SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Klasifikacija rukom gestikuliranih brojeva

Seminarski rad

Raspoznavanje uzoraka i strojno učenje

Manuel Pleš

Osijek 2021

Sadržaj

1. UVOD

U ovom projektnom zadatku vrši se obrada i klasifikacija fotografija rukom gestikuliranih brojeva. Važno je napomenuti da se na rukama nalaze rukavice određene boje te se iste zatim detektiraju i izoliraju koristeći maske u programu. Program je napisan u programskom jeziku Python te Anaconda razvojnom okruženju .Objašnjene su i korištene konvolucijske neuronske mreže (*eng. CNN*) koje spadaju u duboko učenje.

Podatkovni skup korišten za treniranje CNN modela je izrađen prilikom izrade projekta te isti nije moguće pronaći nigdje na internetu. Za izradu podatkovnog skupa napisani su programi koji to uvelike olakšavaju. Podatkovni skup sastoji se od 3000 slika jednako raspoređenih u 12 klasa odnosno 250 slika po klasi. Klase od 0 do 5 predstavljaju brojeve prikazane desnom rukom, a klase od 6 do 11 predstavljaju brojeve prikazane lijevom rukom.

Rezultati testiranja modela prikazani su pomoću matrica zabune i klasifikacijskih izvještaja te su objašnjeni.

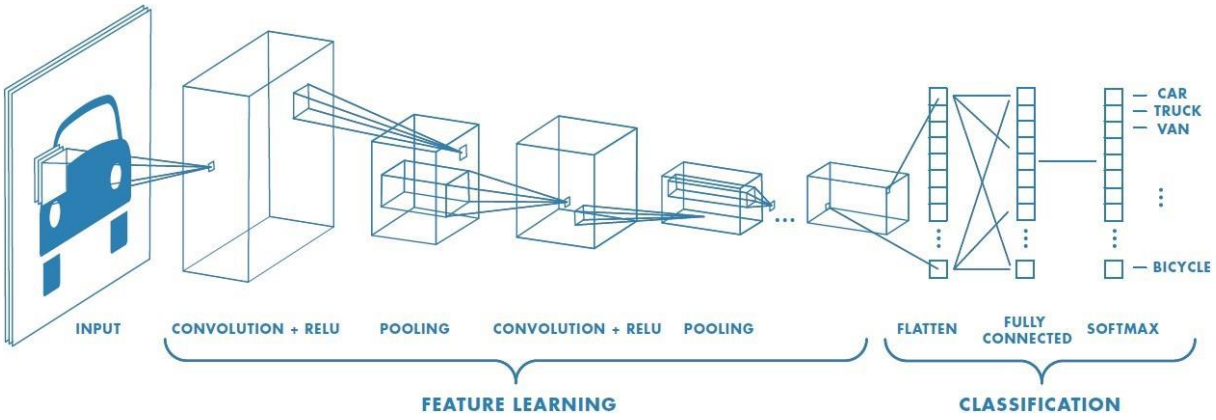
Cilj ovog projektnog zadatka je upoznati se sa konvolucijskim neuronskim mrežama, računalnim vidom i programskim jezikom Python. Također je cilj napraviti vlastiti model konvolucijske neuronske mreže, istrenirati ga i testirati u raznim situacijama.

2. PREGLED PODRUČJA I PROBLEMATIKE

Umjetne neuronske mreže su algoritmi strojnog učenja modelirani po uzoru na biološke neuronske mreže unutar ljudskog mozga. Neuronska mreža predstavlja sustav sastavljen od velikog broja neurona koji su međusobno povezani. Neuroni su strukturirani u slojeve(ulazni sloj, jedan ili više skrivenih i izlazni sloj) i oni služe kao jedinice za obradu podataka koje primaju ulazne vrijednosti, transformiraju ih te ih šalju na izlaz. Način na koji su neuroni međusobno povezani naziva se arhitektura mreže. Arhitektura neuronske mreže koja je od važnosti u ovom projektu je konvolucijska neuronska mreža(*eng. CNN*).

1. Konvolucijska neuronska mreža (CNN)

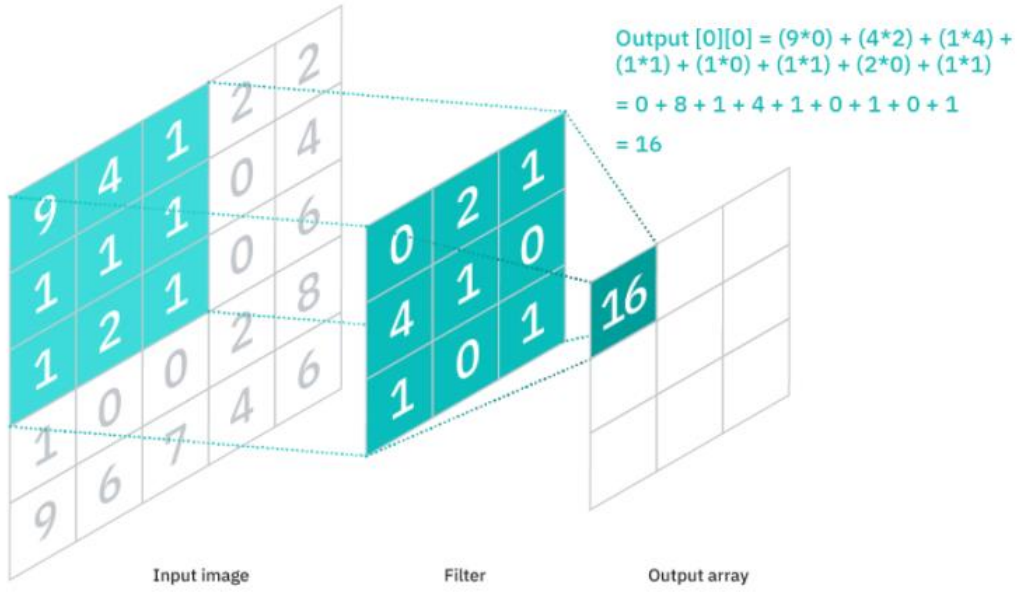
Konvolucijska neuronska mreža trenutno je najkorišteniji oblik dubokih neuronskih mreža u zadacima računalnog vida. Sastoji se od tri glavna sloja: konvolucijski sloj, sloj sažimanja te potpuno povezani sloj.



