SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Lucija Mičić

Automatsko prepoznavanje znakovnog jezika

Zagreb, 2022.

Ovaj rad izrađen je na Zavodu za komunikacijske i svemirske tehnologije pod vodstvom doc. dr. sc. Daria Bojanjca i predan je na natječaj za dodjelu Rektorove nagrade u akademskoj godini 2021./2022.

Sadržaj

Popis slika

1 Motivacija

Cilj ovog rada je tehnološki premostiti jezičnu barijeru s kojom se osobe s oštećenjima sluha svakodnevno susreću. Živimo u svijetu u kojem se prijenos podataka mjeri u milisekundama, a dostava naručene hrane u sitnim minutama, ali rješenja za osobe s invaliditetom ili poteškoćama su i dalje skromna. Takvim problemima bavi se ergonomija i unazad deset godina u tome postiže sve bolje i pouzdanije rezultate. Prilikom razvoja programske podrške, sve više se pažnje posvećuje oblikovanju dizajna koji je prilagođen svima, neovisno o njihovim sposobnostima. Pristupačnost web aplikacija definirana je i smjernicama WCAG (eng. *Web Content Accessibility Guidelines*) [1] kojima se osigurava vidljivost, operabilnost, razumljivost i robusnost sadržaja. Također, većina operacijskih sustava [2] nudi razne prilagodljive opcije, kojima si korisnik može poboljšati iskustvo korištenja, kao što su npr. prilagođavanje veličine fonta, čitači zaslona i sl. Međutim, mali udio tih opcija namijenjen je gluhim i nagluhim osobama, ali i za njih postoje razne aplikacije koje im omogućavaju lakšu komunikaciju [3], kao što su npr. Ava, RogerVoice, Sound Amplifier, TapSOS, Subtitle Viewer i sl. Sve navedene aplikacije nisu prilagođene hrvatskom jeziku, što je također bila važna motivacija pri izradi aplikacije. Hrvatski savez gluhih i nagluhih djeluje još od 1921., a danas broji preko 13000 članova. Na njihovoj web stranici mogu se pronaći primjerci časopisa "Ukratko" u kojem se iznosi i problematika diskriminacije kroz povijest, ali i brojni uspjesi članova saveza. Osim znakovnog jezika, komunikacija osoba s oštećenjima sluha ima još dva oblika - ručna abeceda i simultana znakovno-oralna komunikacija, koja se sastoji od znakovanja i govorenja [4]. U hrvatskom znakovnom jeziku postoje jednoručna i dvoručna abeceda, a u ovom radu bavimo se jednoručnom. Izrada aplikacije za prepoznavanje potpunog znakovnog jezika iziskivala bi i potrebu za velikom bazom podataka, koja nažalost još ne postoji. Također, veliku prepreku stvara i činjenica da izrazi znakovnog jezika ovise i o ekspresijama lica, a ne samo pokretima ruku. Iako je abeceda mali podskup cijelog jezika, neki od navedenih izazova implementacije manifestirali su

se i pri razvoju ovog programskog rješenja, posebice pri osmišljavanju algoritma koji bi prepoznavao i znakove s dinamičkim pokretom, koji su u hrvatskom jeziku zapravo dijakritička slova te slova 'j' i 'z'. Taj problem prepoznavanja ljudsko oko i mozak jednostavno objašnjavaju, ali programsko rješenje ipak zahtijeva dodatnu analizu i izradu kvalitetne baze. Prvotna ideja bila je napraviti bazu binarnih slika, koje bi obrisom definirale pojedino slovo. Već u tom prvom koraku, može se naslutiti da će prepoznavanje dinamičkih pokreta biti veliki problem. Pri prepoznavanju bi onda trebalo napraviti masku fotografije unesene kamerom. Međutim, takvim pristupom se korisniku ograničava sloboda pokreta, jer bi dlan uvijek trebao biti u određenom kvadratiću na zaslonu, a izrada maske uvelike ovisi o pozadini i osvjetljenju prostora u kojem se korisnik nalazi, što dodatno uvodi šumove u podatke koje bi zatim trebalo analizirati. Nakon ovako definiranih zahtjeva intuitivno se nameće ideja rješavanja pomoću lokalnog koordinatnog sustava dlana, jer je ljudskom mozgu jasno da odnosi između određenih točaka na dlanu postoje i jednostavno prepoznaje ruku, odnosno dlan. I zbog ove ideje, radni okvir *Mediapipe Hands* u konačnici se pokazao kao idealno rješenje. Prvi korak bio je prikupiti podatke za bazu, odnosno *csv* datoteku u kojoj su pohranjene koordinate. Zadatak klasifikacije predali smo umjetnoj neuronskoj mreži pri izradi modela, a samo prepoznavanje definirano je na kraju rada s posebnim naglaskom na prepoznavanje slova s dinamičnim pokretom. Naime, ta se slova ne mogu klasificirati modelom neuronske mreže zato što nisu definirana statičnom gestom, već ovise i o položaju u kojem se zateknu u krajnjem trenutku. Dakle, potrebno je kamerom zabilježiti uzastopne pomake ruke i analizirati koordinate dlana.

2 Priprema baze i prilagođavanje podataka

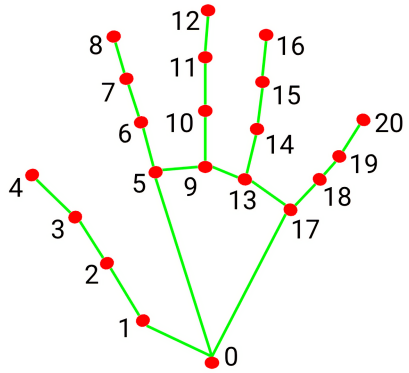
U ovom poglavlju detaljno je opisan postupak obrade koordinata i izgradnje baze te način korištenja radnog okvira *Mediapipe Hands*.

2.1 Mediapipe Hands

Mediapipe Hands [5] radni je okvir namijenjen rješavanju problema strojnog učenja pri detekciji i praćenju dlana. Temelji se na ideji dva povezana modela neuronskih mreža, od kojih je prvi Palm Detection Model, odnosno model za prepoznavanje dlana, neovisno o njegovoj veličini ili položaju na zaslonu. Drugi model je Hand Landmark Model, koji označava dvadeset jednu točku dlana te svakoj dodjeljuje x, y i z koordinate. Kao što je prikazano na slici 1 [6]. Te koordinate, i odnosi između njih ključni su faktori za točno definiranje različitosti između gesta za pojedina slova. X i Y koordinate određene su pikselima, dok Z koordinata definira relativnu dubinu, odnosno udaljenost od zaslona. Pri definiranju modela mogu se podesiti varijable kojima određujemo koliko ruku želimo prepoznati istovremeno te kolika je minimalna vjerojatnost sigurne detekcije dlana. U implementaciji ove aplikacije korištene su varijable:

1. MAX_NUM_HANDS
2. MIN_DETECTION_CONFIDENCE
3. MIN_TRACKING_CONFIDENCE

Nakon određivanja koordinata, one se pohranjuju u kolekciju MULTI_HAND_LANDMARKS. Okvir, također, nudi korištenje koordinata izraženih u metrima, koje se pohranjuju u kolekciju MULTI_HAND_WORLD_LANDMARKS. MULTI_HANDEDNESS je kolekcija u kojoj se čuvaju podaci o tome je li detektirana ruka lijeva ili desna. Varijable ove kolekcije mogu biti korisne ako se prepoznaje pokret određen dvjema rukama kao što su npr. slova hrvatske dvoručne znakovne abecede.



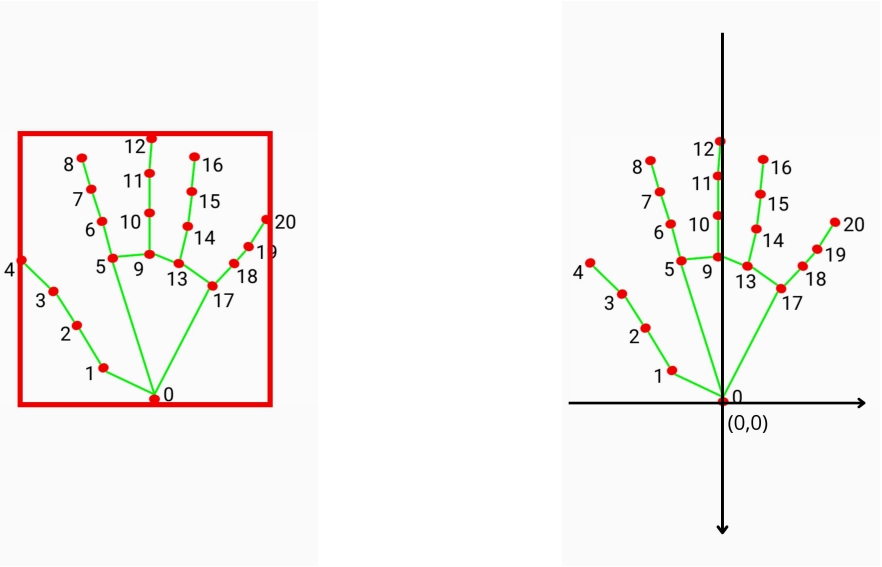
0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP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

