SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I

INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

**DETEKCIJA ZNAKOVNOG JEZIKA**

**Raspoznavanje uzoraka i strojno učenje**

David Hodak

Osijek, 2020.

Sadržaj

[1. UVOD 1](#_Toc81378811)

[1.1. Zadatak projektnog zadatka 1](#_Toc81378812)

[2. KORIŠTENE TEHNOLOGIJE I ALATI 2](#_Toc81378813)

[2.1. MEDIAPIPE 2](#_Toc81378814)

[2.2. Tensorflow 3](#_Toc81378815)

[3. DUBOKO UČENJE 4](#_Toc81378816)

[3.1. Ponavljajuće neuronske mreže 6](#_Toc81378817)

[3.1.1. LSTM 7](#_Toc81378818)

[4. IZRADA PROJEKTA 8](#_Toc81378819)

[4.1. Skupljanje podataka 8](#_Toc81378820)

[4.2. Treniranje modela 9](#_Toc81378821)

[4.3. Evaluacija modela 11](#_Toc81378822)

[4.4. Test u stvarnosti 13](#_Toc81378823)

[4.5. Problemi u implementaciji i mogućnosti poboljšanja 14](#_Toc81378824)

[5. ZAKLJUČAK 15](#_Toc81378825)

[6. LITERATURA 16](#_Toc81378826)

# UVOD

Znakovni jezik način je komunikacije gluhih ljudi te bi im mogućnost postojanja načina za sporazumijevanje s ljudima koji dobro čuju ali ne znaju znakovni jezik bila od velike pomoći.   
Upravo će se u ovom radu istražiti mogućnost korištenja računala kao bi naučilo pokrete potrebne za sporazumijevanje znakovnim jezikom. Kako bi se to postiglo koristit će se model duboke mreže i open-source platforma MediaPipe koja omogućava detektiranje ruku, lica i tijela uz implementaciju u programskom jeziku Pythone.

## Zadatak projektnog zadatka

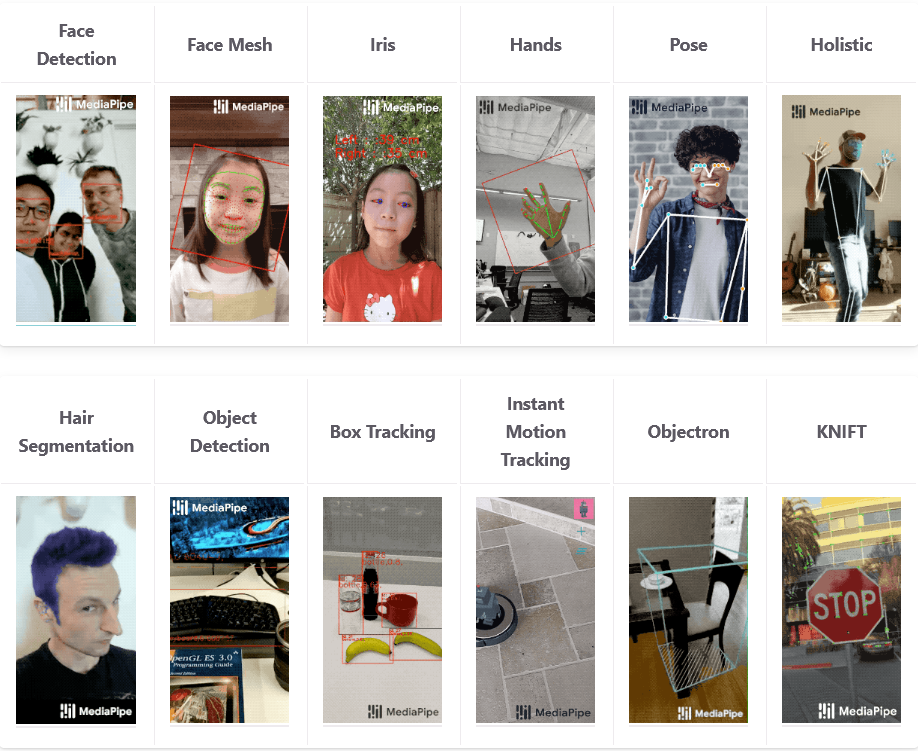
Izraditi model neuronske mreže sposobne za predviđanje riječi znakovnog jezika na osnovu ulaza s kamere koristeći programski jezik Python i potrebne biblioteke. Izraditi evaluaciju modela te komentirati rezultat.