Slika 1. Konvolucijska neuronska mreža

Konvolucijski sloj je prvi i glavni sloj u kojem se odvija većina računanja. Njemu je potrebno par komponenti, a to su: ulazna vrijednost, filter i mapa značajki. U ovom projektu ulazna vrijednost je slika. Detektor značajki odnosno filter ili kernel prolazi kroz sliku i provjerava jeli prisutna značajka. Ovaj proces se zove konvolucija. Detektor značajki je dvodimenzionalno polje težina koje predstavlja dio slike. Iako mogu biti različitih veličina, filter je inače matrica 3x3. Filter se primjenjuje na dio slike i vrši se množenje ulaznih pixela slike i filtera te se rješenje sprema u izlazno polje. Filter se pomiče za jedan korak i proces se ponavlja sve dok ne

prođe cijelu sliku. Konačni izlaz odnosno izlazno polje dobiveno na taj način naziva se mapa značajki.



Slika 2. Primjenjivanje filtera

Nakon konvolucijskog sloja najčešće slijedi sloj sažimanja. Sloj sažimanja se primjenjuje kako bi se smanjio broj parametara na ulazu. Slično kao i kod konvolucijskog sloja operacija sažimanja primjenjuje filter na cijeli ulaz, ali razlika je u tome što ovaj filter nema težine. Umjesto toga, kernel primjenjuje agregacijsku funkciju(average, median, mean, minimum, maximum...) na vrijednosti unutar promatranog dijela i rezultat dodaje u izlazno polje i tako dok ne prođe cijelu ulaznu vrijednost. Agregacijska funkcija koristi se za transformaciju više vrijednosti u jednu vrijednost koja ih najbolje opisuje. Postoje dvije vrste sažimanja, a to su: maksimalno sažimanje i prosječno sažimanje. Kod maksimalnog sažimanja kada se filter pomiče po ulaznoj vrijednosti, on odabire pixel sa najvećom vrijednosti i njega šalje u izlazno polje. Ovaj pristup je češće korišten. Kod prosječnog sažimanja filter se pomiče po ulaznoj vrijednosti i računa prosječnu vrijednost svih promatranih pixela te nju šalje u izlazno polje. Iako se dosta informacija izgubi u sloju sažimanja ipak su tu brojne prednosti kao što su smanjena kompleksnost, poboljšana učinkovitost i ograničen rizik overfittanja.

Na kraju dolazi potpuno povezani sloj koji obavlja klasifikaciju temeljenu na značajkama dobivenim iz prethodnih slojeva i njihovih filtera. Dok konvolucijski slojevi i slojevi sažimanja

često koriste 'relu' aktivacijsku funkciju, potpuno povezani slojevi najčešće koriste 'softmax' aktivacijsku funkciju za precizniju klasifikaciju ulaza, dajući time vjerojatnost od 0 do 1.

2. Nadzirano učenje

Nadzirano učenje je pristup strojnog učenja kojeg definira označeni podatkovni skup. Za svaki ulazni podatak X već se unaprijed zna točan izlazni podatak Y. Takvi podatkovni skupovi su osmišljeni kako bi nadzirali odnosno naučili algoritme da precizno klasificiraju podatke. Algoritam „uči“ iz podatkovnog skupa na način da iterativno radi predviđanja na danom skupu i prilagođava se točnom izlazu. Korištenjem označenih ulaza i izlaza, model može izračunati svoju točnost i poboljšati se s vremenom. Nadzirano učenje se dijeli na dva tipa problema: klasifikacijske probleme i regresijske probleme.

Klasifikacijski problemi koriste algoritam kako bi točno kategorizirali testne podatke npr. odvajanje jabuka od naranči.

Regresijski problemi koriste algoritam kako bi shvatili vezu između zavisnih i nezavisnih varijabli. Korisni su za stvari kao što je predviđanje stanja burze.

U ovom projektnom zadatku koristit će se klasifikacija jer je problem klasifikacijske prirode. Potrebno je ovisno o ulazu(slika geste ruke) odrediti o kojoj je od 12 mogućih gesti riječ.

3. RJEŠENJE

1. Korištene biblioteke

NumPy - biblioteka koja dodaje podršku za velika, višedimenzionalna polja i matrice zajedno sa velikom kolekcijom matematičkih funkcija za manipulaciju istih.

OpenCV-Python – biblioteka osmišljena da riješi probleme računalnog vida.

Tensorflow – biblioteka otvorenog koda koja omogućava stvaranje neuronskih mreža. Koristi se za klasifikaciju, percepciju, razumijevanje, otkrivanje, predviđanje i stvaranje.

Keras – biblioteka otvorenog koda koja u svojoj pozadini može koristiti TensorFlow. CNTK, Theano ili MXNet. U ovom projektu se koristi Tensorflow.

Scikit-image – biblioteka otvorenog koda osmišljena za obradu slika.

Scikit-learn – biblioteka korištena za strojno učenje, u ovom radu je korištena za raspodjelu podatkovnog skupa na skup za treniranje i skup za testiranje

Matplotlib – biblioteka za vizualizaciju podataka bazirana na NumPy poljima

PIL – Python Imaging Library, biblioteka otvorenog koda koja dodaje funkcionalnost otvaranja, manipuliranja i spremanja slika u različitim formatima.

