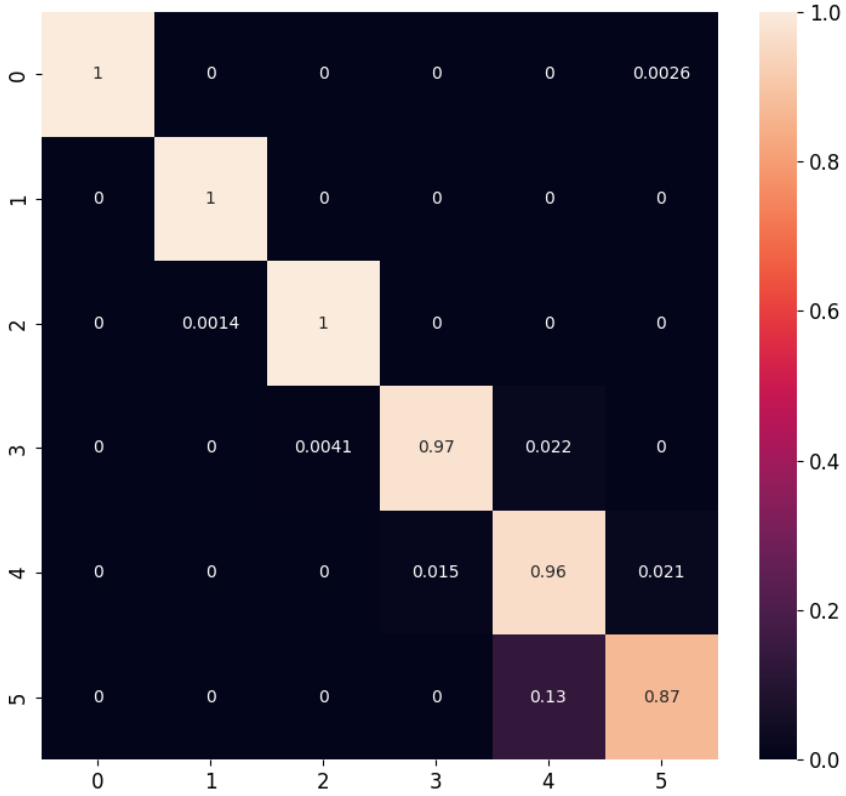
Slika 5.Prikaz točnosti i value lossa pomoću grafa

Na prikazu pomoću grafa jasno možemo vidjeti da u modelu postoji problem overfitting-a, ali nakon ponovnih pokušaja učenja nismo mogli premašiti rezultat dobiveni tijekom ovog pokušaja.



Slika 7. Graf sa dodanom „idealnom“ krivuljom

U nekim boljim uvjetima naš graf bi pratio ljubičastu liniju te bi samo treniranje modela trebalo manje vremena.



Slika 8. Konfuzijska matrica

Konfuzijska matrica pokazuje rezultate koji će se kasnije vidjeti tijekom testiranja u stvarnom okruženju. Najveća greška se događa kod brojeva 4 i 5 gdje model u određenim uvjetima teško raspoznaje palac što dovodi do zabune u procjeni broja.

2.4 Izrada programa za testiranje modela

Prvi zadatak je bila izrada maske koja odgovara našem datasetu.

```
    img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    img = cv2.GaussianBlur(img, (7,7), 3)
    img = cv2.adaptiveThreshold(img, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY_INV, 11, 2)
    ret, new = cv2.threshold(img, 25, 255, cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)
```

Slika 9. Funkcija za izradu maske

Boja je postavljena na grayscale pomoću funkcije cvtColor, funkcija GaussianBlur služi sa postavljanje blura preko regije interesa. Za našu upotrebu vrijednost 7,7 je dosta dobra, neko pravilo je da se ne prelazi vrijednost veličine slike i da je broj neparan. AdaptiveTreshhold služi za izvlačenje ključnih crta objekta. Korištenjem parametra ADAPTIVE_THRESH_GAUSSIAN_C postavlja maksimalnu vrijednost piksela sa obzirom na susjedni piksel, a za usporedbu se koristi vrijednost blura u funkciji iznad. Treshhold radi slični adaptiveThresholdu, ali sa fiksnim vrijednostima. Jednostavno opisano ukoliko je vrijednost piksela veća, a zadane vrijednosti promijeni vrijednost na 255 u ovom slučaju.

