SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I

INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni diplomski studij računarstva

DETEKCIJA ZNAKOVNOG JEZIKA U STVARNOM VREMENU

Projektni zadatak

Bruno Šimunović

Osijek, 2022.

Sadržaj

[1. UVOD 3](#_Toc96988525)

[1.1. Cilj projektnog zadataka 3](#_Toc96988526)

[2. KORIŠTENE TEHNOLOGIJE 4](#_Toc96988527)

[2.1. OpenCV biblioteka 4](#_Toc96988528)

[2.2. NumPy 4](#_Toc96988529)

[2.3. Tensorflow 4](#_Toc96988530)

[2.4. Mediapipe 5](#_Toc96988531)

[3. DUBOKO UČENJE 6](#_Toc96988532)

[3.1. PROPAGACIJA UNAPRIJED I UNAZAD 7](#_Toc96988533)

[3.2. KONVOLUCIJSKA NEURONSKA MREŽA 8](#_Toc96988534)

[4. IZRADA PROJEKTNOG ZADATKA 10](#_Toc96988535)

[4.1. Prikupljanje podataka 10](#_Toc96988536)

[4.2. Treniranje modela 11](#_Toc96988537)

[4.3. Detekcija u stvarnom vremenu 13](#_Toc96988538)

[4.4. Evaluacija modela 13](#_Toc96988539)

[4.4.1. Rezultati testiranja 14](#_Toc96988540)

[5. ZAKLJUČAK 19](#_Toc96988541)

[6. LITERATURA 20](#_Toc96988542)

# UVOD

Znakovni jezik je osnovni način komunikacije sa gluhim ljudima. Zbog malog broja takvih osoba većina ljudi se ne susreće sa znakovnim jezikom, te prilikom dolaska u situaciju gdje bi im bio potreban ne bi imali načina za sporazumijevanje. Kako bi se problem ove vrste izbjegao današnja tehnologija nam omogućava kreiranje aplikacija koje pomažu sa prijevodom znakovnog jezika i time rješavaju problem komunikacije.

## Cilj projektnog zadataka

Cilj ovog projekta je rješavanje problema komunikacije znakovnim jezikom sa osobom koja ga ne razumije. Rješenje mora prepoznati simbol znakovnog jezika u stvarnom vremenu te ga ispisati na ekran.

# KORIŠTENE TEHNOLOGIJE

Za rješavanje problema projektnog zadatka potrebno je koristiti biblioteke koje su podržane od strane programskog jezika Pythona. Za rješavanje i prepoznavanje znakovnog jezika u stvarnom vremenu koriste se biblioteke OpenCV, NumPy i Tensorflow, Mediapipe.

## OpenCV biblioteka

OpenCV [1] (engl. Open Source Computer Vision) je biblioteka otvorenog koda koja sadrži programske funkcije za računalni vid i strojno učenje. Izgrađen je kako bi osigurao zajedničku infrastrukturu za aplikacije računalnog vida i ubrzao korištenje strojne percepcije u komercijalnim proizvodima. Biblioteka sadrži više od 2500 optimiziranih algoritama, među kojima je i mnogo najsuvremenijih algoritama u području računalnog vida i strojnog učenja. OpenCV se primjenjuje u prepoznavanju osoba preko kamere, praćenja ljudskih pokreta, prepoznavanja krajolika, praćenja objekata u pokretu, obavljanju određenih operacija na računalu uz pomoć kamere, itd.. Postoje implementacije za C++, Python i Java programske jezike uz podršku za Windows, Linux, Android i MacOS operacijske sustave. OpenCV se najviše orijentira prema aplikacijama koje informacije primaju u stvarnom vremenu.

## NumPy

NumPy [4] je biblioteka otvorenog koda za znanstveno računanje u programskom jeziku Python. Sadrži podršku za velike, višedimenzionalne nizove i matrice s velikom zbirkom matematičkih funkcija visoke razine za rad na tim nizovima. Korištenje NumPy u Python-u omogućuje funkcionalnosti koje su usporedive s korištenjem MATLAB programa. NumPy se može koristiti kao višedimenzionalni spremnik generalnih podataka.

## Tensorflow

TensorFlow [3] je besplatna softverska knjižnica otvorenog koda za strojno učenje i umjetnu inteligenciju. Može se koristiti u nizu zadataka, ali se posebno fokusira na obuku i zaključivanje dubokih neuronskih mreža. TensorFlow je razvio tim Google Brain za internu Googleovu upotrebu u istraživanju i proizvodnji. Treniranje modela može se izvesti preko procesorske ili grafičke jedinice. Mogućnost integriranja u Python programski jezik čini ga pogodnim za korištenje prilikom kreiranja i evaluacije modela.