Shutil – biblioteka korištena za kopiranje i premještanje slika

2. Izrada podatkovnog skupa

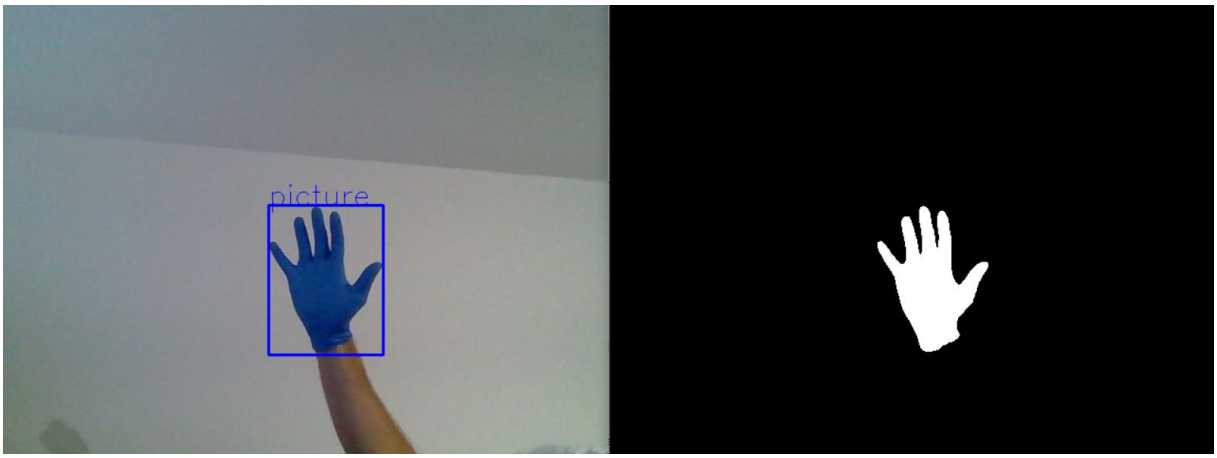
Prvo je napravljen folder *freshDataset_train* koji u sebi sadržava dvanaest foldera s nazivima zero,one,three,four…eleven. Prvih šest foldera(zero-five) predstavlja prvih šest klasa gdje svaki naziv foldera opisuje broj gestikuliran desnom rukom, a drugih šest foldera(six-eleven) opisuje isto to samo za lijevu ruku.

Zatim je napisana *tekePicture.ipynb* skripta koja kad se pokrene korisniku pokreće kameru te otvara dva prozora. U jednom prozoru je prikazana normalna slika, a u drugom prozoru je prikazana binarna slika. U pozadini program čita svaki frame, zatim iz frame-a propušta samo boju iz određenog raspona(masku), traži konture iz propuštenog i računa površinu konture. Ukoliko je površina veća od određene vrijednosti(u ovom slučaju 300 kako ne bi označavalo sitne dijelove plave boje) pomoću metode *boundingRect()* se dobiju minimalne dimenzije

potrebne za crtanje okvira oko konture zajedno sa koordinatama origina okvira. Koristeći dobivene podatke uzima se izvorni frame i na njega se crta okvir na točno mjesto s točnim dimenzijama. Ono što nije prikazano na slici je omogućavanje spremanja sadržaja u okviru pritiskom tipki 1-9 te tipki 't' i 'e' . Pritiskom tipke '1' u folder „one“ se sprema trenutna slika, isto vrijedi za ostale tipke, a tipke 't' i 'e' služe za spremanje u foldere „eleven“ i „ten“. U folder se sprema maska sadržaja u okviru.

```
#dodavanje kamere
camera = cv2.VideoCapture(0)
frames = []

while(1):
    #čitanje videa po frameovima
    _, imageFrame = camera.read()
    #pretvaranje slike iz BGR(RGB color space) u HSV(hue-saturation-value)
    hsvFrame = cv2.cvtColor(imageFrame, cv2.COLOR_BGR2HSV)

    #POSTAVLJANJE RANGE-ova BOJA I DEFINIRANJE MASKI
    white_lower = np.array([94, 80, 2], np.uint8)
    white_upper = np.array([120, 255, 255], np.uint8)
    white_mask = cv2.inRange(hsvFrame, white_lower, white_upper)

    kernal = np.ones((5, 5), "uint8")

    #Blue color
    blue_mask = cv2.dilate(white_mask, kernal)
    res_blue = cv2.bitwise_and(imageFrame, imageFrame,
                               mask = blue_mask)

    #Stvaranje konture za praćenje plave boje
    contours, hierarchy = cv2.findContours(blue_mask,
                                           cv2.RETR_TREE,
                                           cv2.CHAIN_APPROX_SIMPLE)

    boxes = []

    for pic, contour in enumerate(contours):
        area = cv2.contourArea(contour)
        if(area > 300):
            x, y, w, h = cv2.boundingRect(contour)
            boxes.append((x, y, w, h))
            xmin = x
            xmax = xmin + w
            ymin = y
            ymax = ymin + h
            #Crtanje okvira
            imageFrame = cv2.rectangle(imageFrame, (x, y),
                                       (x + w, y + h),
                                       (255, 0, 0), 2)
            cv2.putText(imageFrame, 'picture', (x, y),
                        cv2.FONT_HERSHEY_SIMPLEX, 1.0,
                        (255, 0, 0))

    cv2.imshow("Color Detection in Real-Time", imageFrame)
    cv2.imshow("BlackNWhite", white_mask)
```

Slika 3. Program tekePicture.ipynb za izradu podatkovnog skupa



Slika 4. Prikaz označene i binarne slike

Na ovaj način je izrađen podatkovni skup za svaku klasu posebno. Za svaku klasu je mijenjano osvjetljenje te su ruke pozicionirane na različitim udaljenostima od kamere za što različitije rezultate.

