Računarska gimnazija

Beograd, Knez Mihailova 6

Maturski rad iz primene računara

Klasifikator crteža

Mentor:

Ivan Drecun

Učenik:

Igor Dojnov IV2

Beograd, 2022.

Sadržaj

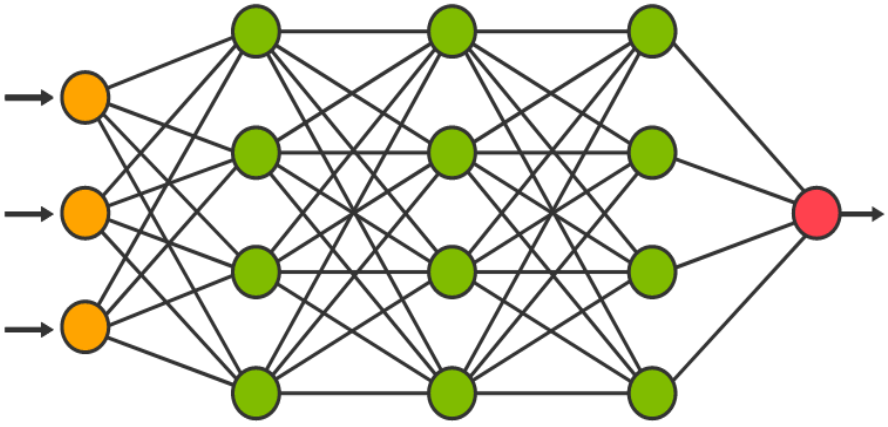
Uvod

Šta je klasifikator crteža?

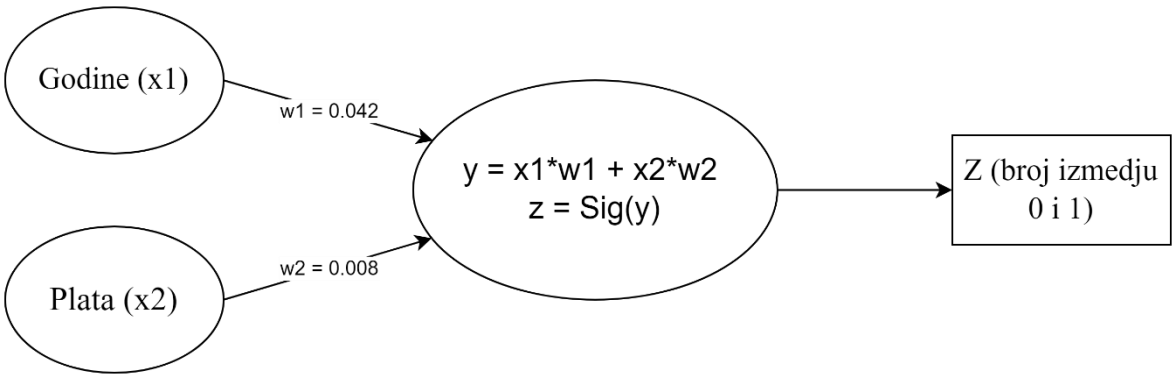
Klasifikator crteža je program koji može da pogađa šta je nacrtano na ekranu. U ovom projektu specijalno se fokusiramo na životinje i to samo na njih 6 jer je to optimalan broj podataka za koju se može napraviti klasifikator sa limitiranim podatcima i osnovnim znanjem o neuronskim mrežama.Kako bi smo to postigli koristićemo tehniku mašinskog učenja zvanu neuronske mreže koja se najčešće koristi za ovakve vrste problema.

Šta su neuronske mreže?

Neuronska mreža je tehinka mašinskog učenja koja predstavlja sistem koji se sastoji od određenog broja međusobno povezanih čvorova koje nazivamo veštačkim neuronima. Sastoji se iz više neurona koji dobijaju vise vrednosti na ulazu i proizvode jedan na izlazu. Neuronska mreža se sastoji iz nekoliko slojeva neurona. Prvi sloj je ulazni i jedini koji prima podatke iz spoljašnje sredine. Poslednji sloj je izlazni i on proizvodi konačan rezltat dok su svi izmedju slojevi skriveni i prosledjuju i obrađuju podatke od ulaza do izlaza.