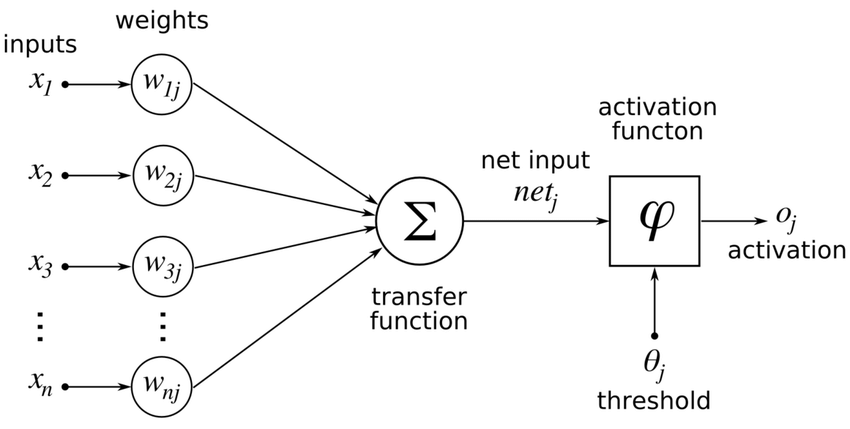
## Mediapipe

Mediapipe [2] je framework otvorenog koda izrađen od Googlea za pomoć pri kreiranju programskih ili aplikacijskih rješenja koje koriste metode strojnog učenja. Više platformski, dobro optimiziran te multimodalan. Omogućuje detekciju objekata, segmentaciju kose i lica, praćenje pokreta, detekciju čovjeka te njegovih ekstrema itd.

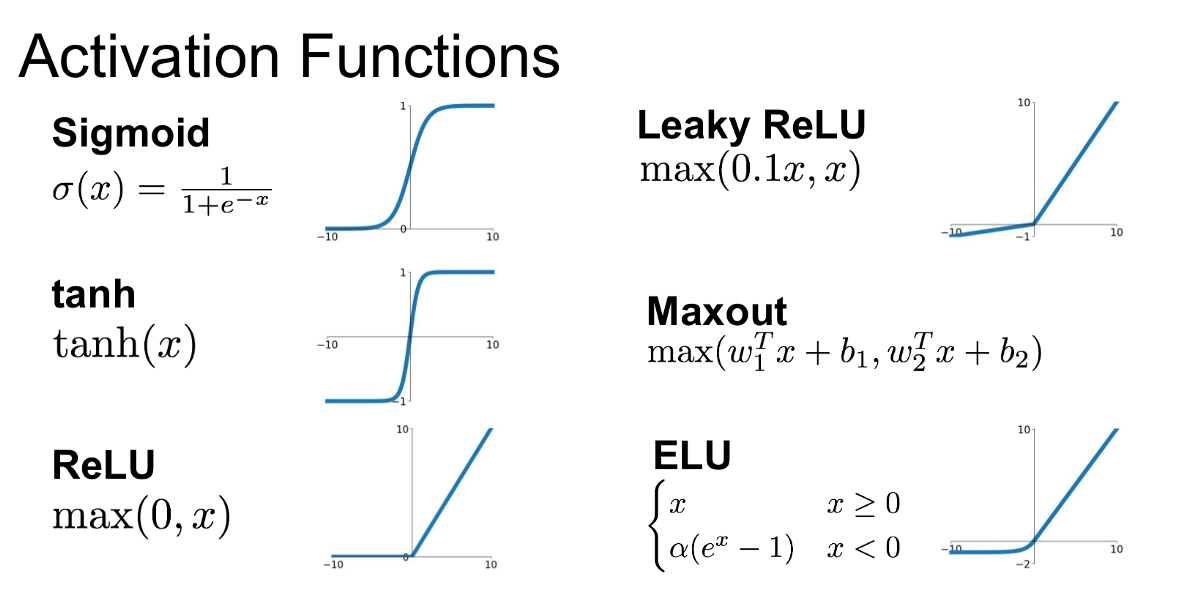
# DUBOKO UČENJE

Duboko učenje [5] je grana strojnog učenja koja za složenu vrstu podataka nudi rješenja do kojih se dolazi slijedom istreniranih nelinearnih transformacija. Sastoji se od velikog broja procesora zvanih neuroni koji pokušavaju imitirati ljudski mozak. Trenutno se koristi u raznim stvarnim problemima kao npr. Automobilska industrija, predviđanje rasta/pada dionica, kamerama, medicini itd. Potreba za dubokim učenjem stvara se kada je potrebno procesirati veliki broj značajki gdje su podaci nestrukturirani.