Zatim je 20% slika nasumično odabrano i prebačeno u novi *freshDataset_test* folder koji je iste strukture kao i *freshDataset_train* koristeći *test_dataSeparator.ipynb*.

```
import shutil
import random
import os

#dataset loading
pathToTrainTestFolder = 'pathToFolder'
train_folderUp = 'freshDataset_train'
test_folderUp = 'freshDataset_test'
train_folders = ['zero','one','two','three','four','five','six','seven','eight','nine','ten','eleven',]

input_paths = []
y = []

for j in range(len(train_folders)):
    #Dohvacanje imena slika
    input_paths = []
    y = []
    for i in sorted(os.listdir(f'{train_folderUp}/{train_folders[j]}')):
        input_paths.append(os.path.join(f'{pathToTrainTestFolder}/{train_folderUp}/{train_folders[j]}', i))
        y.append(i)
    input_paths, y

    twentyPercent = round(len(input_paths)*0.2)
    shuffled = []

    randomNumbers = random.sample(range(len(input_paths)), twentyPercent)

    for randomNumber in randomNumbers:
        shutil.move(input_paths[randomNumber], f'{pathToTrainTestFolder}/{test_folderUp}/{train_folders[j]}')
```

Slika 5. Uzimanje podataka za testni skup

3. Izrada modela

Koristeći Keras u Pythonu je izrađen model konvolucijske neuronske mreže.

```
#MODEL
model = Sequential()
model.add(Conv2D(32, kernel_size=3, activation="relu", input_shape=(64,64,1)))
model.add(Conv2D(16, kernel_size=3, activation="relu"))
model.add(Flatten())
model.add(Dense(12, activation="softmax"))

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=20, batch_size=16)

model.save('freshDatasetModel.h5')
```

Slika 6. Izrada modela konvolucijske mreže

Napravljen je model metodom *Sequential()* te je u njega prvo dodan konvolucijski sloj. *Conv2D* kao parametre prima:

- filters – broj filtera
- kernel_size - predstavlja veličinu 2D konvolucijskog prozora
- activation - koristi se za izbor aktivacijske funkcije koju se želi primijeniti nakon konvolucije(u ovom slučaju je to 'ReLu' aktivacijska funkcija)
- input_shape - tuple koji predstavlja shape slika, za 28x28 RGB slike bi bio (28,28,3) i zadnji član predstavlja broj kanala, a u ovom slučaju je (64,64,1) jer su sve slike zbog raznolikosti prethodno postavljene na 64x64 i pošto se radi o binarnim slikama kanal je 1.

Zatim je dodan još jedan konvolucijski sloj umjesto sloja sažimanja, ali ovaj puta sa 16 filtera. Nakon konvolucijskog sloja dodan je *Flattening* layer koji sve podatke pretvara u jednodimenzionalno polje i šalje ih potpuno povezanom sloju na klasifikaciju.

Na kraju je dodan potpuno povezani sloj. *Dense* kao parametre prima:

- izlaznu veličinu povezanog sloja

activation – aktivacijska funkcija, koristi se ' softmax' za precizniju klasifikaciju ulaza, dajući time vjerojatnost od 0 do 1.

Prije treniranja modela potrebno je odrediti broj epoha i batch size. Batch_size je parametar koji definira broj uzoraka koji će propagirati kroz mrežu. Manji batch size zahtjeva manje

memorije i veća je brzina učenja, ali je pogreška procjene preciznosti gradijenta veća. Ovaj parametar je potrebno prilagoditi sposobnostima računala stoga je postavljen na veličinu 16. Broj epoha označava broj prolazaka kroz dataset za treniranje te može biti između 1 i beskonačno. Svakom novom epohom se dobiva na točnosti i pouzdanosti modela. Treniranje je moguće zaustaviti u bilo kojem trenutku čak iako nisu sve epohe odrađene. Broj epoha nije određen već ga je potrebno mjenjati dok se ne dobije željena točnost. U ovom radu za broj epoha je postavljena vrijednost 20. Model za treniranje koristi prethodno napravljen podatkovni skup *freshDataset_train* iz kojeg sa slijedećim kodom podaci učitavaju u određena polja.

```
#train_dataset loading
input_paths = []
y = []
base_dir = 'freshDataset_train'
for f in sorted(os.listdir(base_dir)):
    if f == 'background':
        continue
    dir_path = os.path.join(base_dir, f)
    if os.path.isdir(dir_path):
        for i in sorted(os.listdir(dir_path)):
            input_paths.append(os.path.join(dir_path, i))
            y.append(f)
input_paths, y

#test_dataset loading
input_paths2 = []
y2 = []
base_dir = 'freshDataset_test'
for f in sorted(os.listdir(base_dir)):
    if f == 'background':
        continue
    dir_path = os.path.join(base_dir, f)
    if os.path.isdir(dir_path):
        for i in sorted(os.listdir(dir_path)):
            input_paths2.append(os.path.join(dir_path, i))
            y2.append(f)
input_paths, y2
```

Slika 7. Učitavanje putanja slika u polja

```
X = []
for img_path in input_paths:
    img = io.imread(img_path, as_gray=True)
    img = resize(img, (64, 64),
                 anti_aliasing=True)
    X.append(img)

X2 = []
for img_path in input_paths2:
    img = io.imread(img_path, as_gray=True)
    img = resize(img, (64, 64),
                 anti_aliasing=True)
    X2.append(img)
```

```
X_train = X
X_test =  X2
y_train = y
y_test = y2

X_train = np.expand_dims(X_train, 3)
X_test = np.expand_dims(X_test, 3)

temp = []
```

y_train i y_test polja su zatim pretvorena iz polja stringova(„zero“,“one“, two“…) u odgovarajuće cjelobrojne vrijednosti te je napravljeno slijedeće.

```
y_train = to_categorical(y_train, num_classes=12)
y_test = to_categorical(y_test, num_classes=12)
```

Slika 8. Priprema podataka za treniranje modela

Nakon završetka treniranja modela, isti se sprema u direktorij u kojem se nalazi projekt pod željenim imenom. Model je sada spreman za korištenje.

4. Korištenje modela

Kako bi se model mogao koristiti u stvarnom vremenu napravljena je skripta *capture.ipynb* koja radi na isti način kao i prethodno spomenuta skripta *takePicture.ipynb* koja se koristila za izradu podatkovnog skupa. Skripta je modificirana sa par linija koda stoga nema potrebe sve opet objašnjavati već samo ovo što je novo.