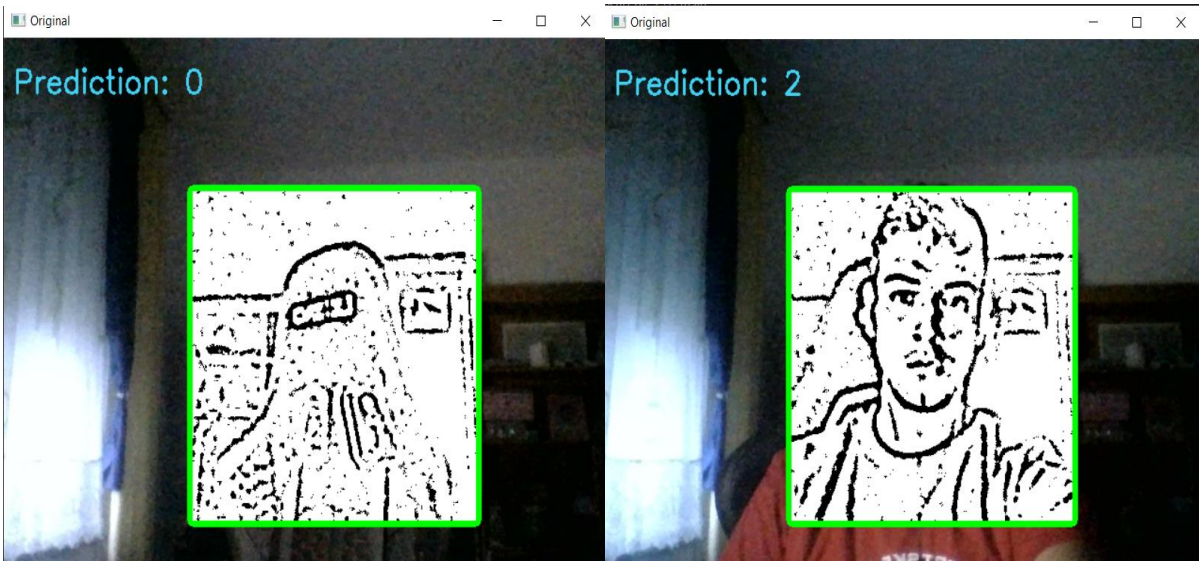
Za kraj je potrebno samo učitati model, postaviti prozor za web kameru i napraviti predikciju.

```
img = np.resize(img, (4,300, 300,1))
pred = classes[np.argmax(model.predict(img)[0])]
cv2.putText(window, 'Prediction: %s' % (pred), (fx,fy), font, 1.0, (245,210,65), 2, 1)
```