Slika 1: Dvadeset i jedna ključna točka dlana

2.2 Prilagođavanje koordinata

Koordinate, koje se dobiju od modela za prepoznavanje trodimenzionalnih koordinata točaka na dlanu, određuju se pikselima u odnosu na veličinu otvorenog prozora, što bi značilo da se ista gesta neće jednako klasificirati ako je prikazana npr. na sredini prozora i u donjem desnom kutu. Odnosi između točaka ostali bi, naravno, isti, ali ovako neobrađeni podaci unosili bi veliku nesigurnost pri analiziranju različitosti položaja tih točaka za dinamičke pokrete. Stoga je, prije izrade modela neuronske mreže, bilo potrebno prilagoditi koordinate tako da ne ovise samo o vijednostima piksela. Prvi korak u tome bio je izdvojiti dlan i postaviti ga u novi koordinatni sustav određen ekstremnim vrijednostima x i y koordinata. Implementaciju toga možemo zamisliti kao da uzimamo najljeviju i najdesniju točku dlana, zatim, kroz vrijednost x koordinate povučemo pravac paralelan s okomitim obrubom vanjskog prozora [7]. Analogno tome radimo i za najnižu i najvišu točku dlana, samo ovaj put pravac provlačimo paralelno u odnosu na horizontalni obrub prozora kroz točku određenu s vrijednošću y koordinate. Kao što je prikazano slikom 2a. Tako dobiveni pravokutnik postaje novi koordinatni sustav za koji je ponovo potrebno izračunati i relativne koordinate svake točke u odnosu na nove osi, a to se jednostavno da napraviti budući da su odnosi između ključnih točaka dlana i dalje isti. Implementacija toga svodi se na pronalazak zapešća koje će uvijek biti ishodišna

točka s koordinatama (0,0) i u odnosu na tu točku računaju se ostale koordinate. Dakle, prvi korak je točku zapešća učiniti ishodištem. Koordinate su pohranjene u polje i određene su pikselima, stoga se zadatak svodi na oduzimanje početnih vrijednosti koordinata točke zapešća od svih koordinata ostalih točki, uključujući i samu točku zapešća. Time su vrijednosti koordinata točke zapešća postavljene na nulu, a ostalim točkama određena je relativna vrijednost u odnosu na ishodište. Idući korak je normirati koordinate, za to je potrebno pronaći maksimalnu vrijednost u novom polju koordinata i zatim sve koordinate podijeliti tom vrijednošću. Navedenim algoritmom dobiju se koordinate koje su dovoljno precizne i dobro definiraju pojedinu gestu, stoga ih se može pohraniti u *csv* datoteku, koja služi kao svojevrsna baza u svrhu daljnje analize i obrade.



(a) Izvajanje dlana

(b) Koordinatni sustav

Slika 2: Prilagođavanje koordinata

2.3 Izrada baze

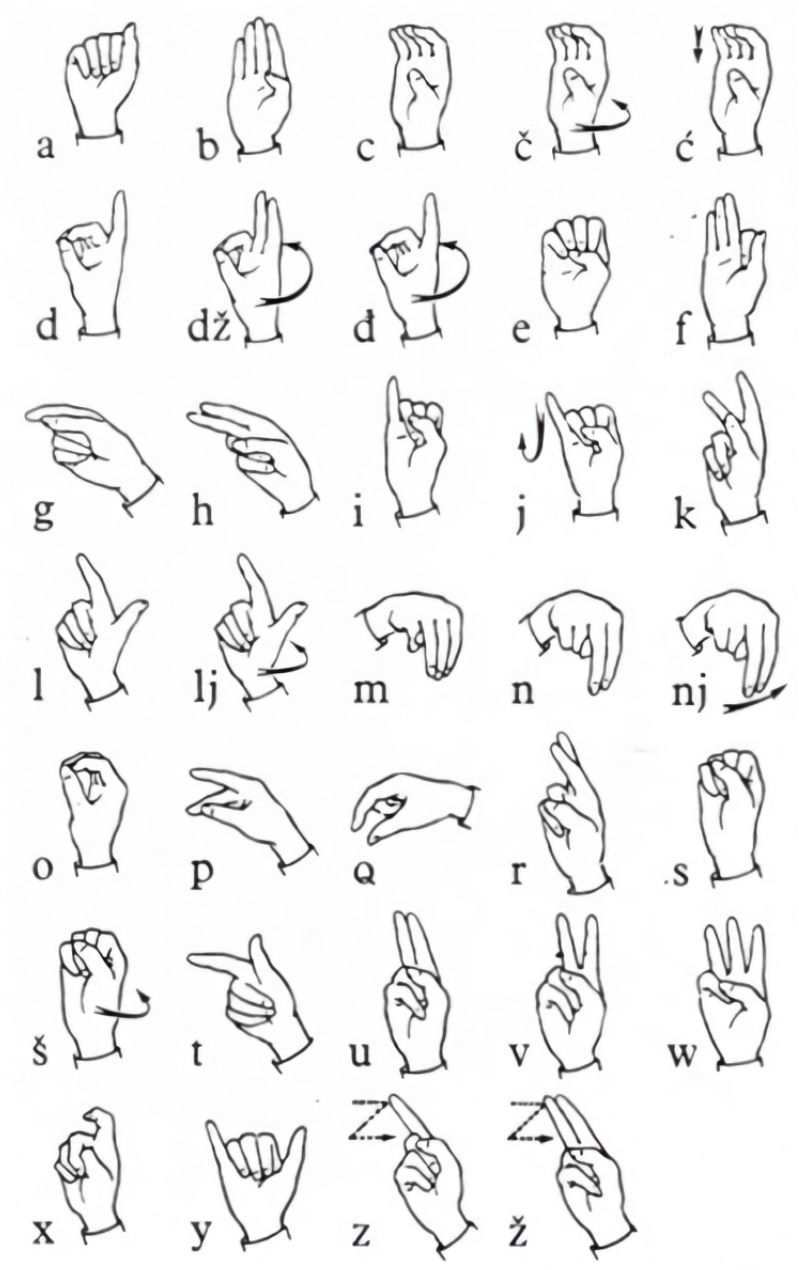
Postupak izrade baze može se podijeliti na tri zadatka:

1. Odabir slova čije ćemo koordinate pohranjivati u bazu

2. Pisanje programa za izradu baze

3. Provjera rezultata

Elementi baze su koordinate, stoga se pohranjuju u *csv* (engl. *Comma Separated Value*) datoteci, međutim, nepotrebno je pohranjivati sve znakove jer se većina dinamičkih znakova može kasnijom analizom prepoznati iz određenih slova sa statičnom gestom. Na slici 3, prikazana je hrvatska jednoručna abeceda znakovnog jezika [8]. Vidimo da su slova 'č' i 'ć' vrlo slična slovu 'c', također slovo 'đ' veoma podsjeća na slovo 'd', a slovo 'dž', zbog položaja kažiprsta i srednjeg prsta, nalikuje slovu 'u'. Ovakvim postupkom smanjuje se potrebni broj elemenata u bazi. Nadalje, slovo 'j' u početnom trenutku pokreta izgleda poput slova 'i', slovo 'lj' prikazuje se kao slovo 'l', uz rotaciju dlana unatrag, a analogno tome vidimo da su i slova 'š' i 'nj' nastali pomakom položaja ruke za slova 's' i 'n'. Za slova 'z' i 'ž', nije u potpunosti jednostavno naslutiti mogu li se prepoznati iz nekog drugog znaka. Slovo 'z' u početnom položaju slično je slovu 'd', a slovo 'ž' nalikuje slovu 'u'. Međutim, postoje razlike u kutovima koje prsti zatvaraju sa x osi, pa slova 'z' i 'ž' ipak dodajemo u skup slova za pripremu baze. Zaključno, konačni se skup slova od kojih ćemo napraviti bazu sastoji od: 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'k', 'l', 'm', 'n', 'o', 'p', 'r', 's', 't', 'u', 'v', 'z' i 'ž'. Ostale znakove prepoznavat ćemo algoritamski iz početnog skupa slova.

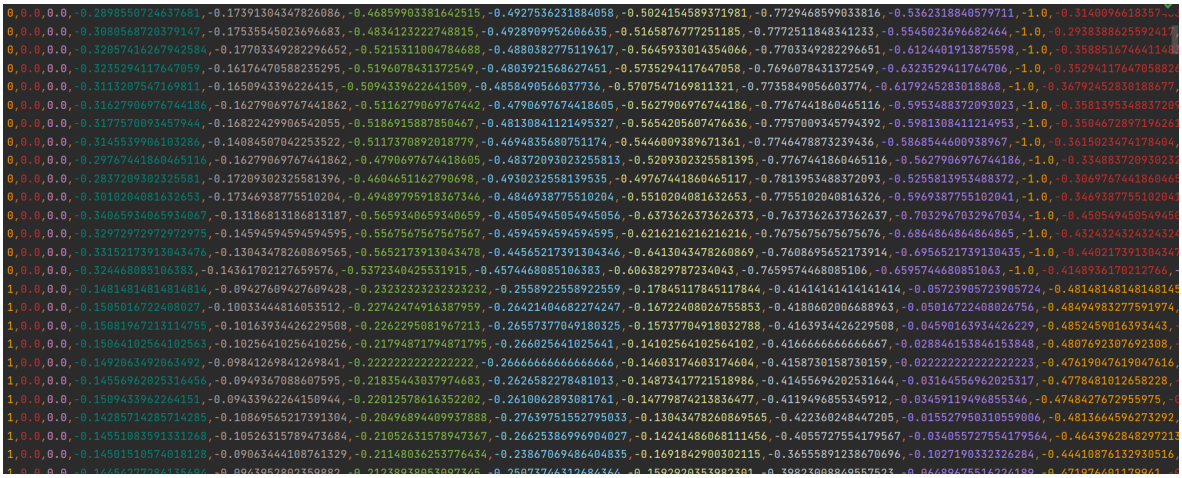


Slika 3: Jednoručna abeceda

Nakon odabira slova za bazu, potrebno je napisati program koji će takve podatke zabilježavati kamerom i pohranjivati ih u datoteku. Međutim, za razlikovanje odabranih slova, potrebno im je prvo dodijeliti indekse. U posebnu *csv* datoteku treba pohraniti

svako slovo u zaseban red, koji će poslužiti kao indeks. Za prikupljanje što boljih podatka, korištene su biblioteke *NumPy* [9], za rad s nizovima i matricama, *OpenCv* [10], za rad s kamerom, i *MediaPipe*, za praćenje i određivanje koordinata dlana. Prilikom pokretanja programa, otvara se prozor na kojem se konstantno prikazuju ulazi kamere i nad tim ulaznim podacima poziva se metoda *predict* modela za prepoznavanje ruke i izračun koordinata. Na taj se način neprestano provjerava pojava ruke na zaslonu. Ukoliko je model registrirao pojavu ruke, određuju joj se relativne i normirane koordinate koje se mogu pohranjivati u bazu. Pritiskom na tipku "k", program se prebacuje u način rada za prikupljanje podataka i očekuje od korisnika da odabirom broja od 0 do 9 odredi indeks slova za koje se prikupljaju podaci o koordinatama. Ako u bazi postoji određeni redak s tim indeksom, indeks se povećava za deset, a na taj način može se napraviti baza za proizvoljan broj elemenata. Primjerice, to bi značilo da se za prikupljanje podataka o slovu 'a' odabire broj nula, za slovo 'b' broj jedan itd., a kad se opet odabere broj nula, spremat će se podaci za slovo s indeksom deset. Koordinate koje se dobiju modelom za prepoznavanje ruke i određivanje koordinata, uređeni su parovi s dva elementa, koje treba pretvoriti u jednodimenzionalan niz s pohranjenim svim koordinatama glavnih točaka dlana i indeksima slova. Svaki redak u bazi predstavlja jedan ulaz za neuronsku mrežu. I na kraju tako uređene nizove podataka treba pohraniti u bazu, a za to postoji posebna Pythonova biblioteka *csv* [11], koja nudi funkcije za izmjenu i upravljanje *csv* datotekama.