Prvo se učitava model.

```
#Učitavanje modela
model = load_model('freshDatasetModel.h5')
```

Slika 9. Učitavanje modela

Zatim se na modelu poziva metoda *predict_proba()* kojoj se kao parametar predaje maska koja je unutar nacrtanog okvira. Time model vrši predviđanje na onome što je unutar okvira i vraća polje s vjerojatnostima. U tom polju se nalazi dvanaest vrijednosti od kojih je svaka vrijednost odgovarajuća vjerojatnost da se radi o određenoj klasi. Npr. [0, 0, 0, 0.4, 0.6, 0, 0, 0, 0, 0, 0, 0, 0,] označava da je model predvidio da je podatak na kojem se vršilo predviđanje 40% broj tri, a 60% broj četiri dok je 0% vjerojatnost da se radi o bilo kojoj drugoj klasi. Na okvir je zatim postavljen broj koji predstavlja klasu o kojoj je riječ. Argmax() metoda vraća indeks najvećeg broja u polju koja ujedno predstavlja i klasu o kojoj se radi.

```
predicted = model.predict_proba(resize(white_mask[ymin:ymax, xmin:xmax], (64, 64),anti_aliasing=True).reshape(1,64,64,1))

if(predicted.max() >= confidence_threshold):
    imageFrame = cv2.rectangle(imageFrame, (x, y),
                               (x + w, y + h),
                               (255, 0, 0), 2)
    cv2.putText(imageFrame, transformToHandAndNumber((np.argmax(predicted))), (x, y),
                cv2.FONT_HERSHEY_SIMPLEX, 1.0,
                (255, 0, 0))
elif show_background:
    imageFrame = cv2.rectangle(imageFrame, (x, y),
                               (x + w, y + h),
                               (255, 0, 0), 2)
    cv2.putText(imageFrame, "background", (x, y),
                cv2.FONT_HERSHEY_SIMPLEX, 1.0,
                (255, 0, 0))
```

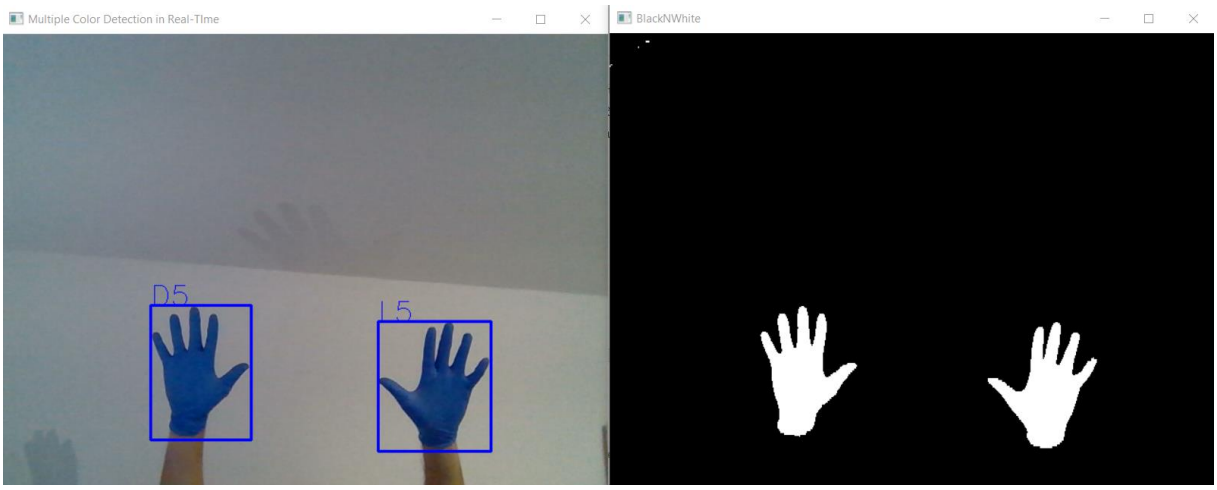
Slika 10. Predviđanje rezultata korištenjem modela i njegovo prikazivanje na okviru

Također je napravljena funkcija transformToNumberAndHand() koja prima broj i ukoliko je taj broj u intervalu [0,5] vraća string koji je kombinacija slova 'D' i tog broja, a ukoliko je broj u intervalu [6,11] vraća string koji je kombinacija slova 'L' i tog broja prebačenog u interval [0,5]. To je napravljeno radi lakšeg razumijevanja. L označava lijevu ruku a D desnu ruku. Umjesto da za lijevom rukom prikazani broj 5 na okvir ispiše '11' sada će ispisati L5.

```
def transformToHandAndNumber(number):
    for i in range(6,12):
        if str(number) == str(i):
            return 'L'+ str(i-6)

    return 'D' + str(number)
```

Slika 11. Funkcija transformToHandAndNumber

Konačno pri pokretanju skripte *capture.ipynb* pali se kamera te se otvaraju prikazani prozori i geste se mogu početi izvoditi.



Slika 12. Pokrenut program

4. TESTIRANJE MODELA

Budući da je model treniran na podatkovnom skupu koji sadržava fotografije maski ruku samo jednoga čovjeka i to u dobro osvjetljenoj prostoriji ispitat će se funkcionalnost modela na drugačijim rukama i njegova funkcionalnost u prostoriji koja je slabije osvjetljena. Za potrebe testiranja ponovno je korištena skripta *takePicture.ipynb* kako bi se u prostoriji sa slabijim osvjetljenjem napravio novi testni podatkovni skup. Taj podatkovni skup se sastoji 40 slika maski po klasi odnosno sveukupno 280 slika. Nakon toga je ponovljeno isto samo sa rukama druge osobe u normalno osvjetljenoj prostoriji. Napravljen je još jedan podatkovni skup za testiranje koji sadržava slike koje su napravljene u jednakim okolnostima kao i podatkovni skup za treniranje kako bi se mogla napraviti usporedba. Za svaki skup je određena po jedna matrica zabune i jedan klasifikacijski izvještaj. U matrici zabune su inače prikazane slijedeće vrijednosti:

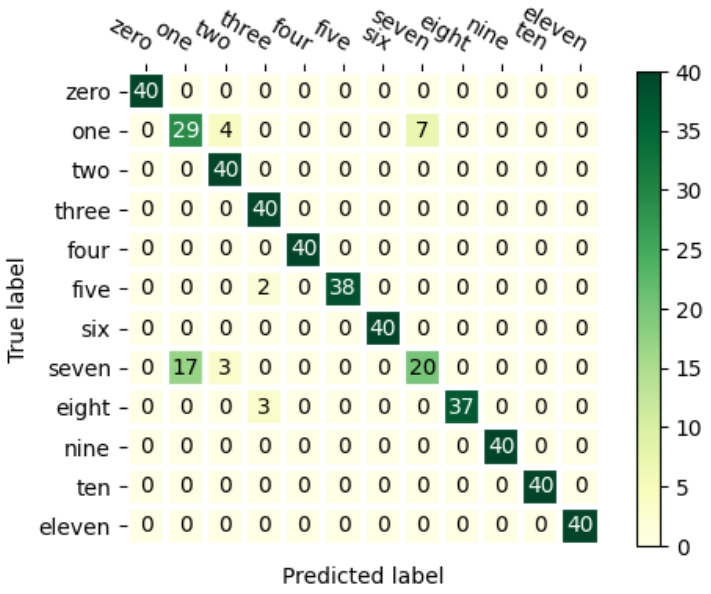
- True Positive(TP) – broj pozitivnih klasa koje je model predvidio kao pozitivne
- True Negative(TN) – broj negativnih klasa koje je model predvidio kao negativne
- False Positive(FP) – broj negativnih klasa koje je model predvidio kao pozitivne
- False Negative(FN) – broj pozitivnih klasa koje je model predvidio kao negativne

Pošto se u ovom projektu radi o višeklasnoj klasifikaciji ove vrijednosti se moraju pronaći odnosno izračunati za svaku pojedinu klasu.

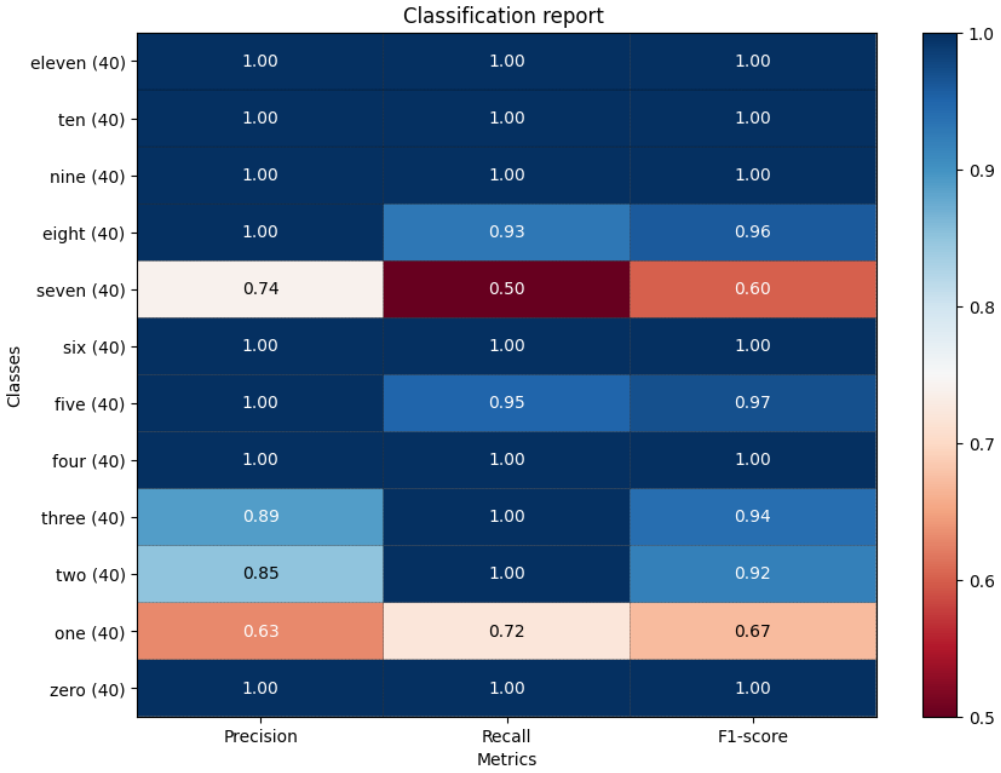
Iz matrice zabune su također dobiveni slijedeći podaci:

- Točnost (engl. accuracy) - udio točno klasificirani primjera u cijelom skupu.
- Preciznost (engl. precision) je udio točno klasificiranih primjera u skupu koje model klasificira kao klasa +(pozitivna).
- Odziv (engl. recall) - udio točno klasificiranih primjera u skupu primjera koji pripadaju klasi +.
- F1 score – vrijednost dobivena kombinacijom odziva i preciznosti
- $F1 - score = \frac{2TP}{2TP+FP+FN}$

