Matematički fakultet, Univerzitet u Beogradu

Prepoznavanje gestikulacija u svrhe razlikovanja simbola znakovnog jezika

Autor:

Milica Vučić

26. septembar 2024.

Sadržaj

1 Uvod

Prepoznavanje gestikulacija rukama već godinama predstavlja jednu od oblasti istraživanja prilikom interakcije čoveka sa računarskim sistemima. Tehnologije iz godine u godinu napreduju sve više te su razvijeni sistemi gde ljudi na osnovu pokreta ruku mogu da kontrolišu razne uređaje, igraju video igre, budu deo virtuelne realnosti i slično.

Jedna od posebno interesantnih i značajnih primena prepoznavanja gestikulacija javlja se u oblasti prepoznavanja simbola znakovnog jezika. Napredovanjem sistema mašinskog učenja, istražiavanja podataka i dubokog učenja razvijeni su sistemi koji mogu vršiti precizno predviđanje simbola čak i u realnom vremenu. Ovakvi sistemi osim pokreta ruku koriste i mimiku ljudskog lica kako bi preveli gestove u govorni ili pisani oblik. Činjenica je da znakovni jezik predstavlja barijeru u komunikaciji njegovih korisnika i šire populacije te često dovodi do društvene izolovanosti i nesporazumevanja. Cilj razvoja ovih sistema jeste upravo taj da se prepreke u međusobnoj komunikaciji smanje i podrže autonomiju zajednice ljudi koji znakovni jezik aktivno koriste.

U ovom radu, bavićemo se prepoznavanjem simbola američkog znakovnog jezika sa fotografija objedinjenih u skup podataka Sign Language MNIST. Konkretno, rešavamo klasifikacioni problem u kojem je neophodno da se svakoj fotografiji dodeli klasa (simbol) kojoj ona odgovara što preciznije. Izložićemo tehnike bazirane na konvolutivnim neuronskim mrežama i demonstrirati ih uz pomoć programskog jezika Python i pratećih biblioteka kao što su tensorflow, keras i slične.

2 O skupu podataka

Skup podataka **Sign Language MNIST** predstavlja jedan od skupova iz MNIST familije. Kreiran je sa ciljem da pomogne u razvoju algoritama koji će moći da vrše detekciju, a zatim i ispravno prepoznavanje simbola američkog znakovnog jezika. Skup podataka sadrži 34627 instanci koje su već podeljene u dva skupa:

- trening skup: sadrži 27455 instanci;
- test skup: sadrži 7172 instance.

Instanca predstavlja jednu fotografiju koja je predstavljena u crno beloj boji (1 kanal). Fotografije su veličine 28x28, odnosno svaka sadrži 784 piksela u totalu. Za svaku fotografiju iz skupa imamo obezbeđenu i oznaku odgovarajuće klase. Inicijalno, labele su reprezentovane brojevima od 0 do 24,

Figure 1: Američki znakovni jezik

s tim što je neophodno obratiti pažnju da nisu sva slova prisutna u skupu. Naime, slova J (9. po redu), kao i slovo Z (25. po redu) ne nalaze se među fotografijama iz razloga što njih nije moguće predstaviti bez pokreta ruke, te ih nije moguće ni predstaviti na statični način.

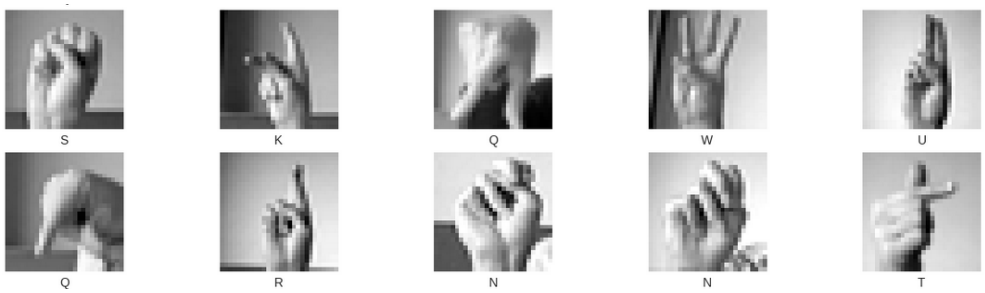

Figure 2: Fotografije sa pridruženim oznakama klasa

Uz pomoć grafičke reprezentacije u vidu histograma možemo zaključiti da su nam klase poprilično balansirane, odnosno da nemamo neka odskakanja u brojnosti elemenata neke od klasa što nam svakako olakšava posao oko kreiranja modela. Da nije bio takav slučaj, morali bismo da primenimo neku od tehnika za uzorkovanje.