Naposljetku, izrađena je baza, kao na slici 4, koja prikazuje prvih nekoliko koordinata za slova 'a' i 'b'. Zadnji korak jest provjeriti podatke, a svaki redak trebao bi se sastojati od četrdeset i tri broja odvojena zarezom. Prvi broj je indeks slova, iduća dva su x i y koordinata točke zapešća, što se, zatim, ponavlja za ostalih dvadeset glavnih točaka dlana. Također, treba provjeriti koliko je postojećih podataka za svako slovo, a taj će se korak onda ponavljati i prilikom izrade modela, budući da će neka slova imati manju vjerojatnost točnog prepoznavanja. Za analizu toga koristi se matrica zabune koja



Slika 4: Izgled baze

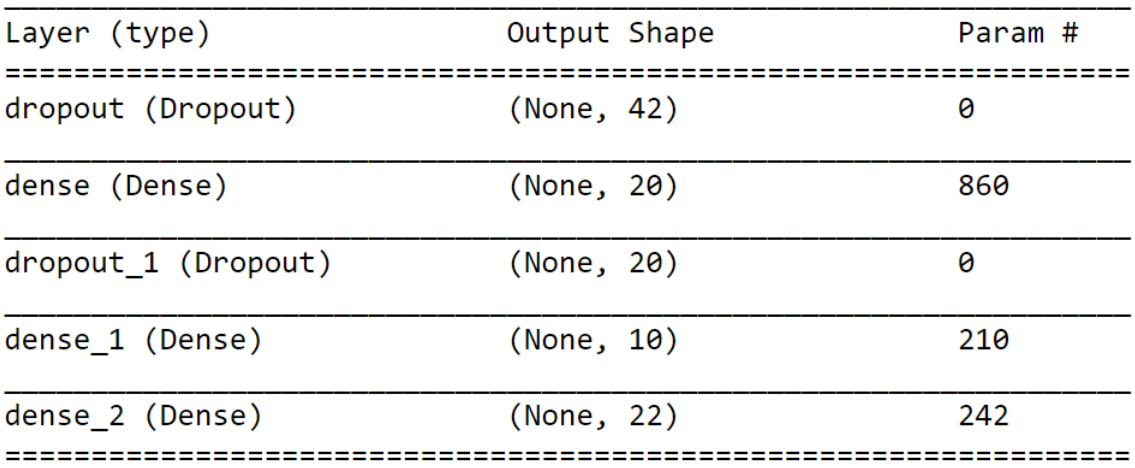
olakšava odluku kojem je slovu potrebno dodati još podataka.

3 Model neuronske mreže

Umjetna neuronska mreža skup je međusobno povezanih elemenata (neurona) koji obavljaju jednostavne funkcije [12]. Zbog odličnog rješavanja problema klasifikacije i predviđanja, neuronske mreže najčešće se koriste za zadatke poput: raspoznavanja uzoraka, obradu nekompletnih podataka, obradu slike i govora, probleme optimizacije i sl. Motivacija za korištenje neuronske mreže u implementaciji programa bio je veliki skup podataka koje je potrebno analizirati i na temelju toga zaključiti o kojem slovu je riječ, što bi bio izrazito težak zadatak pri izradi algoritamskog rješenja. Glavne prednosti mreže su svojstvo samostalnog učenja i mogućnost učenja iz nejasnih podataka, odnosno, podataka sa šumom. Nadalje, obrada i rezultat ne ovise previše o jednom elementu, što kao posljedicu ima implicito znanje mreže, odnosno, znanje koje je pohranjeno u mreži teško je interpretirati.
Cilj ovog poglavlja je opisati implemetaciju modela neuronske mreže koji iz ulaznog skupa podataka donosi odluku o kojem slovu je riječ. Ovakav način učenja naziva se nadziranim učenjem jer su ulazi i očekivani izlazi već poznati.

3.1 Implementacija

Dobiveni ulazni skup podataka potrebno je podijeliti u skup za učenje i skup za testiranje, za to se može koristiti metoda *train_test_split* biblioteke *Sklearn* [13]. Ostale korištene tehnologije koje je bitno istaknuti su *Tensorflow* [14] i *Keras* [15], koje služe za kreiranje i treniranje modela. Za početak je potrebno osmisliti arhitekturu mreže i neurona, odnosno rasporediti ih u slojeve. Za ovaj zadatak koristi se vrlo jednostavna arhitektura, koja se sastoji od četrdest i dva neurona u ulaznom sloju, što odgovara broju koordinata određenih za pojedino slovo, zatim dva sloja s dvadeset odnosno deset neurona te izlazni sloj s dvadeset i dva neurona, što odgovara broju slova čije koordinate se pohranjuju u bazu. Sažetak modela prikazan je slikom 5.



Slika 5: Sažetak modela

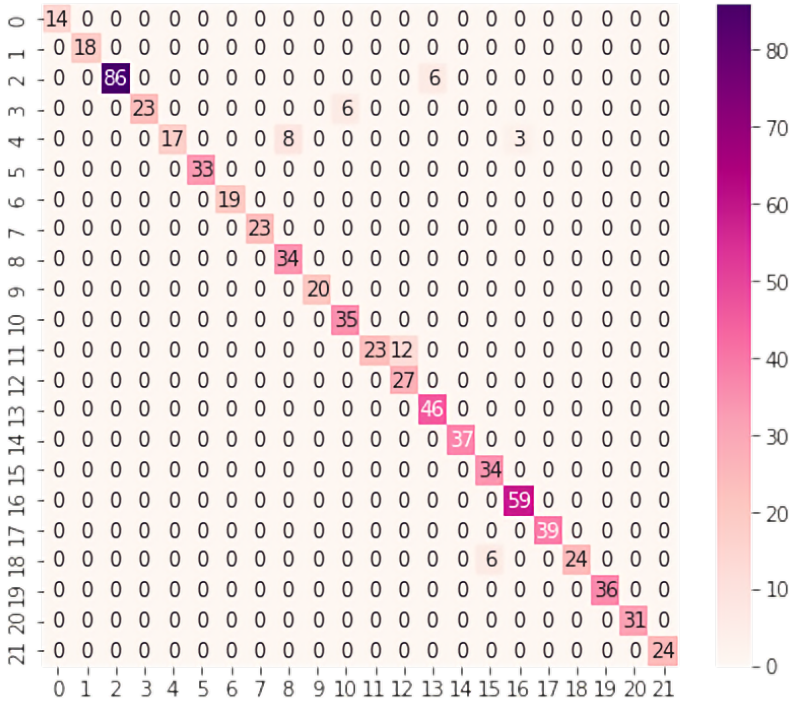
Metodom *compile* definiraju se optimizacijski algoritam, funkcija gubitka i mjerilo točnosti. Odabrani optimizacijski algoritam je *Adam* (eng. *Adaptive Moments*) [16], koji je ujedno i najčešće korišten algoritam u ovakvim zadacima. Uspješnost modela ovisi o vrijednostima stope učenja, a kako bi se maksimizirala uspješnost modela, stopa učenja treba biti prilagodljiva procesu učenja. Zbog toga, stopa u početku može biti veća kako bi ubrzala konvergenciju, ali ju kasnije treba smanjiti kako ne bi počela divergirati i unositi nestabilnosti u proces učenja. Optimizacijski algoritam *Adam* ubrzava učenje korištenjem tzv. momenata, kojima prati kretanje gradijenta i kvadrata gradijenta uz eksponencijalno zaboravljanje i zatim vrši korekciju parametara koristeći prosječne gradijente. Za funkciju gubitka odabrana je *sparse_categorical_crossentropy* [17] koja se koristi kad su izlazne vrijednosti modela cijeli brojevi, u slučaju ovog programa to bi bili indeksi slova čije koordinate su pohranjene u bazu, odnosno brojevi od nula do dvadeset i jedan. Točnost modela mjeri se vjerojatnošću točne klasifikacije, odnosno usporedbom dobivenog i očekivanog izlaza.

Metoda *fit* pokreće učenje mreže, kao argumenti prosljeđuju joj se uzorci za učenje, broj epoha, veličina grupe i skup za validaciju odnosno testiranje. Modeli zbog svo-

jih specifikacija i podataka s kojima rade, mogu zauzimati puno memorijskog prostora, zbog toga model na kraju možemo pretvoriti u *TensofFlow Lite* model. Za rad s takvim modelom potrebno je korsititi sučelje *TensorFlow Lite Interpreter* [18] čija uporaba je objašnjena u poglavlju 4.2. Klasifikator. Ukoliko model u formatu *.hdf5* ne zauzima puno prostora, ovaj korak može se preskočiti.