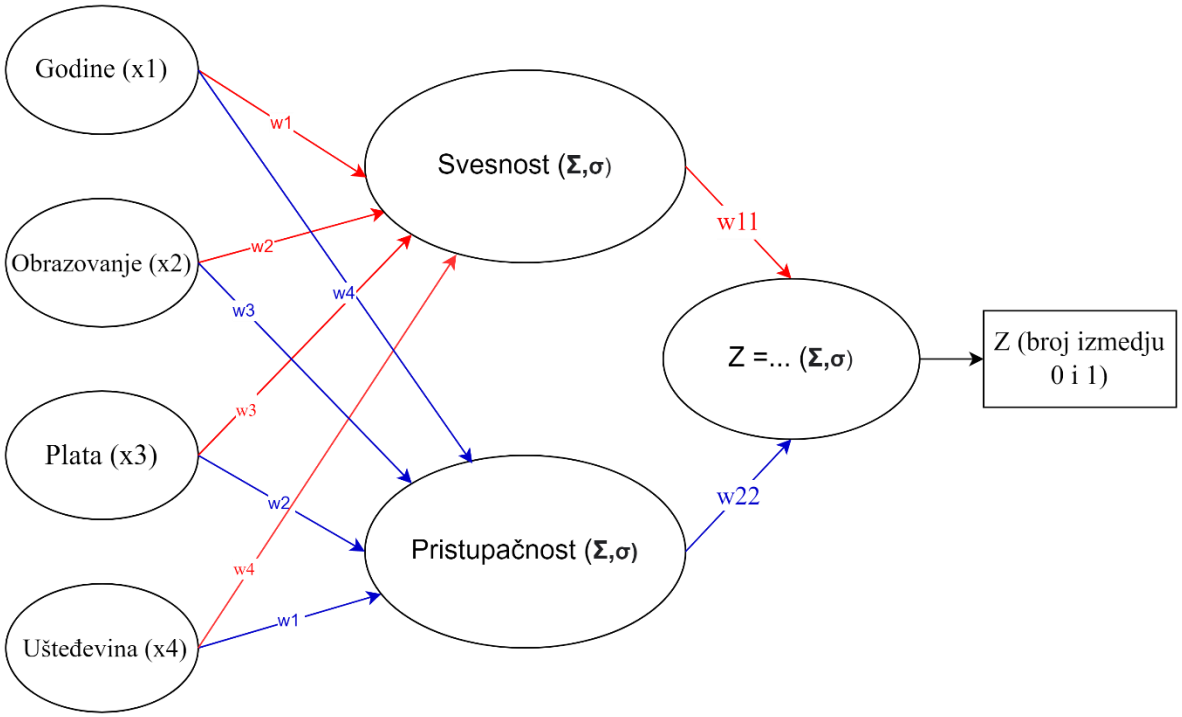


Kako rade neuronske mreže?



Na modelu iznad vidimo najprostiju neuronsku mrežu od samo jednog neurona koji dobija dva podatka (platu i godine) o nekoj osobi i na osnovu toga određuje da li će osoba kupiti životno osiguranje ili neće. U neuron pored plate I broja godina ubacuje se I težina tih podataka tj. koliko su relevantni u ovom slučaju. Zatim se u neuronu izračunava zbir proizvoda težina i parametara i onda se radi sigmoidna funkcija koja omogućava da rezltat bude izmedju 1 i 0. Na kraju dobijamo neki broj Z I možemo da kažemo da ako je Z veće od 0 I manje od 0.5 osoba nije kupila osiguranje a ako je veće od 0.5 I manje od 1 osoba jeste kupila osiguranje.

Naravno ova mreža neće biti najpreciznija jer ne prihvata veliki broj unosnih podataka podataka ali što je više proširimo davaće preciznije rezultate kao u sledećem primeru:



Ovde možemo da vidimo da pored ulaznog I izlaznog sloja imamo I jedan nevidljivi sloj neurona. On se u ovom slučaju povezuje sa svim neuronima iz susednih slojeva a koliko će zapravo svaki od njih uticati na njega odredjuje težina veze (w). Što preciznije odredimo težine to će naš model biti preciznije tako da će na primer godine i obrazovanje biti mnogo značajnije (imati veću težinu) kada određujemo svesnost dok će plata i ušteđevina imati veću ulogu u odredjivanju pristupačnosti.

Istorija i primena

Prva ideja o računarima koji rade inspirisani ljudskim mozgom nastala je jos 1943. ali će tek 1958. biti razvijen prvi neuroračunar od strane Frank Rosenblatt i Čarlsa Vajtmana. Već 80ih i 90ih godina prošlog veka ljudi će početi da shvataju značaj i važnost ove tehnologije te će početi da se uči o njoj na mnogim naprednim univerzitetima a danas je ima svugde.