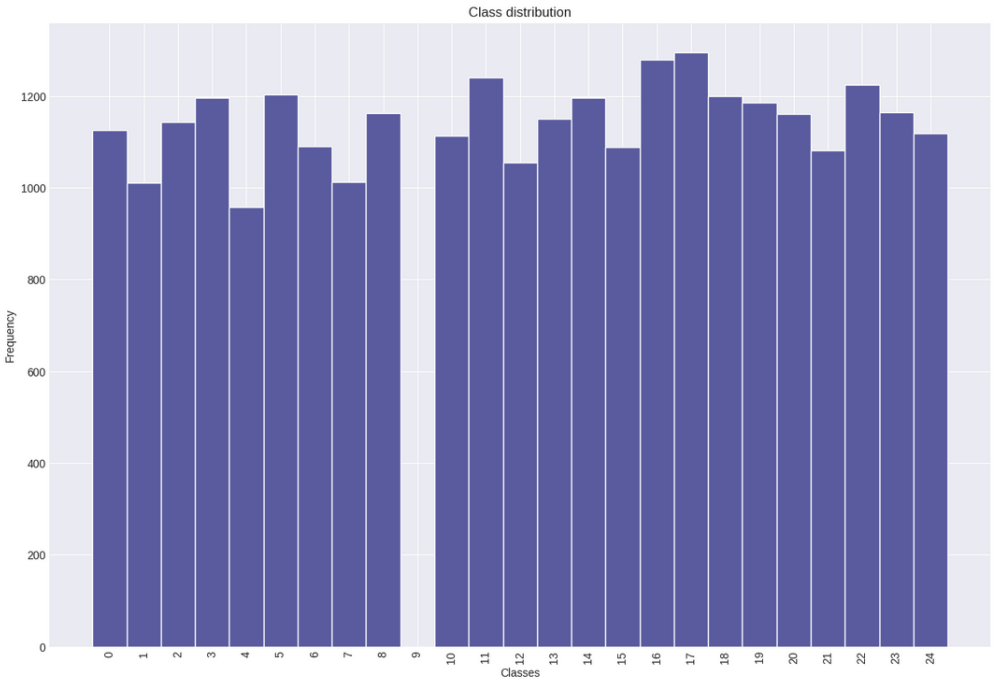

Figure 3: Distribucija klasa

3 Pretprocesiranje podataka

Pretprocesiranje podataka predstavlja veoma važan korak u procesu kreiranja kvalitetnog modela mašinskog učenja. Naime, pretprocesiranjem mi vršimo detaljniju pripremu podataka tako da oni odgovaraju potreba odgovarajućeg algoritma. Kako će dalji fokus ovog rada biti na konvolutivnim neuronskim mrežama i modelima koji njih sadrže u osnovi, primenićemo poprilično jednostavne korake pretprocesiranja:

- **Promena dimenzija ulaznih podataka**: svaka instanca želimo da predstavimo kao matricu dimenzija 28x28;
- **Skaliranje piksela na opseg [0, 1]**: želimo da svi podaci tj. pikseli budu na istoj skali kako bismo ubrzali proces konvergencije modela, a kasnije i njegove rezultate na test podacima;
- **Konvertovanje labela u binarne nizove**: pošto radimo višeciljnu klasifikaciju, neophodno je svakoj klasi pridružiti vektor koji će na i-tom mestu imati jedinicu, a na svim ostalim nule, gde i predstavlja numeričku oznaku konkretne klase.

```
train_data = train_data.reshape(-1, IMG_SIZE, IMG_SIZE, 1) /
    255.0
test_data = test_data.reshape(-1, IMG_SIZE, IMG_SIZE, 1) /
    255.0

from sklearn.preprocessing import LabelBinarizer

label_binrizer = LabelBinarizer()
label_binrizer.fit(train_labels)

train_labels = label_binrizer.transform(train_labels)
test_labels = label_binrizer.transform(test_labels)
```

Listing 1: Pretprocesiranje podataka

4 Kreiranje modela

U ovom radu, fokus će biti na kreiranju modela zasnovanih na **konvolutivnim neuronskim mrežama** koje se već godinama pokazuju kao veoma korisno sredstvo za rad sa fotografijama. Takođe, poredićemo rezultate dobijenih modela sa nekim modelima koji su već napravljeni, trenirani kao i korišćeni za rešavanje različitih problema sa veoma dobrim performansama.