Duboko učenje se često naziva i duboka neuronska mreža zbog korištenja arhitekture neuronske mreže. Takva mreža se sastoji od brojnih neurona (Slika 3.1.) koji sadrže vrijednost koja se mjenja tokom učenja zvana težina. Dodatno, pri učenju neuronima aktivacijska funkcija omogućava korištenje korisnih i odbacivanje nerelativnih informacija, tj. onda omogućava učenje. Postoje razne aktivacijske funkcije [7] (Slika 3.2.) , a među najpoznatijima su reLU, Sigmoid, Leaky reLu, Tanh. Ovisno o vrsti našeg problema biramo tip aktivacijske funkcije. Neki neuroni mogu aktivirati niz drugih neurona u ovisnosti kao je neuronska mreža istrenirana i time se dolazi do predviđenog rezultata.



**Slika 3.1.** Neuron neuronske mreže

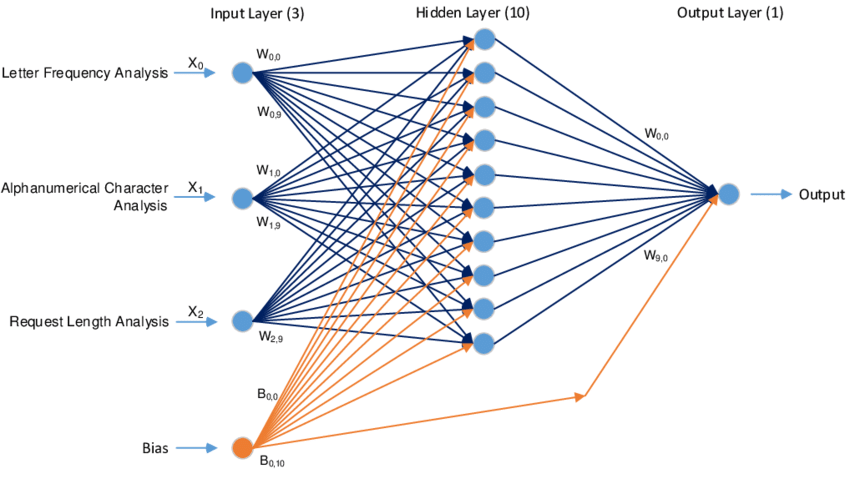
****

**Slika 3.2.** Aktivacijske funkcije

Neuronska mreža (Slika 3.3.) sastoji se od ulaznog sloja, srednjih slojeva i izlaznog sloja. Unutar ulaznog sloja možemo imati više ulaza sa ulaznim vrijednostima koje se prosljeđuju srednjem sloju. Srednji sloj se još naziva i skriveni sloj, a može ih biti više. Što je više skrivenih slojeva to je mreža kompleksnija, te ima mogućnosti rješavanja kompleksnijih problema. I na kraju imamo izlazni sloj gdje dobivamo rješenje neuronske mreže. Neuronska mreža ima poseban proces učenja a to je propagacija unaprijed i unazad. Postoji par vrsta neuronskih mreža poput konvolucijske neuronske mreže,neuronske mreže s povratnom vezom, kohonelove samo organizirajuče mreže itd.

## PROPAGACIJA UNAPRIJED I UNAZAD

Propagiranje naprijed je način prelaska s ulaznog sloja (lijevo) na izlazni sloj (desno) u neuronskoj mreži. Proces pomicanja s desna na lijevo tj. unatrag od izlaznog prema ulaznom sloju naziva se propagacija unatrag. Propagacija unatrag je poželjna metoda za prilagodbu težina budući da se brže približava kako se krećemo od izlaza do skrivenog sloja. Ovdje mijenjamo težine skrivenog sloja koji je najbliži izlaznom sloju, ponovno izračunavamo gubitak i ako je potrebno dodatno smanjiti grešku onda ponavljamo cijeli proces i tim redoslijedom idemo prema ulaznom sloju.



**Slika 3.3.** Neuronska mreža

## KONVOLUCIJSKA NEURONSKA MREŽA

Konvolucijske neuronske mreže (Slika 3.4.) razlikuju se od ostalih neuronskih mreža po svojoj superiornoj izvedbi sa ulazima slike, govora ili audio signala. Imaju tri glavne vrste slojeva, a to su: Konvolucijski sloj, Pooling sloj i potpuno povezani sloj. Konvolucijski sloj je prvi sloj konvolucijske mreže. Dok konvolucijski slojevi mogu biti praćeni dodatnim konvolucijskim slojevima ili pooling slojevima, potpuno povezani sloj je završni sloj. Sa svakim slojem, konvolucijska neuronska mreža povećava svoju složenost, identificirajući veće dijelove slike. Raniji slojevi fokusiraju se na jednostavne značajke, kao što su boje i rubovi. Kako slike napreduju kroz slojeve konvolucijske neuronske mreže, on počinje prepoznavati neke uzorke dok konačno ne identificira željeni objekt.