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

model.fit(
    XTrain,
    YTrain,
    epochs=1000,
    batch_size=128,
    validation_data=(XTest, YTest)
)
```

Na slici 6 prikazana je matrica zabune [19], koja slikovito prikazuje vjerojatnost točnog prepoznavanja pojedinog slova.



Slika 6: Matrica zabune

4 Prepoznavanje i klasifikacija

Zadnji korak u izradi aplikacije jest napraviti program za prepoznavanje znakova. U ovom poglavlju detaljno je opisan svaki zadatak i problem koji se pojavio pri izradi. Za početak, opisan je postupak pisanja programa i tehnologije koje su se koristile. Posebno je objašnjen i klasifikator koji vraća indeks prepoznatog slova. I na samom kraju, obrađeno je prepoznavanje znakova s dinamičkim pokretom, pritom analizirajući koordinate.

4.1 Rad programa

Program za prepoznavanje [20] sličan je programu za izradu baze. Tehnologije koje se upotrebljavaju su: *OpenCv*, za upravljanje kamerom i prikaz podataka na zaslonu, za rad s nizovima korištena je biblioteka *NumPy* te *MediaPipe*, za prepoznavanje dlana i glavnih točaka dlana iz svake ulazne slike kamere. Za početak, postavlja se kamera i veličina zaslona te postavke modela za prepoznavanje dlana i određivanje koordinata. U ovom programu omogućeno je prepoznavanje samo jedne ruke u isto vrijeme, a ta varijabla može se podešavati na način da se povećavanjem tog broja napravi i aplikacija za prepoznavanje dvoručne abecede hrvatskog znakovnog jezika. Također, postavljene su varijable *min_detection_confidence* na 0.7 i *min_tracking_confidence* na 0.5. One služe prilagođavanju modela *Mediapipe-a*, kako bi se osiguralo što točnije prepoznavanje ruke na zaslonu. Nakon toga, učitava se model za prepoznavanje ruke i određivanje koordinata te klasifikator za prepoznavanje slova. Ako model prepozna ruku, metodom *draw_landmarks* ocrtavaju se glavne točke dlana, a ta metoda preuzeta je sa službene stranice *MediaPipe Hands*. Zatim se pokreće *while* petlja, koja se zaustavlja pritiskom tipke "Esc", a do tad klasifikator pokušava odgonetnuti slovo prikazano statičnom gestom. Ako korisnik želi prepoznati slovo s dinamičkim pokretom, potrebno je pritisnuti tipku "d", pri čemu kreće odbrojavanje od tri sekunde nakon kojih se donosi odluka o

kojem slovu je riječ. Samo prepoznavanje odvija se tako da se konstantno zaprimaju ulazi kamere koji prolaze kroz model za prepoznavanje ruke i koordinata. Zatim se te koordinate spremaju u niz te se, kao i kod izrade baze, prvo ocrtava pravokutnik oko dlana i izračunavaju nove relativne i normirane koordinate koje se mogu predati klasifikatoru na prepoznavanje. Klasifikator vraća indeks prepoznatog slova, koji je zadan njegovim rednim brojem u *csv* datoteci s oznakama. Ako je program u načinu rada za prepoznavanje znakova s dinamičkim pokretom, taj će se isti proces odvijati i tri sekunde nakon odabira navedenog načina rada, tako da, na temelju početnih i konačnih koordinata pokreta, možemo vršiti analizu i odrediti kojem slovu odgovara određeni pokret ruke. Od velike je važnosti točno prepoznavanje znaka u početnom trenutku jer o tome ovisi hoće li se uopće znak s dinamičkim pokretom prepoznati pozivom funkcije *recognise_dynamic_gesture* koja je detaljno objašnjena u poglavlju 4.3.

4.2 Klasifikator

Klasifikator je program za učitavanje modela. Ovaj korak može se preskočiti ako se nije koristio *TensorFlow Lite Model*. U suprotnom potrebno je prilagoditi model, koristeći sučelje *Interpreter*. Prvo je potrebno učitati model, kojim se definira *Interpreter*. Zatim treba prilagoditi *tensors*, kojima se definiraju ulazi modela u obliku multidimenzionalnog polja, u slučaju ovog rada to je niz od četrdeset i dvije koordinate. Nakon pokretanja *Interpretera*, odnosno modela, dobivene izlaze potrebno je prilagoditi kako bi odgovarali predviđenom načinu rada, točnije, potrebno je prepoznati indeks prepoznatog slova.

4.3 Prepoznavanje i analiza slova s dinamičkim pokretom

Prepoznavanje znakova s dinamičkim pokretom najzahtjevniji je dio rada upravo jer se tim korakom spaja strojno učenje s algoritamskim rješenjem. Za određivanje o kojem slovu je riječ potrebna su dva skupa koordinata - prvi je skup u početnom trenutku, a drugi je skup u trenutku nakon što završi odbrojavanje od tri sekunde. Nakon toga, skupovi koordinata položaja ruke mogu se uspoređivati, a na temelju njihove promjene zaključuje se slovo prikazano pokretom. Cilj je pronaći najveće razlike i ponavljajuće pravilnosti u podacima, što se postiže analizom koordinata. Kod ovakvog uspoređivanja, potrebno je razumjeti kako se radi s dva različita koordinatna sustava - dlan u početnom trenutku ima različiti opisani pravokutnik od onoga u krajnjem trenutku, stoga nije riječ o pravim udaljenostima između koordinata, već taj pojam olakšava prikaz intuitivno jasnog odnosa, to jest, odnosa između koordinata koji je vidljiv i pri samom izvođenju pokreta. U idućim tablicama prikazane su koordinate za svako slovo u početnom i konačnom trenutku te dio koda koji prepoznaje slovo s dinamičkim pokretom na temelju tih podataka. U isječcima koda, *if* petlje započinju provjerom *hand_sign_id*-a, koji određuje indeks slova pohranjenog u bazi, a *landmark_list* služi za pohranu x i y koordinata dvadeset jedne ključne točke dlana, stoga je dimenzija ovog niza četrdeset i dva.

Slova Č i Ć

Uvodnom analizom i odabirom slova za bazu, zaključeno je da su slova 'Č' i 'Ć' najsličnija slovu 'C' u početnom trenutku, stoga se i prepoznaju iz slova 'C'. Nadalje, pokret ruke za slovo 'Č' većinski se ne pomiče u y smjeru, dok za slovo 'Ć' vrijedi suprotno. Stoga je za slovo 'Č' prikazana tablica s x koordinatama u početnom i krajnjem trenutku za četiri primjera, a analogno tomu prikazana je i tablica za y koordinate koje se mijenjaju pri prikazu geste za slovo 'Ć'. Jasno je da se, prilikom izvođenja pokreta za slovo 'Č', x koordinate primiču vrijednosti nula. Radi lakše vizualizacije, zamislimo kako

izgleda slovo ’Č’ u početnom trenutku i promatrajmo položaj točke 12 koja je određena kao vrh srednjeg prsta po *Hand Landmark Modelu* te položaj točke 4. Te su točke najviše udaljene od točke zapešća u kojoj je vrijednost x koordinate nula. Pomakom ruke u krajnji položaj prilikom izvođenja pokreta za slovo ’Č’, uočava se kako su te točke bliže zapešću. Međutim, navedene točke mogu poslužiti samo kao smjernice za shvaćanje kako se koordinate mijenjaju, no nisu dobri kandidati za usporedbu, jer su to točke koje su na samom rubu opisanog pravokutnika, stoga će njihova vrijednost uvijek biti blizu vrijednosti minus jedan. Za određivanje uočenih pravilnosti, potrebno je proučiti koordinate iz tablice 1. Prikazano je kako se, od svih koordinata, najviše mijenja vrijednost x5, tako da je razlika vrijednosti između t0 i t3 u svim primjerima sigurno veća od 0.2. Ta se vrijednost može mijenjati i ovisno o tome tko ju izvodi, pa vrijednosti u početnom trenutku neće biti iste. Iz tog razloga, granice navedene vrijednosti nisu strogo određene jer, prepoznavanjem slova ’C’ u početnom trenutku, prepoznata gesta bit će klasificirana kao ’Č’ ili ’Ć’ ili, u najgorem mogućem slučaju, kao "ništa". To je prikazano i u isječku koda.

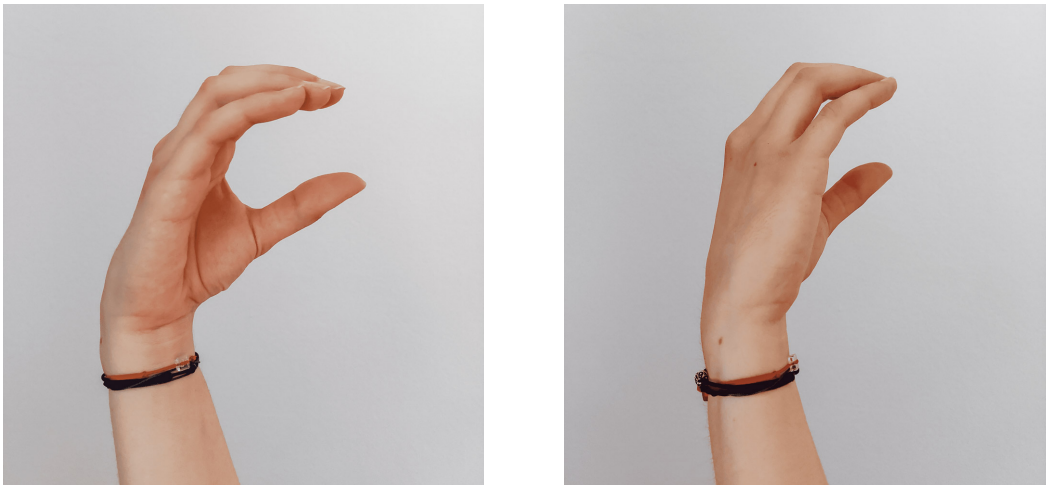
Tablica 1: X koordinate slova ’č’

koordinata	t_0	t_3	t_0	t_3	t_0	t_3	t_0	t_3
x0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
x1	-0.33	-0.3	-0.29	-0.28	-0.26	-0.28	-0.3	-0.17
x2	-0.59	-0.46	-0.55	-0.47	-0.5	-0.47	-0.52	-0.28
x3	-0.79	-0.61	-0.77	-0.63	-0.72	-0.63	-0.71	-0.36
x4	-0.99	-0.75	-0.98	-0.78	-0.94	-0.78	-0.91	-0.46
x5	-0.54	-0.22	-0.54	-0.29	-0.44	-0.29	-0.45	-0.16
x6	-0.7	-0.41	-0.7	-0.47	-0.61	-0.47	-0.61	-0.27
x7	-0.85	-0.57	-0.84	-0.62	-0.75	-0.62	-0.75	-0.38
x8	-1.0	-0.7	-0.97	-0.73	-0.9	-0.73	-0.88	-0.46
x9	-0.46	-0.22	-0.46	-0.28	-0.36	-0.28	-0.39	-0.16
x10	-0.65	-0.45	-0.65	-0.5	-0.52	0.5	-0.57	-0.29
x11	-0.83	-0.63	-0.8	-0.65	-0.69	-0.65	-0.73	-0.4
x12	-0.99	-0.73	-0.96	-0.74	-0.88	-0.74	-0.88	-0.48
x13	-0.37	-0.27	-0.35	-0.33	-0.27	-0.33	-0.31	-0.2
x14	-0.55	-0.51	-0.55	-0.56	-0.41	-0.56	-0.48	-0.33
x15	-0.74	-0.7	-0.71	-0.72	-0.57	-0.72	-0.65	-0.45
x16	-0.91	-0.83	-0.87	-0.82	-0.74	-0.82	-0.81	-0.54
x17	-0.28	-0.38	-0.23	-0.43	-0.18	-0.43	-0.25	-0.27
x18	-0.44	-0.59	-0.4	-0.64	-0.33	-0.64	-0.39	-0.41
x19	-0.59	-0.72	-0.54	-0.76	-0.47	-0.76	-0.53	-0.49
x20	-0.74	-0.82	-0.69	-0.86	-0.62	-0.86	-0.67	-0.57

Prepoznavanje slova 'Č' implementirano je usporedbom položaja točke 5 u početnom i krajnjem trenutku odbrojavanja od tri sekunde. Kao što je prikazano slikom 7. Te koordinate smještene su u *landmark_list* i *landmark_list_b* s indeksom deset. Budući da su sve koordinate negativne, jer se nalaze lijevo od zapešća, može se računati i s apsolutnim vrijednostima tih koordinata, bitno je samo da razlika položaja bude veća od 0.2.