4.1 Ukratko o konvolutivnim neuronskim mrežama

Konvolutivne neuronske mreže predstavljaju kateogoriju modela dubokog učenja koji su specijalizovani za procesiranje podataka koji su struktuirani u matrice, kao što su na primer fotografije. Zahvaljujući konvolutivnim neuronskim mrežama, ostvareni su značajni rezultati u različitim oblastima mašinskog učenja, a jedna od najistaknutijih je oblast računarskog vida (*engl. Computer Vision*).

Najosnovnija arhitektura neuronske mreže sastoji se iz narednih komponenti:

- **Konvolutivni slojevi**: može biti jedan, a najčešće ih ima više. Ovi slojevi služe da se kerneli (ili kako se često nazivaju filteri) primenjuju na ulazne podatke kako bi se uočili šabloni kao što su ivice, teksture, oblici i slično. Svaki filter proizvodi novu matricu koja se fokusira na konkretnom obrascu i tako dobijamo tenzore;
- **Aktivacione funkcije**: nelinearne funkcije;
- **Agregacioni slojevi**: slojevi koji služe za agregaciju informacija u cilju smanjenja izračinavanja i količine parametara u mreži. Jedan od

najčešći primer agregacije je agregacija maksimumom, međutim postoje i drugi načini da se ona realizuje;

- **Potpuno povezani slojevi**: reprezentuju klasičnu potpuno povezanu neuronsku mrežu sa ulaznim, izlaznim i jednim ili više skrivenih slojeva.

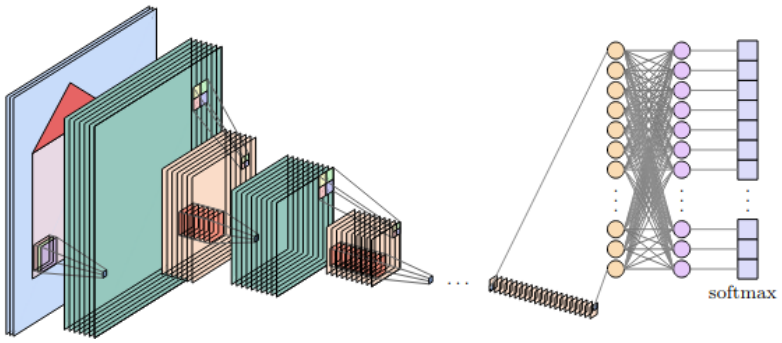

Figure 4: Arhitektura konvolutivne neuronske mreže

4.2 Jednostavna konvolutivna neuronska mreža

```
def simple_cnn_model():
    simple_cnn = models.Sequential([
        Conv2D(32, (3, 3), activation= 'relu', input_shape=
   (28, 28, 1)),
        MaxPooling2D((2, 2)),
        Conv2D(32, (3, 3), activation= 'relu'),
        Flatten(),
        Dense(63, activation= 'relu'),
        Dense(NUM_OF_CLASSES, activation= 'softmax')
    ])

    return simple_cnn
```

Listing 2: Arhitektura jednostavne neuronske mreže

Kao početni model kreiramo jednostavnu konvolutivnu neuronsku mrežu. Ona se sastoji od narednih komponenti:

- Konvolutivnog sloja koji od parametara ima 32 filtera, veličinu kernela (3, 3) i kao aktivacionu funkciju koristi ReLU.
- MaxPooling sloj za agregaciju koji će biti matrica dimenzije 2x2.
- Konvolucioni sloj isti kao i prethodno opisani.
- Sloj za izravnjanje matrice u jedan vektor.
- Dva skrivena sloja:
  - Prvi sa 63 neurona i aktivacionom funkcijom ReLu
  - Drugi koji predstavlja ujedno i izlazni sloj te iz tog razloga ima 24 neurona i kao aktivaciju koristi softmax funkciju.