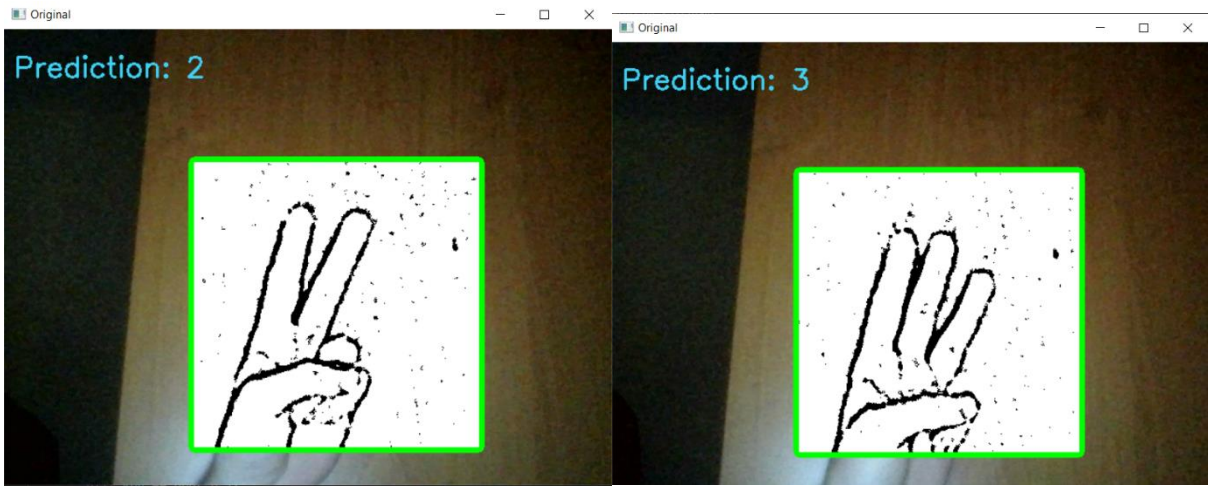
Slika 10. Primjer koda za prediction

2.5 Testiranje u stvarnim uvjetima

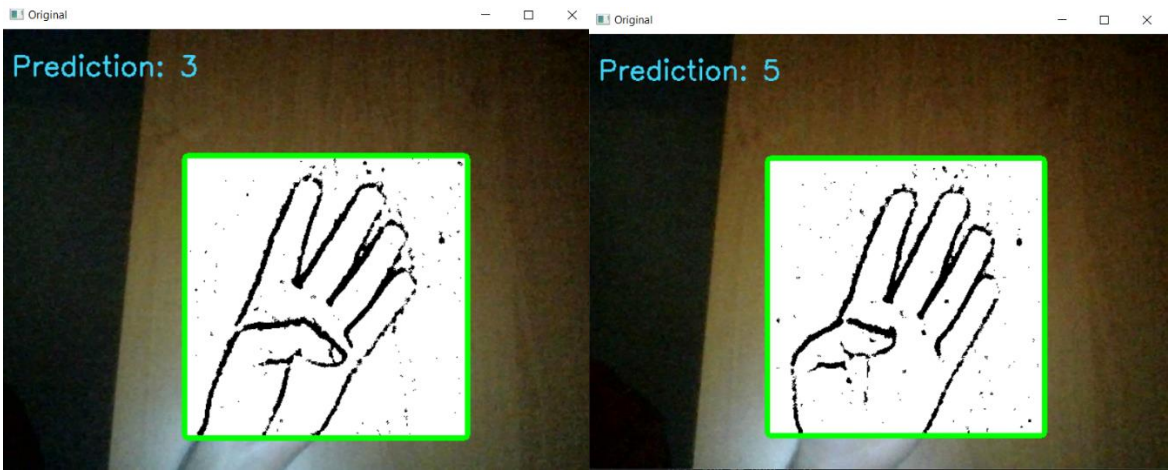
Dobivena maska je iznenađujuće dobra u raspoznavanju objekata čak i lošim uvjetima osvijetljenosti. Zbog toga je potrebno prepoznavanje raditi na čistoj površini. Dobiveni rezultati su relativno točni, izvršavaju se u stvarnom vremenu bez uzimanja slike. Problemi nastaju kada postoje različite crte drugih objekata u kadru i kada se ruka nalazi pod čudnim kutom. Također može doći do zabune ukoliko se palac nalazi na dlanu, a dlan je okrenut prema kameri.



Slika 11,12. Primjer prepoznavanja objekata



Slika 13,14,15. Primjer točnog prepoznavanja



Slika 16,17. Primjer lošeg prepoznavanja

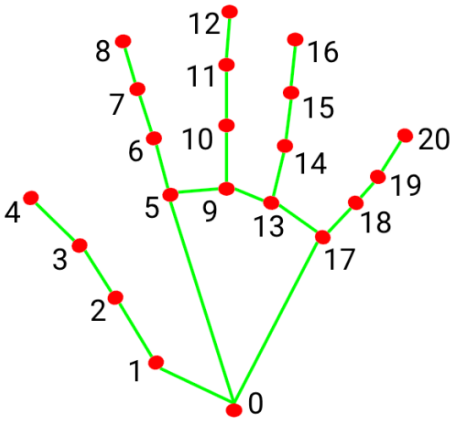
Kao što je vidljivo na primjerima lošeg prepoznavanja greška se najčešće događa zbog krivog prepoznavanja palaca kao još jednog prsta i nedovršenog outline-a zbog loših uvjeta za prepoznavanje.

3. IZRADA MEDIAPIPE PROGRAMA

Izradom CNN modela zanimalo nas je koliko bi to rješenje bilo lošije/bolje od moguće izvedbe programa bez treniranja modela već korištenjem pre-trained modela i kontura ruke.

3.1 Mediapipe

Mediapipe je open-source projekt sponzoriran od strane Google-a i namijenjen je jednostavnom i brzom integriranju AI-a u razne projekte. Neki od njihovim modela su: Hair segmentation, 3D object recognition, Face detection i na kraju hand tracking. Media pipe radi na način da prepoznaje ključne točke svake ruke i uz pomoć matematičkih kalkulacija određuje njen položaj.