```
    #prepoznavanje č i ć iz slova c s indeksom 2
    if hand_sign_id == 2:

        x1 = landmark_list[10]
        x2 = landmark_list_b[10]
        y1 = landmark_list[23]
        y11 = landmark_list[9]
        y2 = landmark_list_b[23]
        y22 = landmark_list_b[9]
        dist1 = abs(y1 - y11)
        dist2 = abs(y2 - y22)

        if dist1 - dist2 > 0.35:
            return "ć"
        elif abs(x1) - abs(x2) > 0.20:
            return "č"
        else:
            return "ništa"
)
```

U tablici 2 prikazane su y koordinate u početnom trenutku i tri sekunde nakon za slovo ’Ć’. Promatrajući taj pokret, prikazan slikom 8, uočava se kako se svi prsti približavaju palcu, pa je moguće proizvoljno odabrati, primjerice srednji prst, koji je određen točkama 9, 10, 11, i 12 po slici 1. Za uspoređivanje uzimaju se točke koje ne mogu određivati opisani pravokutnik, kao što je ovom slučaju točka 12, već, primjerice, točka 11. Iz tablice se zaključuje kako se vrijednost y koordinate mijenja za približno 0.4 u intervalu od tri sekunde. Međutim, ovo je primjer zadatka u kojem se, ipak, uspoređuje udaljenost između točaka 4 i 11 u početnom i krajnjem trenutku, jer je to smanjivanje udaljenosti upravo ono što definira pokret ruke za slovo ’Ć’, odnosno tako se opisuje približavanje svih prstiju palcu. U programskom kodu, to je implementirano tako da se dohvati vrijednost y koordinata točaka 4 i 11 u početnom trenutku. Te koordinate pohranjene su u *landmark_list* s indeksima devet odnosno dvadeset i tri. Isto to radi se i za trenutak nakon tri sekunde. Zatim se računa udaljenost u početnom i krajnjem trenutku te se postavlja uvjet kojim treba vrijediti da je udaljenost u početnom trenutku veća od one nakon tri sekunde.



(a) Početni trenutak (b) Trenutak nakon tri sekunde

Slika 7: Pokret slova ’č’

Tablica 2: Y koordinate slova ’ć’

koordinata	t_0	t_3	t_0	t_3	t_0	t_3	t_0	t_3
y0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
y1	-0.08	-0.12	-0.07	-0.08	-0.05	-0.07	-0.06	-0.09
y2	-0.27	-0.26	-0.21	-0.19	-0.2	-0.18	-0.22	-0.21
y3	-0.32	-0.32	-0.23	-0.22	-0.21	-0.21	-0.26	-0.25
y4	-0.33	-0.32	-0.2	-0.22	-0.19	-0.2	-0.24	-0.27
y5	-0.66	-0.62	-0.65	-0.53	-0.62	-0.53	-0.62	-0.57
y6	-0.88	-0.6	-0.88	-0.5	-0.85	-0.52	-0.84	-0.54
y7	-0.95	-0.56	-0.95	-0.46	-0.94	-0.48	-0.94	-0.48
y8	-0.96	-0.54	-0.98	-0.41	-0.98	-0.43	-0.99	-0.43
y9	-0.67	-0.64	-0.66	-0.55	-0.64	-0.55	-0.65	-0.59
y10	-0.93	-0.63	-0.93	-0.54	-0.91	-0.56	-0.91	-0.56
y11	-1.0	-0.56	-0.99	-0.46	-0.99	-0.49	-0.99	-0.47
y12	-0.99	-0.5	-0.99	-0.38	-1.0	-0.41	-1.0	-0.4
y13	-0.65	-0.61	-0.61	-0.51	-0.6	-0.52	-0.62	-0.55
y14	-0.91	-0.6	-0.88	-0.52	-0.87	-0.54	-0.88	-0.54
y15	-0.98	-0.55	-0.95	-0.46	-0.95	-0.49	-0.96	-0.47
y16	-0.99	-0.51	-0.98	-0.4	-0.99	-0.44	-0.99	-0.42
y17	-0.57	-0.51	-0.53	-0.41	-0.51	-0.41	-0.54	-0.44
y18	-0.75	-0.51	-0.7	-0.41	-0.69	-0.43	-0.71	-0.42
y19	-0.84	-0.51	-0.81	-0.41	-0.79	-0.43	-0.8	-0.41
y20	-0.9	-0.51	-0.89	-0.4	-0.87	-0.42	-0.85	-0.4

(a) Početni trenutak (b) Trenutak nakon tri sekunde

Slika 8: Pokret slova ’ć’

Slovo Đ

Promatrajući pokret za slovo ’Đ’, prikazanog slikom 9, uočava se kako je u početnom trenutku identično slovu ’D’, stoga će se iz njega i prepoznavati. Odnosno, očekivano jest da se, u početnom trenutku, treba prepoznati slovo ’D’ s indeksom tri u datoteci s oznakama slova, a zatim će se, zbog prirode pokreta, uspoređivati odnosi koordinata. Ponovno se nameće pitanje koje točke odabrati za usporedbu. Ono što se isprva vidi jest da se pokretom minimalno mijenjaju y koordinate, stoga se, u ovom primjeru, ne razmatraju. Tijekom odabira točaka, ne razmatraju se ni točke koje se nalaze na bridovima opisanog pravokutnika, međutim primijetno je kako se položaj točke 4 i sve točke srednjeg prsta najviše pomiču od početnog položaja. Za implementaciju uzimaju se točke 4 i 12 sa slike 1 te se uspoređuju njihovi pomaci. Radi preglednosti, u tablici 3 prikazane su koordinate u početnom trenutku i krajnjem trenutku samo za te dvije točke.

Tablica 3: X koordinate slova ’d’

koordinata	t_0	t_3	t_0	t_3	t_0	t_3	t_0	t_3
x4	-0.14	-0.45	-0.14	-0.43	-0.15	-0.43	-0.09	-0.51
x12	-0.09	-0.48	-0.12	-0.5	-0.11	-0.49	-0.04	-0.53

X koordinate točaka 4 i 12 zauzimaju mjesta s indeksima osam i dvadeset i četiri u *landmark_list*. Iz tablice se primjećuje kako su te vrijednosti uvijek negativne, što je i za pretpostaviti, budući da se nalaze lijevo od zapešća prikazanog na zaslonu, odnosno, ishodišta. Shodno tomu, može se računati s njihovim apsolutnim vrijednostima. Zatim se postavljaju uvjeti, kojima razlika vrijednosti x koordinate točke 4 u početnom trenutku i trenutku nakon tri sekunde mora biti veća od 0.25 ili razlika vrijednosti x koordinate točke 12 mora biti veća od 0.35. Implementacija je prikazana u nastavku s isječkom programskog koda:

```
    #prepoznavanje đ iz slova d s indeksom 3
    if hand_sign_id == 3:
        x1 = landmark_list[8]
        x2 = landmark_list_b[8]
        x11 = landmark_list[24]
        x22 = list_b[24]
        if abs(x2) - abs(x1) > 0.25 or abs(x22) - abs(x11) > 0.35:
            return "đ"
        else:
            return "ništa"
```

(a) Početni trenutak (b) Trenutak nakon tri sekunde

Slika 9: Pokret slova ’đ’

Slovo DŽ

Prepoznavanje slova ’DŽ’ zanimljivo je jer se prepoznaje iz slova ’U’. Razlog tomu su položaji kažiprsta i srednjeg prsta, koji su prikazani slikom 10, a sama analiza i uspoređivanje koordinata dosta su slični analizi slova ’Đ’. Uspoređuju se x koordinate točaka sa slike 1 jer po y osi nema velikih pomaka, te je potrebno izbjegavati rubne točke koje služe za definiranje opisanog pravokutnika dlana. Odabir točaka za uspoređivanje, u navedenom primjeru, nije banalan, stoga se koristi tablica 4. Popis koordinata koji je korišten pri implementaciji je, naravno, puno veći od navedene tablice, ali zbog lakšeg razumijevanja, ovdje su prikazane samo vrijednosti x koordinata točaka koje imaju najveći pomak u odnosu na početni trenutak. Posebno su zanimljive točke kojima x koordinata prelazi iz pozitivne u negativnu vrijednost, što vrijedi za sve navedene koordinate točaka 18, 19 i 20. Za implementaciju odabiru se dvije, npr. točke 19 i 20, jer imaju najveću razliku vrijednosti u početnom trenutku i trenutku nakon tri sekunde. Uvjetne vrijednosti određene su analizom koordinata tih točaka, a vidi se da vrijede i za podatke iz tablice 4. U nastavku je prikazan i programski kod za prepoznavanje slova

’DŽ’.

Tablica 4: X koordinate slova ’dž’

koordinata	t_0	t_3	t_0	t_3	t_0	t_3	t_0	t_3
x3	-0.1	-0.27	-0.05	-0.27	-0.05	-0.29	-0.06	-0.3
x4	-0.01	-0.33	0.06	0.35	0.05	-0.37	0.04	-0.38
x18	0.22	-0.12	0.2	-0.15	0.2	-0.19	0.19	-0.17
x19	0.16	-0.23	0.18	-0.26	0.18	-0.3	0.16	-0.28
x20	0.1	-0.32	0.16	-0.35	0.16	-0.38	0.12	-0.37

```
    #prepoznavanje dž iz slova u s indeksom 18
    if hand_sign_id == 18:
        x1 = landmark_list[38]
        x2 = landmark_list_b[38]
        x11 = landmark_list[40]
        x22 = landmark_list_b[40]

        if abs(x2) + abs(x1) > 0.30 or abs(x22) + abs(x11) > 0.40 :
            return "dž"
        else:
            return "ništa"
```

(a) Početni trenutak (b) Trenutak nakon tri sekunde

Slika 10: Pokret slova ’dž’

Slovo J

Slovo ’J’ u početnom trenutku pokreta identično je slovu ’I’, međutim, u trenutku nakon tri sekunde, mali prst trebao bi se pomaknuti iz gornjeg lijevog kuta pravokutnika u donji lijevi kut, kao što je prikazano na slici 11. Nadalje, pravokutnici su u različitim trenutcima također različiti, tako da je potrebno analizirati koordinate u početnom i krajnjem trenutku te uočiti pravilnosti koje se ponavljaju pri svakom izvođenju pokreta za slovo ’J’. Na opširnijem skupu podataka uočeno je da se velike promjene događaju kod x koordinata i y koordinata, stoga iduća tablica sadrži vrijednosti za x i y koordinate, koje su imale najveći pomak pri testiranju.

Tablica 5: X i Y koordinate slova 'j'

koordinata	t_0	t_3	t_0	t_3	t_0	t_3	t_0	t_3
x4	-0.02	-0.67	-0.1	-0.86	-0.09	-0.71	-0.16	-0.68
y4	-0.64	-0.3	-0.67	-0.06	-0.67	-0.19	-0.62	-0.2
x6	-0.31	-0.73	-0.34	-0.86	-0.34	-0.81	-0.42	-0.94
x7	-0.26	-0.9	-0.3	-0.97	-0.3	-0.96	-0.37	-1.0
x10	-0.14	-0.69	-0.19	-0.81	-0.17	-0.81	-0.26	-0.93
x11	-0.13	-0.88	-0.18	-0.93	-0.15	-0.94	-0.23	-0.95
x20	0.15	-0.64	0.1	-0.88	0.13	-0.73	0.06	-0.75
y20	-1.0	0.03	-1.0	0.08	-1.0	0.18	-1.0	0.2

Za implementaciju odabiru se dvije koordinate, a uočljivo jest kako se najveći pomaci realiziraju u točki 20, koja je ujedno i vrh malog prsta. Uvjeti se definiraju po volji, pazeći pritom na odnose koordinata u različitim trenutcima npr. postavljanje uvjeta, u kojima se traži da zbroj apsolutnih vrijednosti y koordinate točke 20 u početnom i krajnjem trenutku bude veći od 1 ili da zbroj apsolutnih vrijednosti x koordinate točke 20 u početnom trenutku i trenutku nakon tri sekunde bude veći od 0.7 kao što je napisano u ponuđenom programskom kodu:

```
1     #prepoznavanje j iz slova i s indeksom 8
2     if hand_sign_id == 8:
3
4         y1 = landmark_list[41]
5         y2 = landmark_list_b[41]
6         x1 = landmark_list[40]
7         x2 = landmark_list_b[40]
```

```
8           if abs(y1) + abs(y2) > 1 or abs(x1) + abs(x2) > 0.7 :
9               return "j"
10          else:
11              return "ništa"
```

(a) Početni trenutak (b) Trenutak nakon tri sekunde

Slika 11: Pokret slova 'j'

Slovo LJ

Kod prepoznavanja slova 'LJ', postoji problem koji uzrokuju opisani pravokutnici na početku snimanja i nakon tri sekunde. Rotacijom ruke iz položaja slova 'L', za maksimalno devedeset stupnjeva unazad dobije se pokret slova 'LJ'. Kao što se može vidjeti na slici 12. Međutim, rubne točke oba pravokutnika ostaju iste, stoga su pomaci mali za većinu x koordinata. Tako da se uvjeti, u ovom slučaju, ne definiraju samo najvećim razlikama u odnosima x koordinata početnog i krajnjeg trenutka, već i najmanjim pomacima. Takvi uvjeti povezuju se konjunkcijom jer je bitno da su svi zadovoljeni, za

razliku od prethodno objašnjenih slova s dinamičnim pokretom. Ako se pozorno prouči pokret, jasno je kako se točke najmanje pomiču oko zamišljene osi rotacije smještene negdje između kažiprsta i srednjeg prsta, stoga se, od x koordinata tih točaka, zahtijevaju najmanji pomaci. U tablici 6, prikazane su x koordinate za točke 4, 9, 10 i 18, predočene na slici 1, u trenutku početka odbrojavanja i tri sekunde nakon za četiri slučaja. Vidi se da se vrijednosti x koordinate točaka 4 i 18 pomiču puno više u odnosu na x koordinate točaka 9 i 10.

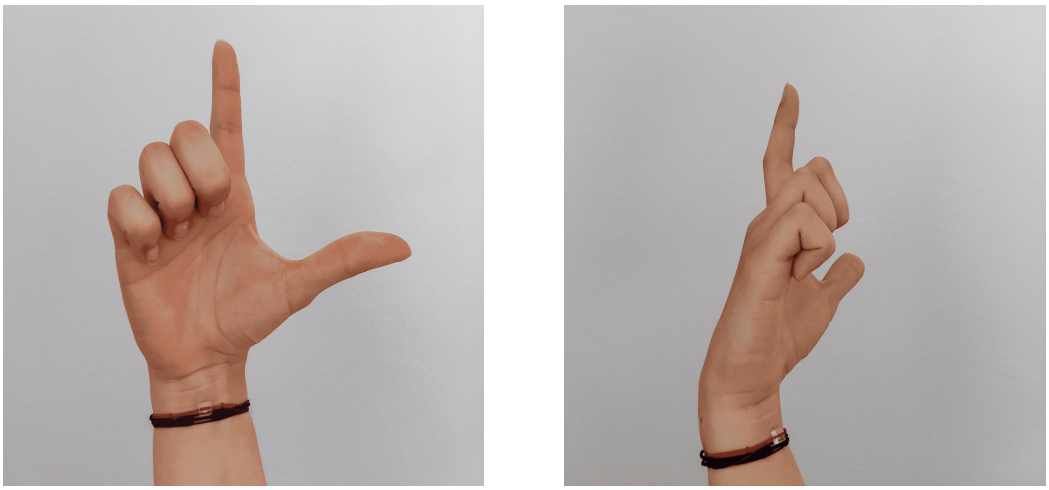
Tablica 6: X koordinate slova 'lj'

koordinata	t_0	t_3	t_0	t_3	t_0	t_3	t_0	t_3
x4	-0.59	-0.2	-0.66	-0.29	-0.62	-0.29	-0.62	-0.35
x9	-0.02	0.04	-0.14	-0.02	-0.09	-0.01	-0.07	0.02
x10	-0.06	-0.21	-0.15	-0.29	-0.1	-0.28	-0.08	-0.24
x18	0.19	-0.2	0.14	-0.25	0.18	-0.27	0.19	-0.2

Za implementaciju odabrane su točke 9 i 18, odnosno njihove x koordinate. Zahtijeva se minimalni pomak za točku 9 i maksimalni pomak za točku 18. Sve koordinate pohranjene su u *landmark_list*, s tim da su x koordinate na parnim mjestima u nizu, a y koordinate na neparnim. Vrijednosti x koordinate točaka 9 i 18, nalaze se na mjestu s indeksom osamnaest, odnosno, trideset i šest u *landmark_list*.