4.3 Duboka konvolutivna neuronska mreža

```
def deep_cnn_model():
    deep_cnn = models.Sequential([
        Conv2D(64, (3, 3), activation= 'relu', input_shape= (
   IMG_SIZE, IMG_SIZE, 1)),
        MaxPooling2D((2, 2)),
        Dropout(0.10),

        Conv2D(64, (3, 3), activation= 'relu'),
        MaxPooling2D((2, 2)),
        Dropout(0.10),
```

```
10
11          Conv2D(64, (3, 3), activation= 'relu'),
12          Flatten(),
13          Dense(32, activation= 'relu'),
14          Dropout(0.10),
15          Dense(NUM_OF_CLASSES, activation= 'softmax')
16      ])
17
18      deep_cnn.compile(optimizer= 'adam', loss=
    CategoricalCrossentropy(), metrics= 'accuracy')
19      return deep_cnn
```

Listing 3: Arhitektura jednostavne neuronske mreže

U drugom pokušaju vršimo kreiranje duboke konvolutnivne neuronske mreže, odnosno mreže koja će imati više slojeva. Takođe, ovde koristimo i tehniku koja je u mašinskom učenju poznata pod nazivom *dropout*. Ova tehnika podrazumeva da su nasumično odabrani neuroni, ili bolje rečeno određeni procenat nasumično odabranih neurona, ignorisani u toku faze treniranja modela. Na ovaj način težimo ka tome da sprečimo preprilagođavanje modela i samim tim omogućimo da predviđanje novih vrednosti daje veću tačnost. Arhitektura tako kreiranog modela identična je kao i u prethodnom slučaju sa tim što između dela konvolucije i dela koji predstavlja potpuno povezanu neuronsku mrežu imamo sledeće slojeve:

- Konvolucioni sloj sa 64 filtera dimenzije 3x3 i aktivacionom funkcijom ReLu.
- Agregacioni sloj dimenzije 2x2.

Između svakog agregacionog i konvolutivnog sloja dodajemo dropout sloj koji ima ulogu da vrši regularizaciju odbacivanjem 10% od broja neurona koji učestvuju u trening procesu. Dropout sloj dodajemo i u delu sa potpuno povezanim slojevima.

4.4 Konvolutivna neuronska mreža sa podešavanjem parametara

U prethodnim pokušajima parametre naših modela kao što su brojevi neurona, broj filtera, njihove dimenzije i slično zadavali smo po nahođenju. Međutim, ispitivanje toga koliko parametri doprinose radu modela na ovakav način nije nešto što često treba praktikovati. U algoritmima mašinskog učenja vrednosti parametra igraju jako bitnu ulogu prilikom kontrukcije adekvatnog modela. Zbog toga koristimo gotove algoritme koji imaju upravo tu ulogu - pronalaženje parametara pomoću kojih će model davati najbolje rezultate.

Jedan od tih algoritama je **RandomSearch** koji je dostupan u okviru keras frameworka.

```
def create_model(hyperparameters):
    model = models.Sequential([
        Conv2D(hyperparameters.Int('num_filters_l1', 32, 256, step= 32),
                                    kernel_size=(3, 3), activation='relu', input_shape=(IMG_SIZE, IMG_SIZE, 1)),
        MaxPooling2D((2, 2)),

        Conv2D(hyperparameters.Int('num_filters_l2', 32, 256, step=32),
                                    kernel_size=(3, 3), activation='relu'),
        MaxPooling2D((2, 2)),

        Flatten(),
        Dense(hyperparameters.Int('dense_units1', 32, 256, step=32), activation='relu'),
        Dense(hyperparameters.Int('dense_units2', 32, 256, step=32), activation='relu'),
        Dense(NUM_OF_CLASSES, activation='softmax')
    ])

    model.compile(optimizer= 'adam', loss= CategoricalCrossentropy(), metrics= 'accuracy')
    return model
```

Listing 4: Učenje parametara

Model inicijalno definišemo analogno prethodnim arhitekturama s tim što možemo primetiti da neki paramteri kao što su u ovom slučaju broj filtera i broj neurona u skrivenim slojevima nisu dati. Na ovaj način mi kroz proces treniranja želimo da naučimo koji parametri će nam davati rezultate kakve želimo. Jedan način da ovo realizujemo bilo bi iscrpno pretraživanje celokupnog prostora mogućih vrednosti za parametre (može se realizovati uz pomoć **GridSearch**) ali nam to ne bi bilo vremenski efikasno. Zbog toga koristimo malo poboljšanje koje je dato kroz **RandomSearch**, ali iako imamo brže izvršavanje, nemamo garanciju da ćemo uvek pronalaziti najoptimalnije moguće parametre.