**Slika 3.5.** Konvolucijska neuronska mreža

# IZRADA PROJEKTNOG ZADATKA

Projekt je izrađen u programskom jeziku Python, koristeći biblioteke OpenCV, Tensorflow, Mediapipe i Numpy. Podaci koji se koriste su prikupljeni sa stranice kaggle te su dva različita skupa podataka kombinirana. Kreirani su različiti modeli konvolucijske mreže za riješavanje problema te najbolji je korišten za testiranje. Problem detekcije znakova znakovnog jezika predstavlja klasifikacijski problem.

## Prikupljanje podataka

Za treniranje konvolucijske neuronske mreže odabrani su podaci sa dva različita izvora stranice kaggle kako bi se prikupio što veći broj slika za znakovni jezik. Skup podataka sadrži 80515 slika i 36 klasa. Klase su brojevi i slova znakovnog jezika. Prilikom prve obrade podataka primijenjena je metoda detekcije najznačajnijih točaka dlana, zbog loših rezultata slika je odbačena. Slike za treniranje su korištene u orginalnom stanju (Slika 4.1.).



**Slika 4.1.** Prikupljeni podaci

Prije kreiranja modela mreže prvo je definirana veličina slike i broj slika koji će se trenirati po koraku, te se učitavaju slike pomoću funkcije *flow\_from\_directory( )* i priređuju za treniranje *train\_ds* i za testiranje *test\_ds* pomoću *tensorflow* funkcije *ImageDataGenerator().* Funkciji prosljeđujemo veličinu seta za validaciju što je u ovom slučaju 20% i dodatno parametre za uređivanje slika kako bi se mreža susrela sa različitim slučajevima. Bitno je napomenuti da *ImageDataGenerator()* mijenja vrijednosti piksela slika na vrijednosti između 0 i 1 tako što dijeli svaki piksel sa vrijednosti od 255.

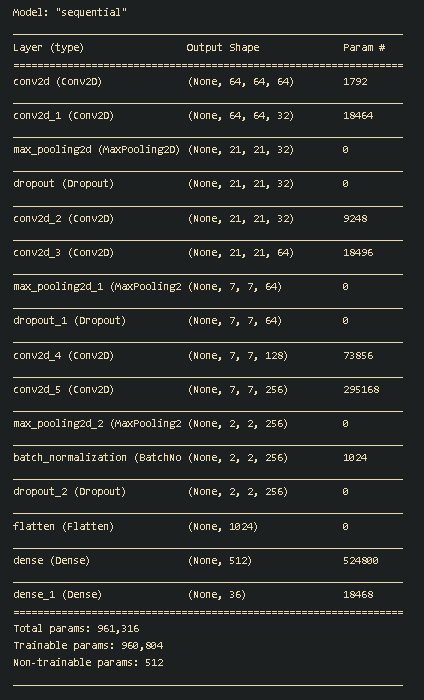
## Treniranje modela

Kako bi se što bolje istrenirala mreža primijenjeno je više modela mreža sa različitim slojevima, različite veličine slika i serija treniranih podataka. Prilikom treniranja mreže korišten je *batchSize* od 20, 32, 64, 126 što utječe na broj podataka koji se treniraju po koraku. Korištena je veličina slike od 256x256, 126x126, 64x64 piksela. Od slojeva je korišten ulazni, konvolucijski, maxpooling, potpuno povezani sloj. Konvolucijski sloj *Conv2D()* kreira konvolucijsku jezgru i time obrađuje slike u ovisnosti o veličini jezgre i broju filtera. Dodatno, koristi se reLU aktivacijska funkcija zbog brzine i jednostavnosti, te se smatra boljom od sigmoid funkcije za neuronske mreže. Nakon konvolucijskg sloja koristi se Maxpooling sloj koji smanjuje broj dobivenih parametara, uz to se koristi i *Dropout() [6]* sloj. Takav sloj nam služi da smanjimo „over-fitting“ mreže tako što ignoriramo određene neurone. Nakon svih tih slojeva korišten je *BatchNormalization()* sloj radi stabilizacije modela. Na kraju ide potpuno povezani sloj u koji sastoji se od više slojeva kako bi postigli jednodimenzionalni vektor te dobili željene vrijednosti vektora. Za sloj koristimo *Flatten()* sloj i *Dense()* sloj sa softmax aktivacijskom funkcijom. Model sa najboljim rezultatima za ovaj problem sadrži 6 konvolucijskih slojeva sa filterima od 32, 64, 128, te 3 Maxpooling sloja, 3 Dropout sloja u iznosu od 0.2 i 0.5, 1 BatchNormalization sloj i na kraju potpuno povezani sloj (Slika 4.2.).