```
1     #prepoznavanje lj iz slova l s indeksom 10
2     if hand_sign_id == 10:
3
4         x1 = landmark_list[36]
5         x2 = landmark_list_b[36]
```

```
6           x11 = landmark_list[18]
7           x22 = landmark_list_b[18]
8           if abs(x1 - x2) > 0.35 and abs(x11 - x22) < 0.15:
9               return "lj"
10          else:
11              return "ništa"
```



(a) Početni trenutak (b) Trenutak nakon tri sekunde

Slika 12: Pokret slova ’lj’

Slovo NJ

Slovo ’NJ’ prepoznaje se iz geste za slovo ’N’. Promatrajući sliku 13, uočava se kako se pomaci, uglavnom, mogu zapaziti samo u smjeru x osi pa će se, u ovoj analizi, zanemariti y koordinate točaka sa slike 1. Uočava se pomak kažiprsta i srednjeg prsta, stoga se analiziraju točke 8 i 12, koje su ujedno i vrhovi navedenih prstiju. X koordinate tih točaka navedene su u tablici 7 u početnom trenutku i trenutku nakon tri sekunde od

početnog. Na temelju tih podataka, mogu se postaviti uvjeti potrebni za implementaciju prepoznavanja slova 'NJ'. Vidi se kako se razlika x koordinata točke 8 u sva četiri slučaja mijenja za više od 0.4, analogno tome promatraju se pomaci x koordinata točke 12 i zaključuje se kako je vrijednost pomaka svugdje veća od 0.5.

Tablica 7: X koordinate slova 'nj'

koordinata	t_0	t_3	t_0	t_3	t_0	t_3	t_0	t_3
x8	-0.07	0.39	-0.05	-0.41	-0.04	0.47	-0.05	0.4
x12	0.03	0.54	0.03	0.59	0.07	0.61	0.05	0.57

Implementacija prethodno objašnjenog postupka izgleda ovako:

```
#prepoznavanje nj iz slova n s indeksom 12
if hand_sign_id == 12:

    x1 = landmark_list[16]
    x2 = landmark_list_b[16]
    x11 = landmark_list[24]
    x22 = landmark_list_b[24]
    if abs(x2 - x1) > 0.4 or abs(x11 - x22) > 0.5:
        return "nj"
    else:
        return "ništa"
```

(a) Početni trenutak

(b) Trenutak nakon tri sekunde

Slika 13: Pokret slova ’nj’

Slovo Š

Slika 3 prikazuje hrvatsku jednoručnu abecedu znakovnog jezika, za pokret slova ’Š’ primjećuje se da nastaje rotacijom geste slova ’S’ ulijevo, što je prikazano slikom 14. Stoga će se, i u ovom primjeru, kao i za slovo ’LJ’ promatrati točke s minimalnim i maksimalnim pomakom. Analizom velikog skupa koordinata, zamijećeno je kako najveće pomake ostvaruju točke 3, 4, 18 i 19, a najmanji pomak imaju točke najbliže zapešću. Zbog preglednosti, u ovom primjeru, za to je odabrana točka 2. U tablici 8, prikazane su x koordinate navedenih točaka u početnom trenutku i trenutku nakon tri sekunde. Za postavljanje uvjeta nisu potrebne sve točke, a u konkretnom primjeru korištene su točke 2, 18 i 19.

Tablica 8: X koordinate slova ’š’

koordinata	t_0	t_3	t_0	t_3	t_0	t_3	t_0	t_3
x2	-0.27	-0.3	-0.26	-0.34	-0.28	-0.37	-0.27	-0.34
x3	-0.16	-0.37	-0.12	-0.4	-0.14	-0.43	-0.13	-0.39
x4	0.05	-0.38	0.08	-0.39	0.05	-0.42	0.06	-0.36
x18	0.28	-0.18	0.27	-0.19	0.25	-0.19	0.22	-0.1
x19	0.2	-0.28	0.22	-0.29	0.2	-0.3	0.18	-0.24

Idući isječak koda prikazuje implementaciju navedenog algoritma za određivanje uvjeta. Iz liste *landmark_list* preuzete su x koordinate točaka 2, 18 i 19 te su određene razlike njihovog pomaka i postavljeni uvjeti za te vrijednosti.

```
    #prepoznavanje š iz slova s s indeksom 16
    if hand_sign_id == 16:

        x12 = landmark_list[4]
        x22 = landmark_list_b[4]
        x18 = landmark_list[36]
        x218 = landmark_list_b[36]
        x19 = landmark_list[38]
        x219 = landmark_list_b[38]
        if abs(x12 - x22) < 0.15 and
            (abs(x18 - x218) > 0.3 or abs(x19 - x219) > 0.3):
            return "š"
        else:
            return "ništa"
```

(a) Početni trenutak

(b) Trenutak nakon tri sekunde

Slika 14: Pokret slova ’š’

Slovo Z

U početnom trenutku pokreta prikazanog slikom 15, slovo ’Z’ nije nalikovalo ni jednom drugom znaku, stoga ga je bilo potrebno dodati u bazu. Slova ’Z’ i ’Ž’, zbog slobode pokreta i različitosti opisanih pravokutnika u svakom trenutku, najzahtjevnija su za analizirati i uočiti pravilnosti. Međutim, y koordinate točaka 3 i 4 te x koordinate točke 8, imaju svojstvo prelaska iz negativnih vrijednosti u pozitivne, stoga su one odabrane za implementaciju prepoznavanja slova ’Z’.

Tablica 9: X i Y koordinate slova ’z’

koordinata	t_0	t_3	t_0	t_3	t_0	t_3	t_0	t_3
y3	-0.37	0.28	-0.38	0.28	-0.39	0.28	-0.43	0.28
y4	-0.53	0.25	-0.54	0.25	-0.55	0.25	-0.57	0.25
x8	-0.53	0.71	-0.52	0.69	-0.49	0.72	-0.41	0.71

Prikazani isječak koda opisuje postupak određivanje uvjeta za prepoznavanje pokreta slova 'Z'. Za pomake y koordinata točki 3 i 4, postavljeni su uvjeti kojim apsolutna vrijednost pomaka mora biti veća od 0.6, odnosno 0.7. Razlika x koordinata točke 8, u početnom trenutku i trenutku nakon odbrojavanja, ovisi o načinu izvođenja pokreta. Međutim, u svim slučajevima, x koordinate točke 8, prelaze iz negativnih vrijednosti u pozitivne. Vizualno, to bi značilo, da se vrh kažipsta u početnom trenutku nalazi s lijeve strane zapešća, tj. ishodišta, a u krajnjem trenutku, desno od njega.

```
    if hand_sign_id == 20:

        y7 = landmark_list[7]
        y9 = landmark_list[9]
        y7b = landmark_list_b[7]
        y9b = landmark_list_b[9]
        x16 = landmark_list[16]
        x16b = landmark_list_b[16]
        if (abs(y7b - y7) > 0.6 or abs(y9b - y9) > 0.7) or (x16b > x16):
            return "z"
        else:
            return "ništa"
```

(a) Početni trenutak

(b) Trenutak nakon tri sekunde

Slika 15: Pokret slova ’z’

Slovo Ž

Analiza koordinata i implementacija prepoznavanja slova ’Ž’, slične su istim procesima prilikom prepoznavanju slova ’Z’. Slovo ’Ž’, zbog uvodne analize, također je dodano u bazu. Stoga se određivanje svodi na prepoznavanje početnog položaja slova ’Ž’ i uspoređivanje tih koordinata s koordinatama u trenutku nakon tri sekunde. Kao što je prikazano slikom 16. U tablici 10 prikazane su x koordinate točaka 6 i 7, te y koordinate točaka 7 i 8, iz čega se može zaključiti kako najveće pomake ostvaruju točke kažiprsta.

Tablica 10: X i Y koordinate slova ’ž’

koordinata	t_0	t_3	t_0	t_3	t_0	t_3	t_0	t_3
x6	-0.39	0.07	-0.35	0.07	-0.37	0.06	-0.38	0.07
x7	-0.43	0.37	-0.39	0.37	-0.40	0.37	-0.42	0.38
y7	-0.80	-0.08	-0.79	-0.05	-0.79	-0.05	-0.80	-0.05
y8	-0.91	0.02	-0.89	0.03	-0.91	0.03	-0.92	0.02

Isječkom koda prikazan je postupak uspoređivanja koordinata i postavljanja uvjeta nad njihovim pomacima. Slovo 'Ž' dvadeset i drugo je slovo pohranjeno u bazi, tj. *csv* datoteci s oznakama, zbog toga je određeno indeksom dvadeset i jedan. Analizom je ustanovljeno kako najveće pomake ostvaruju x koordinate točke 7, odnosno y koordinate točke 8, zbog čega su nad njihovim pomacima postavljeni uvjeti.

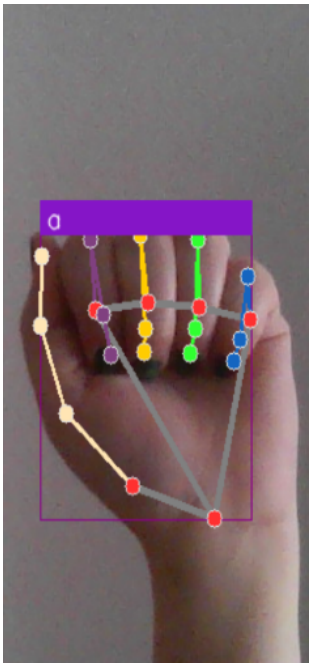
```
    if hand_sign_id == 21:

        x7 = landmark_list[14]
        x7b = landmark_list_b[14]
        y8 = landmark_list[17]
        y8b = landmark_list_b[17]
        if abs(y8b - y8) > 0.75 and abs(x7b - x7) > 0.6:
            return "ž"
        else:
            return "ništa"
```

(a) Početni trenutak (b) Trenutak nakon tri sekunde

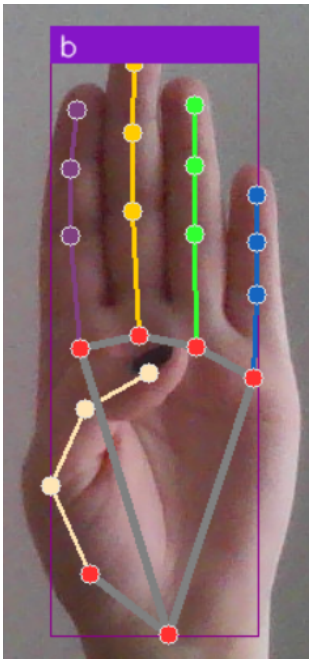
Slika 16: Pokret slova ’ž’

4.4 Primjer korištenja programa

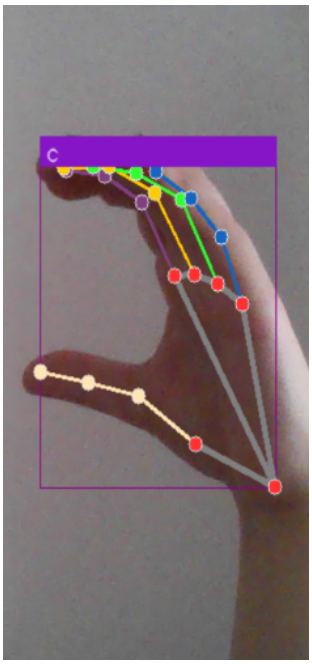
Korištenje programa za prepoznavanje veoma je intuitivno, barem za slova sa statičnim pokretom. Korisnik treba ispred kamere prikazati rukom željenu gestu, a prepoznato slovo prikazat će se u gornjem lijevom kutu pravokutnika. Za prepoznavanje slova s dinamičkim pokretom, od korisnika se očekuje da nakon pritiska tipke "d" napravi pokret određenog slova u intervalu od tri sekunde. Nakon toga, prepoznato slovo ispisuje se na standardni izlaz, a razlog tomu je funkcija *putText* biblioteke *OpenCv* koja podržava samo ASCII znakove, zbog čega se dijakritička slova ne mogu ispisati na zaslon. Na slici 17 prikazano je prepoznavanje slova: ’a’, ’b’, ’c’ i ’d’.



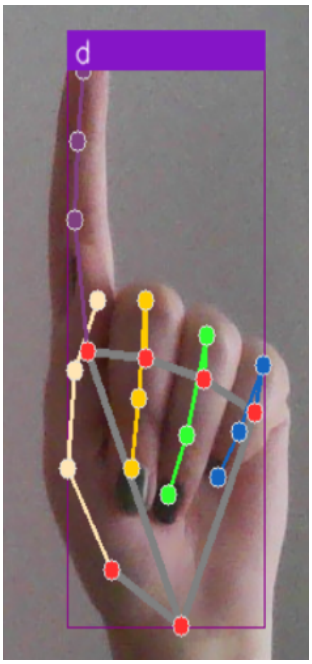
(a) Slovo ’a’



(b) Slovo ’b’



(c) Slovo ’c’



(d) Slovo ’d’

Slika 17: Prepoznavanje

5 Zaključak i daljnji rad

Ovim radom pokazano je da postoji dosta prostora za napredak pri izradi opcija pristupačnosti za osobe s oštećenjima sluha. Iako je ideja aplikacije prilično jednostavna, proces razvoja bio je ipak malo složeniji. Za početak, trebalo je odabrati koje tehnologije koristiti, budući da je cilj bio prepoznavati slova, a adekvatno rješenje za to bila je neuronska mreža. Međutim, priprema podataka zahtijevala je detaljno proučavanje rada radnog okvira *MediaPipe Hands*, kako bi se ustanovila prikladnost tog rješenja s idejom izrade. Nakon toga uslijedila je izrada baze i modela neuronske mreže, te je time završen preduvjetni dio za implementaciju programa za prepoznavanje slova. U ovom radu detaljno je objašnjen i postupak prepoznavanja slova s dinamičkim pokretom, koji je veoma bitan zbog toga što većinu tih znakova čine dijakritička slova, svojstvena hrvatskom jeziku. Za njihovo prepoznavanje bilo je potrebno uspoređivati koordinate pokreta u različitim trenucima i na temelju tih odnosa zaključiti o kojem slovu je riječ. Konačno rješenje određeno je prepoznatim slovom sa statičnom gestom u početnom trenutku i položajem ruke u trenutku nakon tri sekunde, čime se kombinira pristup algoritamskog rješavanja i rješavanja pomoću metoda strojnog učenja. Jednoručna abeceda samo je mali podskup čitavog znakovnog jezika, ali ova ideja mogla bi se primijeniti i na probleme prepoznavanja dvoručne abecede hrvatskog jezika, pa čak i na prepoznavanje jednostavnijih izraza znakovnog jezika. Primjerice, za izradu aplikacije prepoznavanja dvoručne abecede, trebalo bi promijeniti podatke u bazi, međutim, opet je dovoljno pohranjivati koordinate samo jedne ruke, a dodatna analiza provela bi se nad odnosima koordinata prepoznate ruke i druge ruke prikazane na zaslonu. Primjena ovakvih rješenja može poslužiti u edukaciji, čime bi se od ranog djetinjstva smanjivala diskriminacija uzrokovana jezičnom barijerom. Osim u edukaciji, unaprjeđenja ovakvih rješenja, mogu se koristiti i u svrhu poboljšanja komunikacije, primjerice dodavanjem opcije za izgovaranje prepoznatog slova ili izraza [21], omogućila bi se i udaljena komunikacija. Mogućnosti izrade novih opcija pristupačnosti [22] [23] doista ima mnogo,

neke su ograničenije od drugih, ali svaku vrijedi barem isprobati. Cilj ovog rada nije bio samo razviti programsko rješenje zadanog problema, već i osvijestiti činjenicu da među nama postoje ljudi koji se koriste hrvatskim jezikom, ali ih mi svejedno ne razumijemo. Stoga se nadam, da će ovakvih radova biti sve više u skoroj budućnosti.

6 Literatura

[1] Ben Caldwell, Michael Cooper, Loretta Guarino Reid, Gregg Vanderheiden
Web Content Accessibility Guidelines (WCAG) 2.0
2008.

[2] Austin Waffo Kouhoué, Yoann Bonavero, Thomas Bouétou Bouétou, Marianne Huchard
Exploring Variability of Visual Accessibility Options in Operating Systems
2021.

[3] Tatiany X. de Godoi, Deógenes P. da Silva Junior, Natasha M. Costa Valentim
A Case Study about Usability, User Experience and Accessibility Problems of Deaf Users with Assistive Technologies
2020.

[4] Kavčić Dorijana
Hrvatski znakovni jezik : pregled opisanih jezičnih elemenata
2012.

[5] Fan Zhang, Valentin Bazarevsky, Andrey Vakunov, Andrei Tkachenka, George Sung, Chuo-Ling Chang, Matthias Grundmann
MediaPipe Hands: On-device Real-time Hand Tracking
2020.

[6] Službena stranica MediaPipe Hands-a i izvor fotografije
https://google.github.io/mediapipe/solutions/hands.html

[7] Kod za određivanje opisanog pravokutnika
https://python.tutorialink.com/create-a-rectangle-around-all-the-points-returned-from-mediapipe-hand-landmark-detection-just-like-cv2-boundingrect-does/

[8] Jednoručna abeceda
http://uoosbbz.hr/znakovni-jezik/abeceda/jednorucna-abeceda1

[9] Biblioteka NumPy
https://numpy.org/doc/stable/user/whatisnumpy.html

[10] Biblioteka OpenCv
hhttps://docs.opencv.org/4.x/dd/d43/tutorial_py_video_display.html

[11] Biblioteka csv
https://docs.python.org/3/library/csv.html

[12] B. Dalbelo Bašić, M. Čupić, J. Šnajder. Umjetne neuronske mreže
https://www.fer.unizg.hr/_download/repository/UmjetneNeuronskeMreze.pdf

[13] Scikit-learn biblioteka za strojno učenje
https://scikit-learn.org/stable/

[14] TensorFlow biblioteka otvorenog koda
https://www.tensorflow.org/

[15] Keras
https://keras.io/

[16] Sebastian Bock, Martin Weiß
A Proof of Local Convergence for the Adam Optimizer
2019.

[17] Funkcija gubitka sparse_categorical_crossentropy
https://www.tensorflow.org/api_docs/python/tf/keras/metrics/sparse_categorical_crossentropy

[18] Sučelje Interpreter i dohvaćanje TensorFlow Lite modela
https://www.tensorflow.org/api_docs/python/tf/lite/Interpreter

[19] Damir Krstinić, Maja Braović, Ljiljana Šerić, Dunja Božić-Štulić
Multi-label classifier performance evaluation with confusion matrix
2020.

[20] Prepoznavanje znakovnog jezika
https://github.com/arpita739/Real-time-Vernacular-Sign-Language-Recognition-using-MediaPipe-and-Machine-Learning

[21] Pretvaranje teksta u zvuk
https://www.geeksforgeeks.org/convert-text-speech-python/

[22] Kontroliranje miša pokretom
https://www.faun.dev/c/stories/qramkrishna/air-mouse-doing-mouse-operations-using-finger-gestures/

[23] Podešavanje glasnoće pokretima ruku
https://github.com/Vibhugupta10616/Volume-Hand-control

Sažetak

Autorica: Lucija Mičić

Naslov: Automatsko prepoznavanje znakovnog jezika

U ovom radu opisan je postupak izrade aplikacije za prepoznavanje jednoručne abecede hrvatskog znakovnog jezika koristeći metode strojnog učenja, preciznije, umjetne neuronske mreže. Sadržaj rada započinje definiranjem korištenih tehnologija, poput radnog okvira *MediaPipe Hands* i biblioteke *OpenCv*, te načinom njihove primjene. Preduvjet implementaciji programa za prepoznavanje, bile su dobro definirana baza s prilagođenim koordinatama, te model neuronske mreže. Na samom kraju, detaljno je opisan i algoritam za prepoznavanje slova s dinamičnim pokretom, temeljen na analizi odnosa koordinata u početnom i krajnjem trenutku pokreta.

Ključne riječi: hrvatski znakovni jezik, abeceda, neuronske mreže, *MediaPipe Hands*

Summary

Author: Lucija Mičić

Title: Automatic sign language recognition

This paper describes the process of creating applications for recognizing the single-handed alphabet of Croatian sign language using machine learning methods, more precisely, artificial neural networks. The content of the paper begins with defining used technologies, such as the *MediaPipe Hands* framework and *OpenCv* libraries, and how to apply them. Prerequisites for the implementation of the recognition program were well-defined database with customized coordinates and a neural network model. At the very end, an algorithm for recognizing letters with dynamic motion is described in detail, based on the analysis of coordinates at the beginning and end of motion.

Key words: Croatian sign language, alphabet, neural networks, *MediaPipe Hands*