4.5 ResNet50

Prilikom kreiranja modela koji su zasnovani na konvolutivnim neuronskim mrežama često se težilo ka tome da se vrši kreiranje mreža sa što većim brojem slojeva kako bi se postigle bolje performanse modela. Međutim,

povećanjem broja slojeva dolazi do povećanja broja parametara modela, a samim tim hardverski zahtevi postaju veći. Takođe, dešavalo se da i performanse značajno opadaju iako je ideja bila da se postigne potpuno obrnut efekat. **ResNet50** je jedan od modela iz ResNet familije koji predstavlja duboko povezanu neuronski mrežu sa 50 slojeva koja je veoma brzo postala popularna među programerima širom sveta zbog svojih dobrih performansi. Arhitektura mreže uopšteno je sledeća:

- Ulazni sloj: prihvata fotografije u RGB formatu;
- Konvolutivni slojevi;
- Rezidualni blokovi: svaki plok uključuje konvolutivne slojeve, najčešće sa filterima dimenzije (3, 3), normalizaciju izlaza, ReLU aktivacionu funkciju kao i preskakanje konekcija[1];
- Slojeve agregacije;
- Potpuno povezani sloj i izlazni sloj

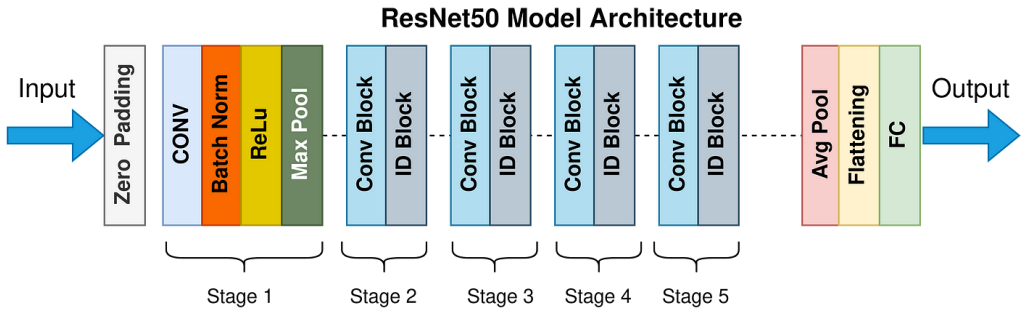

Figure 5: Arhitektura ResNet50 modela

5 Procene kvaliteta modela

Nakon što se završi proces treniranja modela, neophodno je izvršiti evaluaciju modela na novopristiglim podacima, kao i obaviti procenu kvaliteta uz pomoć

[1]Preskakanje konekcija (*engl. Skip connections*) kao što i samo ime govori podrazumeva da se u toku treniranja neuronske mreže vrši preskakanje određenih slojeva, odnosno izlaz jednog sloja postaje ulaz nekog drugog, ne nužno narednog u sekvenci slojeva. Na ovaj način kod rezidualnih neuronskih mreža rešeni su problemi opadanja kvaliteta modela odlaženjem u dublje slojeve mreže.

nekih relevantnih metrika. Najčešće korišćena metrika u slučaju klasifikacije jeste **tačnost** (*engl. accuracy/accuracy score)*. Formulu za tačnost najbolje možemo videti iz **matrice konfuzije** (*engl. confusion matrix)* koja se koristi za grafički prikaz rezultata klasifikacionih algoritama.

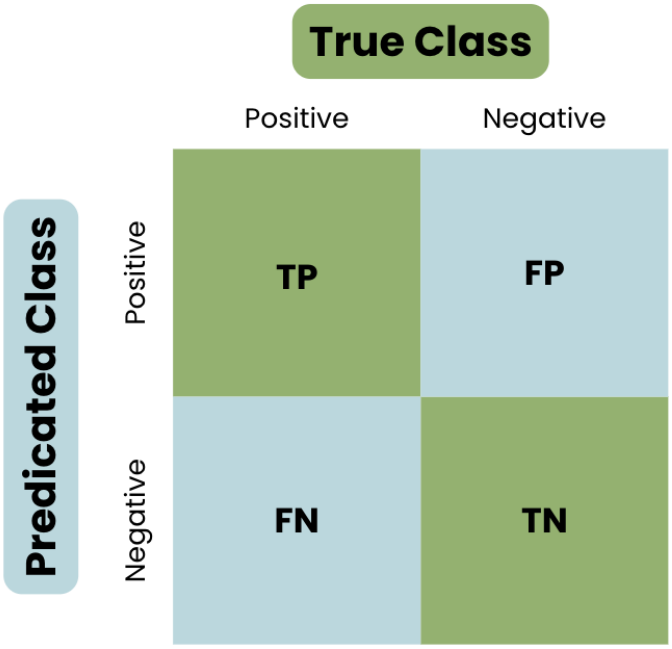

Figure 6: Matrica konfuzije

Formula za tačnost odavde izvodi se kao:

$$accuracy = \frac{\text{tačna predviđanja}}{\text{sva predviđanja}} = \frac{TP + TN}{TP + TN + FP + FN}$$

Ako želimo na osnovu grafičkog prikaza u vidu matrice konfuzije da zaključimo kvalitet klasifikacije, onda je dovoljno posmatrati njenu glavnu dijagonalu. Ukoliko se na njoj nalaze ne-nula brojevi, a po sporednim dijagonalama su nule ili nešto dovoljno njima blisko, onda znamo da klasifikator dobro radi.