S obzirom da imamo više klasa za funkcija gubitka uzimamo *categorical crossentropy* funkciju. Gubitak prikazuje rezultat usporedbe predviđene klase s onom koja je zapravo na izlazu. Rezultat se računa tako da se „kažnjava“ vjerojatnost prema pogrešci udaljenosti od očekivane vrijednosti izlaza, odnosno zbrajaju se pogreške svakog podatka. Također je minimiziran, što znači da manje vrijednosti predstavljaju bolji model.

Za optimizaciju modela je korišten Adam algoritam. Adam algoritam se smatra jednim od najboljih algoritama za primjenu u slučaju konvolucijske neuronske mreže. Dodatno je postavljeno da model prati točnost seta za testiranje pri učenju, ako dođe do većeg broja ponavljanja da zaustavi treniranje kako ne bi došlo do „over-fittinga“ . U ovom slučaju ne smije biti 5 ponavljanja bez napretka.

U početku su modeli bili jednostavniji i pokušavalo se dobiti što veću preciznost neovisno o „over-fittingu“, a poslije se na najboljem modelu taj „over-fitting“ pokušavao reducirati što je rezultiralo ovim modelom.



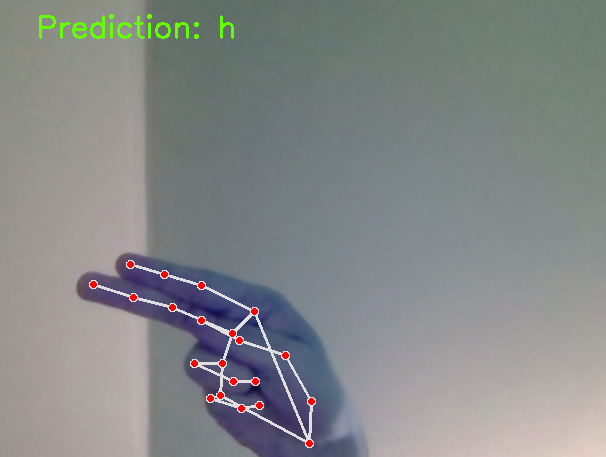
**Slika 4.2.** Model konvolucijse neuronske mreže

## Detekcija u stvarnom vremenu

Detekcija znaka znakovnog jezika se nalazi u *real\_time\_detection\_asl.ipynb* datoteci. Za detekciju je korištena OpenCV i mediapipe biblioteka. Prvo je potrebno učitati željeni istrenirani model što se postiže naredbom *load\_weights()* iz tensorflow biblioteke. Nakon toga se postavlja model za prepoznavanje karakteristika ruke iz mediapipe biblioteke što će se koristiti pri obradi „real-time“ videozapisa kao bi se lakše prepoznala ruka. Za otvaranje kamere i snimanje u stvarnom vremenu se koristi *VideoCapture()* iz biblioteke OpenCV*.* Korištenjem te funkcije dobivamo niz slika koji nastaju iz videa koji se trenutno odvija. Te slike prvo se obrađuju pomoću *blur()* efekta zbog zamućivanja pozadine i još se na njima crtaju važne točke ruke pomoću funkcije *drawHandKeyPoints()* da se ruka što bolje istiće. Kako bi obavili predikciju unutar stvarnog vremena poziva se funkcija *predict()* iz tensorflow biblioteke i njoj se prosljeđuje slika. Funkcija vraća vrijednost od 0 do 36 koje pretvaramo u stvarnu vrijednost i time dobivamo vrijednost našeg istreniranog modela. Na kameri tokom testiranja se može vidjeti rezultat naše procjene (Slika 4.3).

Slika na kojoj se prikazuje tekst, zid

Opis je automatski generiranSlika na kojoj se prikazuje tekst

Opis je automatski generiran

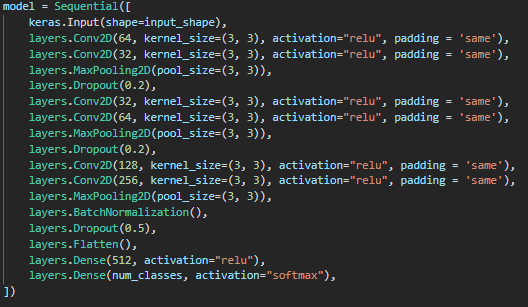
**Slika 4.3.** Testiranje modela u „real-time“