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Slika 18. Ključne točke ljudske ruke

Za korištenje mp_hand_gesture modela potrebno je postaviti indeks za sigurnost („confidence“) i maksimalni broj ruku koje će program pratiti.

3.2 Izrada Mediapipe programa

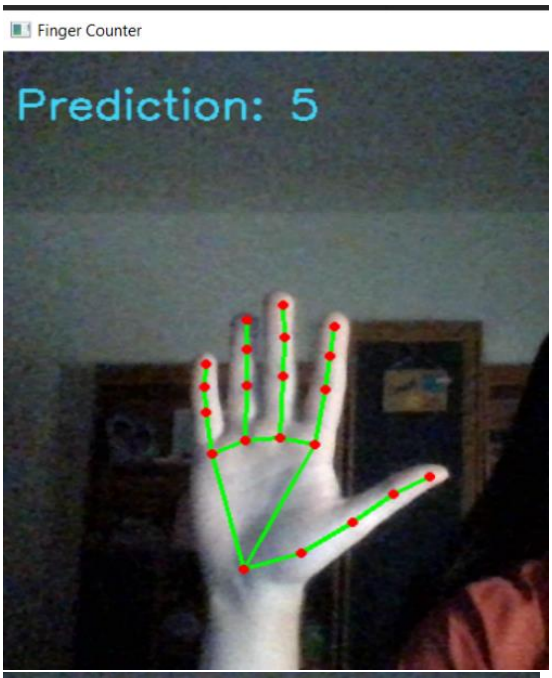
Mediapipe program se sastoji od dva glavna dijela. Gornja for petlja služi za prepoznavanje ruke i ocrtavanje ključnih točaka. Drugi dio programa koristi niz koji se sastoji od brojeva koji u Mediapipe-u označavaju vrh prsta. Ukoliko prepozna da postoji točka koja označava vrh prsta to znači da je prst u zraku i funkcija append doda 1 za broj prstiju. Slično se događa sa palcem samo što je njegova točka dosta bliže sredini ruke od ostalih prstiju pa je potrebno napraviti zasebni if uvjet.

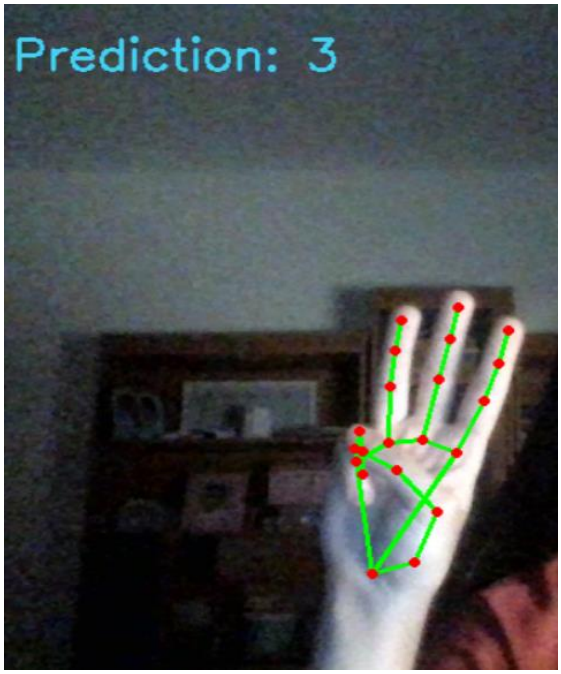
```
lmList = []
if results.multi_hand_landmarks:
    for hand_landmark in results.multi_hand_landmarks:    #Petlja za pracenje ruke
        myHands = results.multi_hand_landmarks[0]
        for id, lm in enumerate(myHands.landmark):
            h, w, c = image.shape
            cx, cy = int(lm.x*w), int(lm.y*h)
            lmList.append([id, cx, cy])

        mp_draw.draw_landmarks(
            image, hand_landmark,
            mp_hand.HAND_CONNECTIONS)

fingers = []
if len(lmList) != 0:
    # Palac
    if lmList[tipIds[0]][1] > lmList[tipIds[0]-1][1]:
        fingers.append(1)  #DA
    else:
        fingers.append(0)  #NE

    # Ostali prsti
    for id in range(1, 5):
        if lmList[tipIds[id]][2] < lmList[tipIds[id]-2][2]:
            fingers.append(1)  #DA
        else:
            fingers.append(0)  #NE

    total = fingers.count(1)
    cv2.putText(image, 'Prediction: %s' % (total), (fx,fy), font, 1.0, (245,210,65), 2, 1)
```