Bitno je dodatno napomenuti da u slučaju našeg problema tačnost predstavlja validnu metriku za procenu kvaliteta jer su nam klase balansirane. Kod nebalansiranih klasa treba obratiti pažnju i koristiti neku drugu metriku ili prvo izvršiti neki vid uzorkovanja radi balansiranja klasa pa tek onda primeniti tačnost.

U tekstu ćemo prikazati u vidu fotografija i prodiskutovati tačnosti koje modeli dostižu na trening podacima.

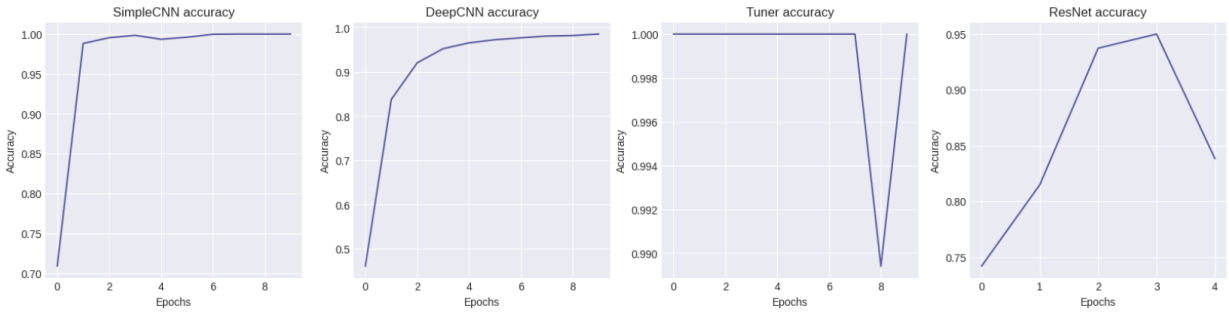

Figure 7: Tačnost koju modeli dostižu na trening podacima

Na osnovu fotografije možemo videti da kod skoro svih modela proces učenja funkcioniše tako da se tačnost kroz epohe povećava. Izuzetak pravi model ResNet kod kojeg imamo opadanje tačnosti kako se epohe primiču poslednjoj. Takođe, interesantno je da kod modela koji vrši optimizovanje hiperparametara imamo nagli pad u osmoj epohi treninga, a zatim i nagli skok tačnosti - ovo nam slikovito pokazuje koliko nasumičnost potencijalno može dovesti i do problema prilikom pronalaženja najboljih parametara.

Na test skupu smo dodatno sračunali tačnost kako bismo videli koji od modela imaju dobru sposobnost generalizacije. Rezultati su prikazani u narednoj tabeli i sortirani opadajuće po kvalitetu:

MODEL	TAČNOST
Jednostavna neuronska mreža	0.8621026213050753
Duboka konvolutivna neuronska mreža	0.9333519241494702
Model sa učenjem hiperparametara	0.9159230340211936
ResNet50	0.5492191857222533

Figure 8: Tačnost koju modeli dostižu na test podacima

6 Zaključak

U ovom radu izložili smo priču o četiri modela kao i o tome kako se oni ponašaju prilikom rešavanja problema koji smo inicijalno razmatrali. Možemo

zaključiti da većina modela na ovakvom, doduše jednostavnom skupu podataka radi zadovoljavajuće, ali je veliko pitanje da li bi bili upotrebljivi u nekim drugim situacijama kao što je na primer klasifikovanje znakova koji se prikazuju u realnom vremenu. Zadaci tog tipa predstavljaju mnogo veći izazov i definitivno jesu nešto za čije rešavanje bi nam bile neophodne kompleksnije metode i dublje poznavanje različitih sfera kako računarstva, tako i anatomije šake, detekcije objekata i slično.

Reference

[1] Predrag Janičić, Mladen Nikolić: Veštačka inteligencija, 2024.

[2] Keras API docs.

[3] ResNet architecture.

[4] Materijali sa kurseva Računarska inteligencija i Istraživanje podataka 1.