## Evaluacija modela

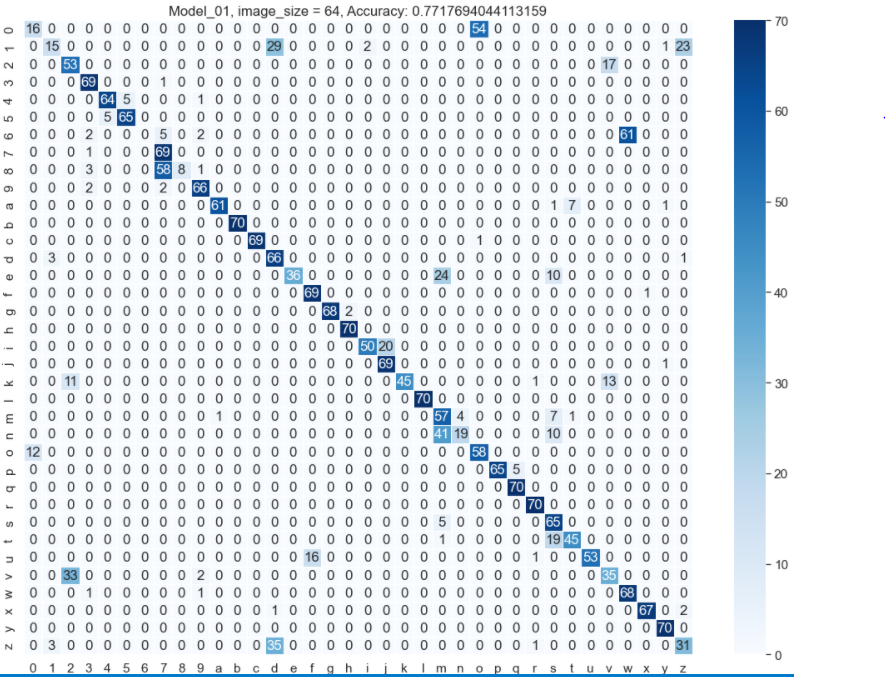
Nakon što smo model trenirali slijedi evalucija modela. Modeli su testirani na 2500 slika. Treniranje je pokrenuo više puta te je najbolje rezultate model\_01 s veličinom slike od 126 piksela. Ostali modeli su imali slične rezultate ili lošije što će biti vidljivo u nastavku. Svim modelima je zajedničko da teško prepoznaju brojeve. To se događa jer količina slika brojeva znakovnog jezika je izrazito malo.

### Rezultati testiranja

Za prvi test je korišten model\_01 (Slika 4.4) . Veličina serije je postavljena na 32, a slika na 64 piksela. Iz ovog modela su dobiveni najbolji rezultati s točnosti od 77% (Slika 4.5) .

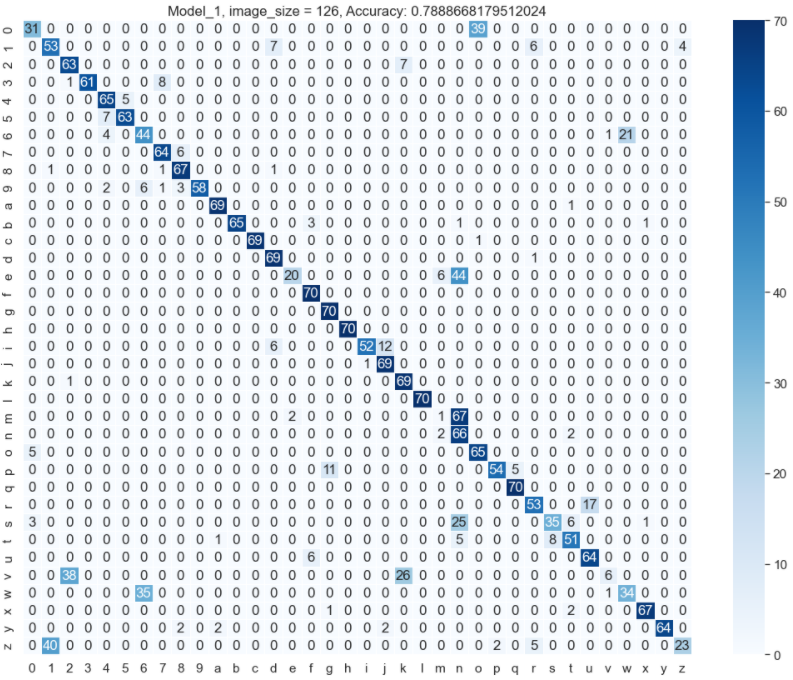


**Slika 4.4.** Model\_01



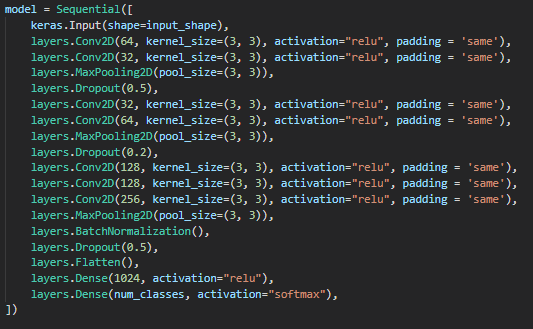
**Slika 4.5.** Matrica zabune za Model\_01, veličina slike 64 piksela

Drugi test je izvršen na istom modelu, ali sa različitom veličinom slike. Veličina slike je promjenjena na 126 piksela. U ovome modelu točnost raste na 78% i pokazao se boljim pri detekciji u stvarnom vremenu tako da je on izabran za korištenje. Rezultati matrice zabune su vidljivi na Slici 4.6.