Koristi se u raznim oblastima od prepoznavanja oblika do vremenske prognoze i medicinskih dijagnoza. To je i dalje poprilično nova tehnologija o kojoj tek imamo još mnogo toga da saznamo i predviđa se da će biti ključna u daljem razvoju računara i veštačke inteligencije.

Program

Platno za crtanje i meni

Kada se program pokrene možemo videti jednostavni meni sa dva dugmeta iznad kojeg se nalazi platno na koje se može nacrtati željena slika koju će računar pokušati da raspozna.

Platno je sasavljeno od 28 redova i 28 kolona kockica koje se levim klikom misa mogu bojiti u crno te tako simulira crtanje na papiru. U meniju se nalaze dva dugmeta, dugme obriši će obrisati ceo do tada nacrtan crtež tako što će vratiti sve kvadratiće u početno stanje tj. obojiti ih u belo.

```
WIN = pygame.display.set_mode((WIDTH,HEIGHT))
pygame.display.set_caption("nacrtaj nesto")


def draw(win, grid, buttons, labels):
    win.fill(BG_COLOR)
    draw_grid(win, grid)

    for button in buttons:
```

```
        button.draw(win)

    for label in labels:
        label.draw(win)

    pygame.display.update()
```

učitavamo i crtamo prozor sa njegovim kontentom

Ovaj deo koda napisan je koristeći ekstenciju za jezik pajton pod imenom pygame. Ova ekstencija jeziku dodaje mnoge komande koje olakšavaju posao pravljenja raznih igrica dok u našem slučaju koristimo komande poput:

- `pygame.display` koja pri pozivanju stvara prozor u kojem se nalazi igrica
- `pygame.mouse` uz pomoć koje određujemo da li je miš pritisnut i takodje nalazimo poziciju miša kada je pritisnut kako bismo znali gde da obojimo platno
- `pygame.event` i `pygame.quit` koristimo za početak i kraj dešavanja to jest u našem slučaju početak programa i kraj koji se poziva kada korisnik pritisne iks dugme u gornjem desnom delu ekrana

Algoritam za pogađanje

Konvertovanje slike

Baziran je na već pomenutim neuronskim mrežama. Ideja je da se nacrtana slika pretvori u niz podataka koji se daje neuronskoj mreži na obradu koja će imati isti broj izlaza kao i broj opcija za pogađanje (6). Na svaki izlaz izbacuje broj izmedju 0 i 1 koji pretstavlja verovatnoću da je osoba nacrtala taj crtež. Na kraju se uzima opcija sa najvećim procentom i pravi se pretpostavka da je osoba crtala taj crtež.

Najpre sliku pretvaramo u niz cifara koje mogu da budu 0 i 255. Vrednošću 0 predstavljamo belu, a vrednošću 255 crnu boju. To radimo tako što prolazimo kroz celo platno dimenzija 28x28 i svaki kvadratić pretstavljamo jednom cifrom koju dodajemo u niz.

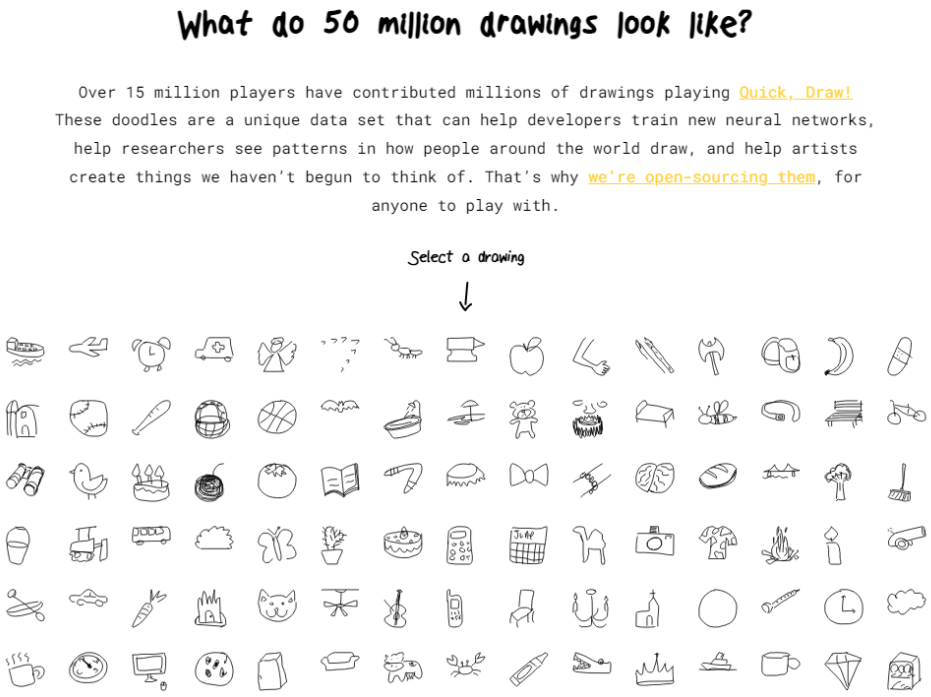
```
X = []

def convert(row, col):

    for row in range(28):
        for col in range(28):

            if grid[row][col] == WHITE:
                X.append(0)

            elif grid[row][col] == BLACK:
                X.append(255)

    return X
```