1. Testiranje modela u dobrim uvjetima



Slika 13. Matrica zabune

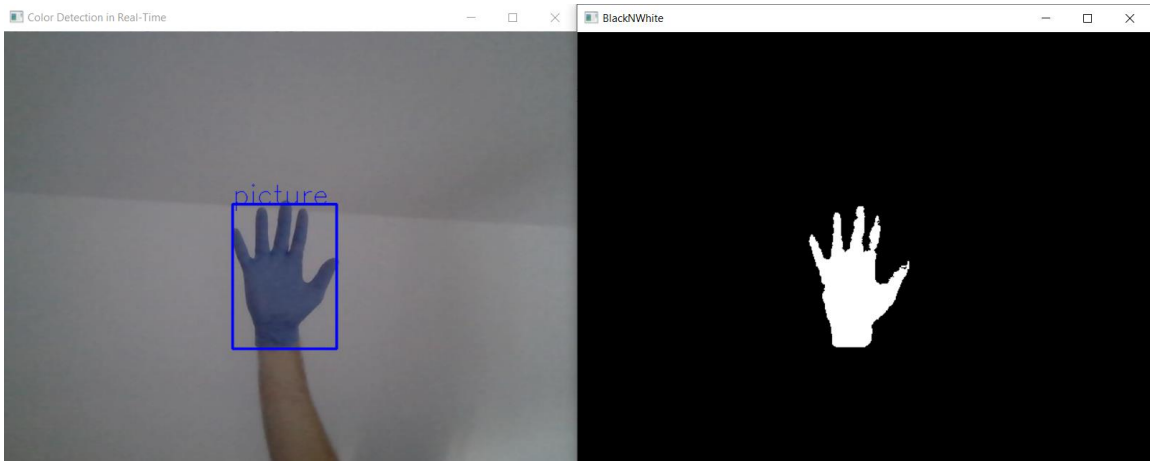


Slika 14. Matrica zabune

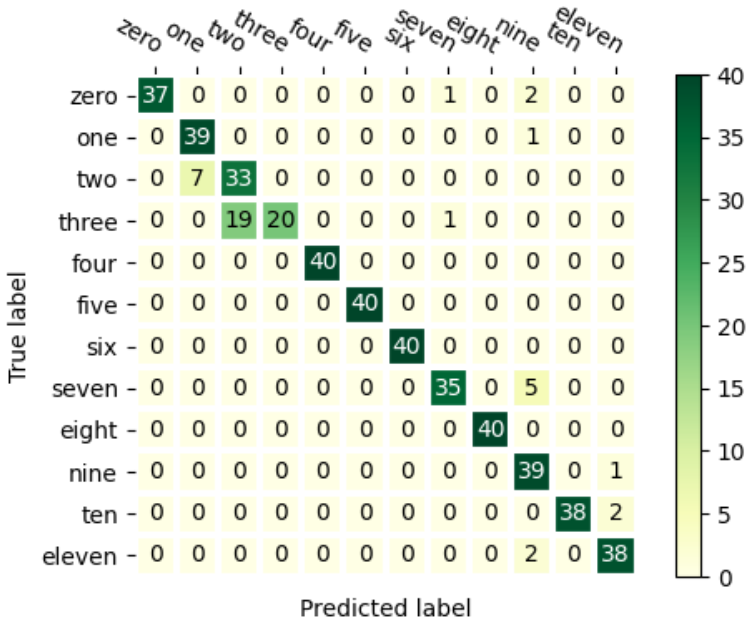
Model je prvo testiran na podatkovnom skupu koji sadržava slike uslikane u uvjetima kao i kod skupa za treniranje. Model u ovom slučaju ima točnost 92.5%. Najbolji rezultati su postignuti za klase zero, four, six, nine, ten i eleven a najlošiji za klase one i seven. Iz matrice zabune se može vidjeti da je model za klasu seven 17 puta predvidio da je riječ o klasi one. Klasa seven predstavlja gestu broja jedan prikazanu lijevom rukom što znači da je model zapravo predvidio da je riječ o desnoj ruci, a radilo se o lijevoj. Odziv modela je 76%, a f1-score 74%.

2. Testiranje modela u lošije osvijetljenom prostoru

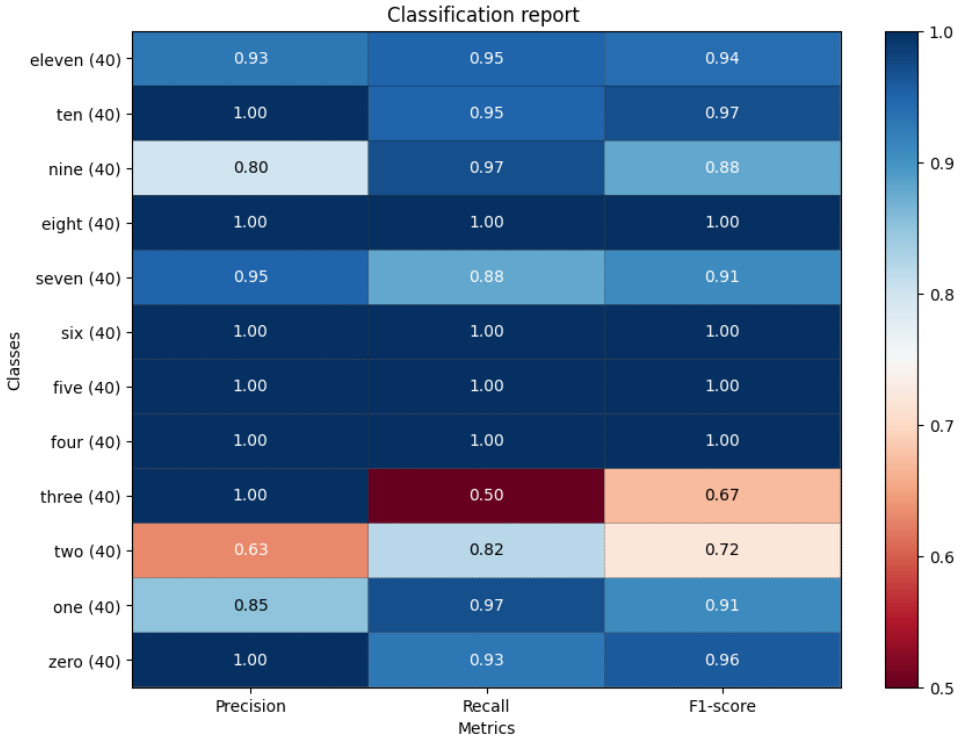
U prostoriji u kojoj se snimao testni podatkovni skup prigušeno je osvjetljene. Na slici ispod vidi se kako je to izgledalo.



Slika 15. Izrada podatkovnog skupa za testiranje u prostoru s lošijim osvjetljenjem