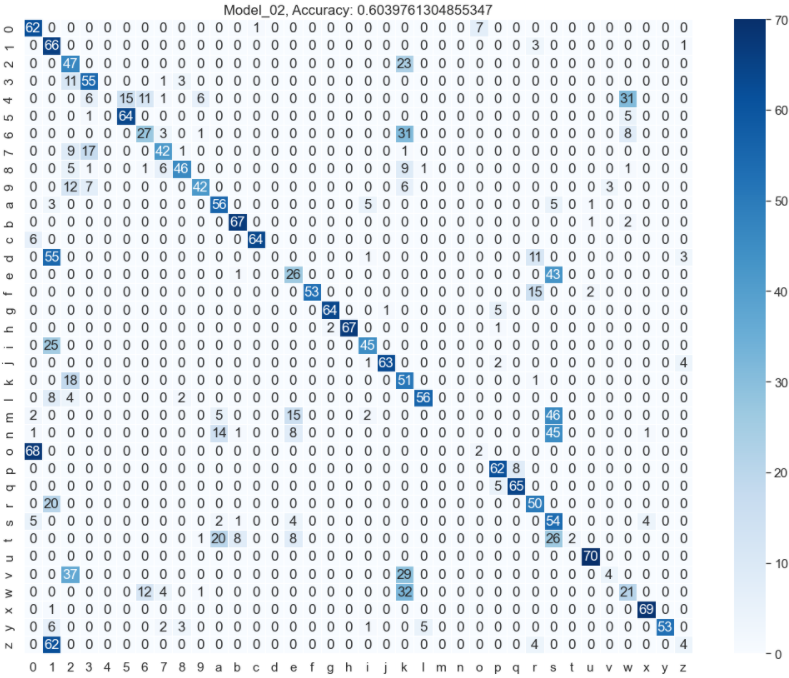


**Slika 4.6.** Matrica zabune za Model\_01, veličina slike 126 piksela

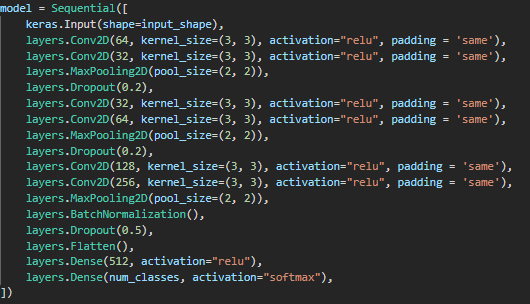
Obavljeno je još par testova na drugim modelima što se može vidjeti u nastavku. Modeli su imali lošije rezultate pa nisu testirani na različitim veličinama slika i serija. Ti modeli su sa različitim veličinama filtera u konvolucijskim slojevima ili sa povećanim odbacivanju parametara. Iz toga je sljedilo da model nije dobro istreniran te je došlo do lošijih rezultata.