Tensorflow i Keras

Tensorflow i keras su biblioteke (API-ji) koji se fokusiraju na mašinsko učenje. Tensorflowov glavni zadatak je da obezbeđuje osnove za razvoj i isporuku rešenja za mašinsko učenje sa velikom brzinom iteracije. Keras omogućava da svoj program pokrenete na platformamam poput telefona zahvaljući velikim GPU klasterima.

Primena neuronske mreže

Za početak je potrebno izvršiti treniranje mreže, čime se dobija težina njenih veza. To radimo tako što je puštamo da prodje kroz 30000već postojećih crteža od svake opcije koju imamo. Ti crteži uzeti su iz baze podataka guglove „Quick, draw“ igrice koja je napravljena s namerom prikupljanja podataka. Radi tako što ljudima zadaje temu i daje im 20 sekundi da je nacrtaju dok računar pokušava da pogodi šta su nacrtali. Sve te slike su prikupljene u veliku bazu podataka koja je otvorenog koda.



Slike učitavamo u **.npy** formatu što ih odmah pretvara u nizove od 784 člana. Učitavamo 6 različitih skupova crteža, limitiramo ih na 30000 po skupu i označavamo sa brojevima od 0 do 5. Za to koristimo „numpy“ biblioteku.

```
X_cat = np.load("Desktop/Zavrsni_Igor_Dojnov/utils/pod/cat.npy")
X_cat = X_cat[:30000]
y_cat = np.zeros(X_cat.shape[0])
```

Učitavamo crteže mačke i označavamo ih sa 0

Nakon toga sve podatke stavljamo u niz X i pravimo njemu simetričan niz y koji sadrži sve odgovarajuće oznake i ta dva niza uz pomoć Keras biblioteke nasumično delimo na deo za treniranje i deo za testiranje.

```
X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y,
test_size = 0.2)
```

Delimo podatke na 80% za trening i 20% za test

Zatim oznake pretvaramo u vektore od pet nula i jedne jedinice koja je pozicionirana na odgovarajućem mestu tako da na primer oznaka 3 postaje vektor [0, 0, 0, 1, 0, 0]. To radimo zato što znamo da će naša mreža proizvesti rezultat koji će isto biti šestočlani vektor. Ako je na primer rezultat [0.1, 0.03, 0,045, 0.0001, 0.75, 0.02, 0.11], tada zaključujemo da ima 75% šanse da je to slika sa oznakom 4.

Ovde definišemo model neuronske mreže:

```
model = Sequential()
model.add(InputLayer(input_shape = input_shape))
model.add(Dense(units = 128, activation = 'relu'))
model.add(Dense(units = 64, activation = 'relu'))
model.add(Dense(n_classes, activation = 'softmax'))
```

Koristimo redni (sequential) tip modela koji učitavamo iz keras biblioteke. Takav tip modela za svaki ulaz ima i jedan izlaz.

Zatim dodajemo dva nevidljiva sloja i jedan izlazni:

- Prvi sloj ima 128 neurona i ima „relu“ aktivaciju koja prosto proverava da li je rezultat manji od 0 i ako jeste pretvara ga u 0 dok ako je veći samo vraća taj broj
- Drugi sloj radi isto to samo je duplo manji
- Treći sloj ima 6 neurona i ima „softmax“ aktivaciju koja predviđa raspodelu verovatnoća i to tako da kad se svi izlazi saberu daju 1

Posle toga kompajliramo model:

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

Ovde koristimo optimajzer „adam“ koji se generalno koristi za mreže koje primaju puno podataka. Takodje izračunavamo gubitak koji želimo da što više smanjimo treningom i procenat tačnosti koji želimo da bude što bliži 100%.