Slika 16. Matrica zabune

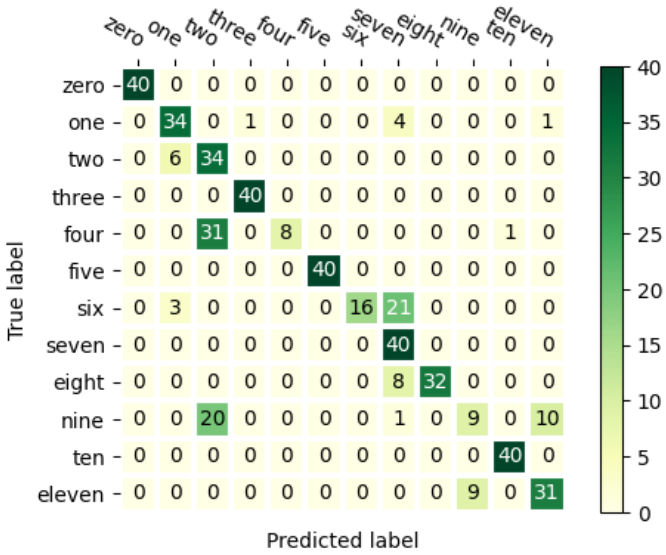


Slika 17. Klasifikacijski izvještaj

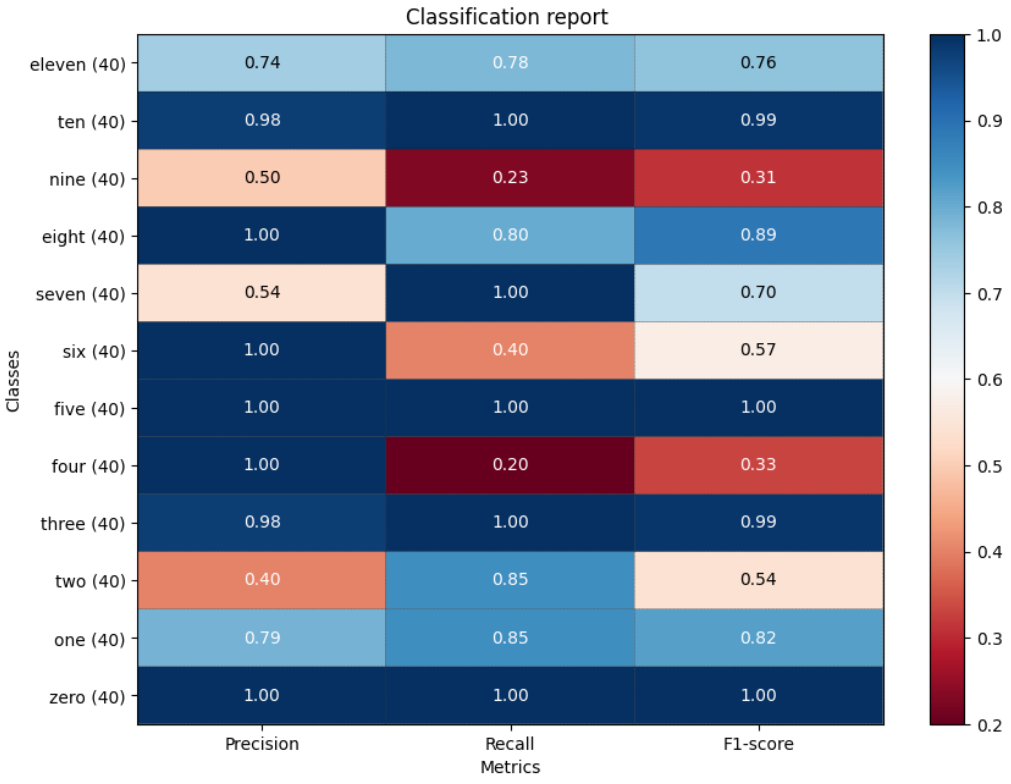
Model je u ovom slučaju imao točnost od 91.5% što je malo manje nego što je imao u dobrim uvjetima. Razlog tome je što model radi na maskama i dok god može iz slike izvući dobro

definiranu masku točnost mu neće opasti. Problematična klasa za prepoznati u ovom slučaju bila je klasa three za koju je 19 puta predvidio kao klasu two. Odziv modela i f1-score su 91%.

3. Testiranje modela na različitim rukama



Slika 18 Matrica zabune



Slika 19. Klasifikacijski izvještaj

Točnost modela u ovom slučaju je 75.8% što je dosta lošije u usporedbi s prošla dva slučaja. Veliki utjecaj na to je imala činjenica da osoba koja izvodila geste za testni skup na drugačiji način prikazuje neke od gesti naspram osobe koja je snimila skup za treniranje. Točnost modela je i dalje zadovoljavajuća s obzirom da je podatkovni skup snimljen u skoro pa idealnim uvjetima i na samo jednom paru ruku. Točnost modela u ovom slučaju se vrlo lako može popraviti tako da se napravi skup za treniranje na što više različitih ruku sa što različitijim gestama.

ZAKLJUČAK

U ovom radu objašnjen je koncept konvolucijskih neuronskih mreža koje su danas najpopularnija klasa neuronskih mreža korištenih za obradu slika. Modelirana je konvolucijska neuronska mreža, napravljen je podatkovni skup te je model treniran nad njim. Pomoću tog modela se vršila višeklasna klasifikacija. Rezultati testiranja modela u različitim situacijama su zadovoljavajući s obzirom da je podatkovni skup za treniranje dosta mal. Točnost modela se može povećati na način da se u skup za treniranje doda više različitih ruku koje na različite načine pokazuju geste. Koncept primjene rukavica određene boje i maski u programu korišten je isključivo kako bi se ruke mogle razlikovati od okoline i kako bi se ruka mogla detektirati bez obzira gdje se ona nalazila u vidnom polju kamere. Sam projektni zadatak je bio poučan u raznim područjima kao što su strojno učenje, računalni vid, te u programskom jeziku Python. Ovaj projektni zadatak također ima prostora za proširenje i napredak. Osim nadopunjavanja podatkovnog skupa može se implementirati korisničko sučelje koje bi omogućilo brzu promjenu boje koja se koristi za prepoznavanje koristeći slidere. Također bi se mogli implementirati slideri kojima bi se mijenjala svjetlina, zasićenje i još mnogo drugih parametara slike kako bi korisnik mogao odrediti najbolje postavke za najbolje rezultate u okruženju u kojem se trenutno nalazi.

LITERATURA

1. https://loomen.carnet.hr/pluginfile.php/462256/mod_resource/content/5/RUSU_LV6.pdf
2. https://towardsdatascience.com/confusion-matrix-for-your-multi-class-machine-learning-model-ff9aa3bf7826
3. https://www.fer.unizg.hr/_download/repository/SU-2015-Vrednovanje_modela.pdf
4. https://www.ibm.com/cloud/learn/convolutional-neural-networks