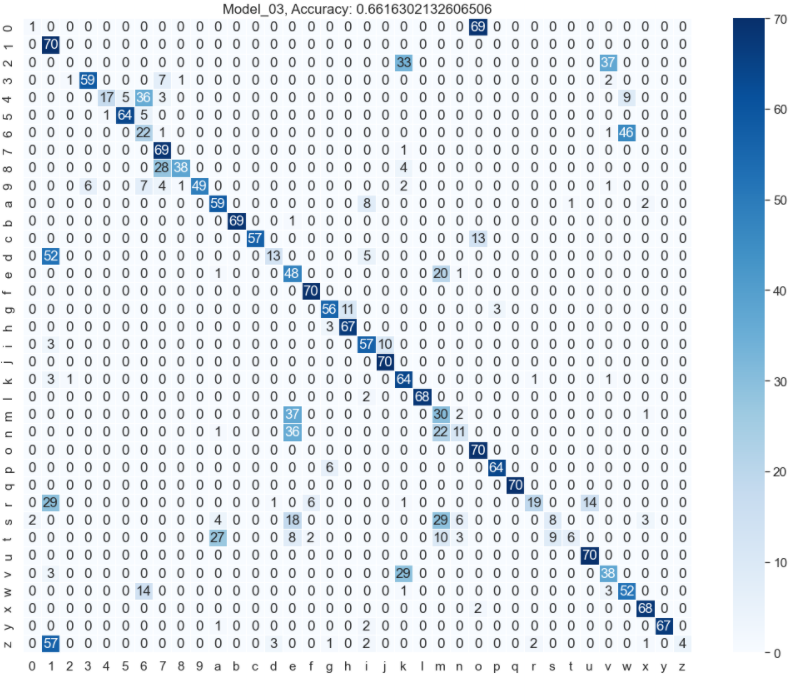
**Slika 4.7.** Model\_02



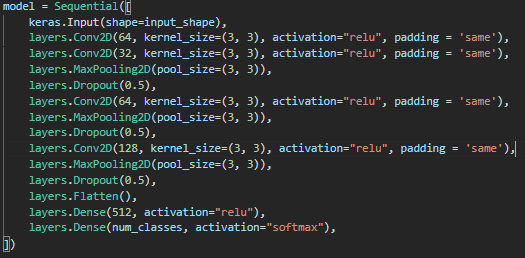
**Slika 4.8.** Matrica zabune za Model\_02, veličina slike 64 piksela



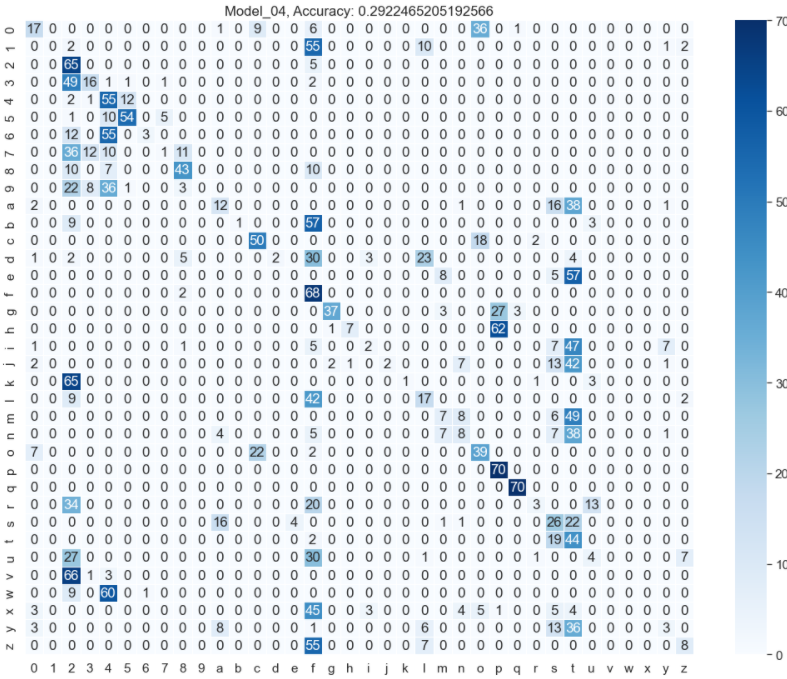
**Slika 4.9.** Model\_03



**Slika 4.10.** Matrica zabune za Model\_03, veličina slike 64 piksela



**Slika 4.11.** Model\_04



**Slika 4.12.** Matrica zabune za Model\_04, veličina slike 64 piksela

# ZAKLJUČAK

U ovome radu, cilj je bio kreirati algoritam za detekciju znakova znakovnog jezika. Rad je ovisio o prikupljenim podacima i treniranju neuronske mreže. Za rješenje algoritma korištene su tri biblioteke Matplotlib, OpenCV, Mediapipe i NumPy u Pythone 3.9.6 programskom jeziku. Većinski dio programa se odnosio na treniranje mreže preko par različitih tipova modela. Nakon obavljenog testiranja, možemo vidjeti da algoritam ne daje najbolje rezultate u svim situacijama. Neuronska mreža postiže i do 78% točnosti u većini situacija, ali je loša u prepoznavanju brojeva zbog manjeg broja uzoraka. Neki znakovi mreži stvaraju probleme zbog sličnosti. Tako da bi zaključak bio, kako bi poboljšali algoritam potrebno je koristiti bolji set podataka i još dodatno dodavati slojeve mreže kako bi rezultati bili zadovoljavajući za svakodnevnu primjenu.

# LITERATURA

[1] OpenCV https://opencv.org/about/

[2] MediaPipe, <https://google.github.io/mediapipe/>

[3] Tensorflow, <https://www.tensorflow.org/>

[4] NumPy <https://numpy.org/about/>

[5] Neural Network Full Course <https://www.youtube.com/watch?v=ob1yS9g-Zcs>

[6] Dropout layer <https://machinelearningmastery.com/dropout-regularization-deep-learning-models-keras/>

[7] Aktivacijske funkcije <https://www.v7labs.com/blog/neural-networks-activation-functions>