Na kraju pokrećemo trening:

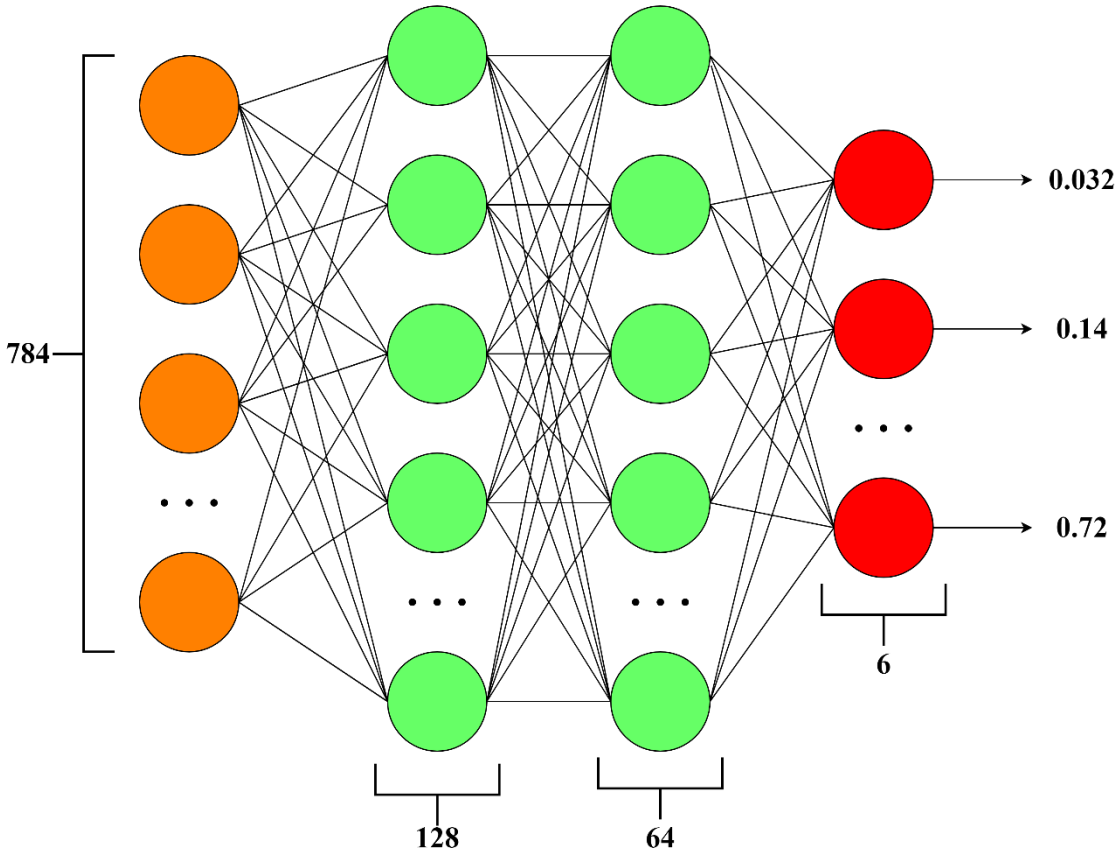
```
model.fit(X_train, y_train, epochs = 12, batch_size = 128, validation_split =
0.2)
```

Program trenira 128 slika odjednom i 12 puta prolazi kroj celu bazu podataka. Ostvaljamo još 20% za validaciju treninga to jest, kao za testiranje treninga.

Potrebno je još samo sačuvati model kako ne bismo morali da ga pokrećemo iznova svaki put kada testiramo program jer ne želimo da svaki put moramo da učitavamo skoro gigabajt slika kako bi smo igrali igru jer su nam ti podatci korisni samo pri treningu.

```
model.save("Deskop/Zavrsni_Igor_Dojnov/utils/model3.h5")
```

Čuvamo ih sa ekstencijom **.h5** koji su napravljeni za čuvanje velikih količina podataka.