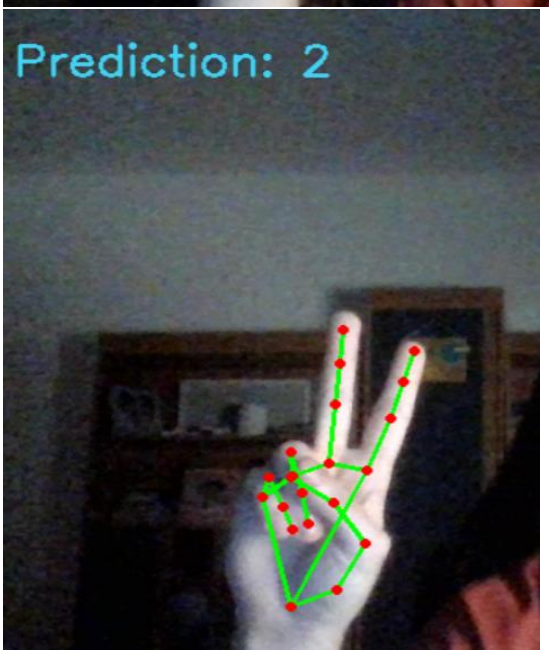
Slika 19. Ključne funkcije MediaPipe programa

Izrada Mediapipe programa je trajala 1/5 vremena originalnog programa. Dok su rezultati puno bolji u stvarnim uvjetima.

3.3 Testiranje u stvarnim uvjetima







Slika 20,21,22. Testiranje MediaPipe programa

Iz testiranja vidljivo je da se program baziran na MediaPipe modelu snalazi bolje u stvarnim uvjetima te postoji i komponenta praćenja ruke tako da nije potrebno određivati regiju interesa(eng. region of interest) za ovaj program.

4.USPOREDBA DVA PROGRAMA

Nažalost razne konfuzijske matrice, grafovi itd. za MediaPipe nisu javno dostupno tako će sve navedeno biti iz iskustva sa radom i testiranjem oba programa.

VLASTITI CNN MODEL

Prednosti:

- „Hands on“ pristup radu gdje završni rezultat ovisi isključivo o vlastitom izboru parametara, dataseta itd.,
- veća sloboda izbora s čime dolazi i dodatno iskustvo rada,
- završni proizvod nije javno dostupan već je osobno vlasništvo,
- pristup raznim matricama, grafovima itd.

Mane:

- točnost jako često ovisi o raznim uvjetima,
- duže vrijeme izrade,
- potraga za dobrim dataset-om može biti izazovna,
- praćenje ruke je zahtjevno za izradu.

MEDIAPIPE

Prednosti:

- lakša i brža izrada programa,
- mogućnosti praćenja ruke,
- veća točnost u stvarnim uvjetima.

Mane:

- manja sloboda tijekom izrade programa,
- manjak grafičkih prikaza rezultata.

5. ZAKLJUČAK

Smatramo kako je projekt bio uspješan, ali naravno par stvari bi moglo biti bolje ili unaprijeđeno za stvarne uvijete. Pokušati ćemo navesti par mogućih rješenja kako bi se mogao unaprijediti originalni projekt.

Korištenjem boljeg dataset-a, sa slikama koje reprezentiraju stvarne uvjete i položaje ruke mogla bi se dobiti veća točnost i bolje prepoznavanje ruke u stvarnim uvjetima. Poboljšanje rezultata također može doći od boljih svjetlosnih uvjeta u prostoriji gdje se program testira.

Jedna od mogućih primjena ovog programa je kontrola računala sa obzirom na pokazani broj, kontrola zvuka, otvaranje programa itd.

Zanimljiv nusprodukt ovog projekta je binarna maska koja bi se uz različite druge modele mogla koristiti za prepoznavanje raznih objekata u prostoriji ili u okruženju recimo robota.

Izrada ovog projekta je bila izrazito edukativna i unatoč svim problemima koji su se znali pojaviti bila je zabavna. Voljeli bih se vratiti ovom projektu za par godina i pokušati napraviti nešto slično sa boljim rezultatima ili drugačijom primjenom.

6.REFERENCE

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.plot_confusion_matrix.html

https://github.com/lordmahyar/keras-visualizer

https://google.github.io/mediapipe/solutions/hands

https://docs.opencv.org/master/

GITHUB REPOZITORIJ: https://github.com/Mihael283/PSU_Finger_counting