Prikaz napravljenog modela neuronske mreže

Predvidjanje

Klikom na dugme pogodi pozivamo funkciju za konvertovanje i zatim na osnovu učitanog modela pravimo pretpostavku za niz X. To vraća vektor sa procentima koji uz pomoć „np.argmax“ funkcije pretvaramo u vektor sa jedinicom na određenom mestu i 5 nula te to pretvara u jednobrojnu oznaku. Zatim na text koji se nalazi na oznaci na kojoj piše „ovo je:“ dodajemo ime životinje koje stoji pored odgovarajućeg broja.

```
if button.text == "Pogodi":
                        convert(0, 0)
                        br = (np.argmax(loaded_model.predict([X])))

                        if br == 0:
                            labels[0].text = "Ovo je: Mačka"
                        elif br == 1:
                            labels[0].text = "Ovo je: Papagaj"
                        elif br == 2:
                            labels[0].text = "Ovo je: Medved"
                        elif br == 3:
                            labels[0].text = "Ovo je: Slon"
                        elif br == 4:
                            labels[0].text = "Ovo je: Riba"
                        elif br == 5:
                            labels[0].text = "Ovo je: Žirafa"
```

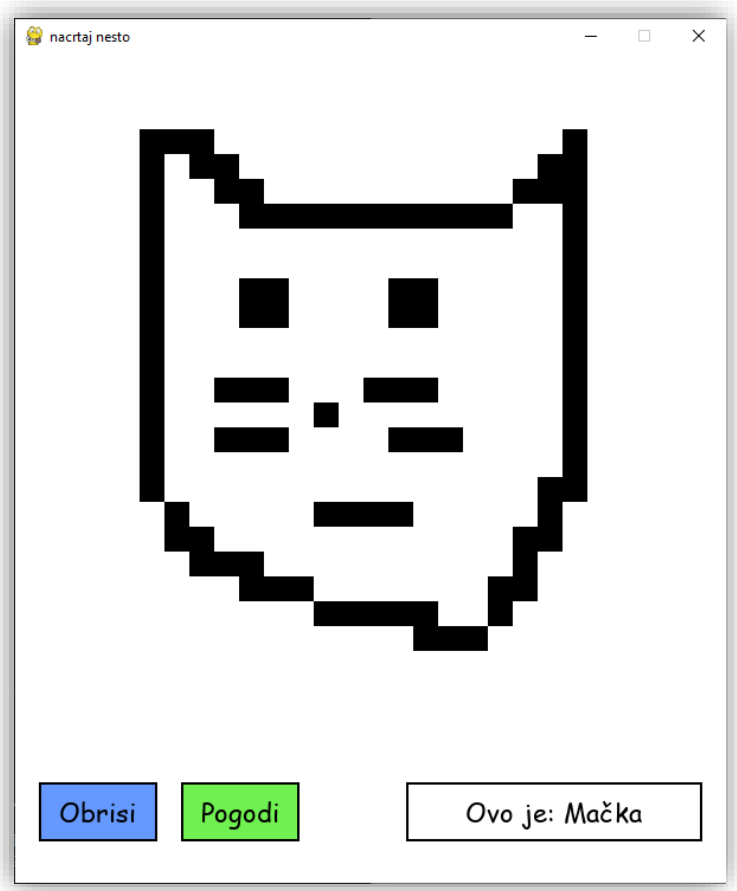
prikaz koda

Tačnost

Nakon treninga izračunato je da program ima 81% tačnosti i gubitak od 0.55. To nije perfektono s obzirom da će od 2 od 10 puta promašiti o čemu je reč. Nakon nekoliko testova primećeno je da najbolje pogadja papagaja dok mnogo češće zna da ne prepozna ribu ili medveda. Zbog ovakvih razloga je program limitiran na samo 6 životinja jer bi ga svaka dodatna klasa samo još više zbunila i smanjila mu tačnost.

Zaključak

Pored toga što je jedna od nauka u koju se najviše ulaže i koja se sve više razvija, mašinsko učenje može da bude i veoma zanimljivo jer ima veliki broj različitih oblasti u kojima može da se upotrebi, kao i ogroman prostor za napredak i nove ideje. Pravljenje nečega kao što su neuronske mreže postaje sve lakše sa brojnim bibliotekama, tutorijalima i radovima na internetu te i neko ko se pre nije susretao sa njima može sam sebe brzo.



Literatura

https://github.com/matf-ml/materijali-sa-vezbi-2022/blob/main/11-CNN/01-MNIST%20CNN%20klasifikacija.ipynb

https://quickdraw.withgoogle.com/data

https://app.diagrams.net/#G1C0dVcDFthpRQhKIsE02jEN6yCXqf_YSn

https://sh.wikipedia.org/wiki/Neuronske_mre%C5%BEe

http://solair.eunet.rs/~ilicv/neuro.html

https://keras.io/about/