# KORIŠTENE TEHNOLOGIJE I ALATI

## MEDIAPIPE

Mediapipe 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[[1]](#footnote-1). Omogućuje detekciju objekata, segmentaciju kose i lica, praćenje pokreta, detekciju čovjeka te njegovih ekstrema itd. Redovno se proširuje i unaprjeđuje.

Razlog odabira ovog *frameworka* za projekt je mogućnost detekcije ruku, a s time i kretnji čime se uveliko olakšava način dobivanja značajki potrebnih za treniranje neuronske mreže. Osim toga, njegova integracija u mnoge programske jezike je vrlo jednostavna te se sve postavi u par minuta.



**Slika 2.1.** Mogućnosti MediaPipe platforme

## Tensorflow

Platforma otvorenog koda za strojno učenje. Sastoji se od mnoštva biblioteka i resursa kojima se omogućava korisnicima da razvijaju svoje ideje. Treniranje modela može se izvesti preko procesorske ili grafičke jedinice. Uz to omogućuje i korisnicima vizualni prikaz krajnjih rezultata treniranja modela ali i rezultata tijekom treniranja uz pomoć vizualizacijskog alata TensorBoard. Mogućnost integriranja u Python programski jezik čini ga pogodnim za korištenje prilikom kreiranja i evaluacije modela.

# DUBOKO UČENJE

Grana strojnog učenja koja za složene reprezentacije podataka nudi rješenja do kojih se dolazi slijedom naučenih nelinearnih transformacija. Koristi se u autonomnim automobilima, kamerama, mobitelima, web stranicama, statističkim analizama, robotici, računalnim igrama itd.   
Problem je u izrazito zahtjevnim podacima, koji pristižu u velikim količinama ali i brzinama (primjerice. računalni vid) gdje su se metode dubokih mreža pokazale bolje od konvencionalnih metoda strojnog učenja.

Većina metoda dubokog učenja koristi arhitekturu neuronskih mreža, zbog čega ih se i naziva duboke neuronske mreže. Osnovna jedinica je neuron. Neuron prima podatke ( ) koji se množe sa specifičnom težinom i zatim se takvi podaci predaju funkciji f koja se još naziva i aktivacijska funkcija čiji produkt se predaje izlazu. Postoje mnoge aktivacijske funkcije(relu, tanh, sigmoid, gelu itd)(Tablica 3.1.) te njihova upotreba ovisi o vrsti problema i rješenju kojeg želimo dobiti. Razlog aktivacijskih funkcija je da produbljuju nelinearnost modela, čime se omogućuje modelu duboke mreže da do određene margine pogreške predviđaju funkcije koje ne slijede linearnost.

**Slika na kojoj se prikazuje tekst, sport

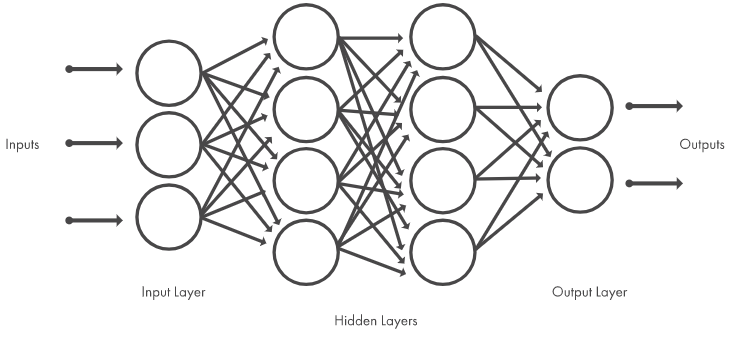
Opis je automatski generiran**

**Slika 3.1.** Shema neurona

Svaka neuronska mreža (slika 3.2.) sastoji se od jednog ulaznog sloja, kroz koji prolaze ulazni podaci koji se prosljeđuju na iduće slojeve nakon što se izvrše proračuni na njima. Središnji slojevi nazivaju se skriveni. Moguće je imati nijedan skriveni sloj. Duboke mreže imaju više skrivenih slojeva, iz čega i naziv duboke mreže. To im omogućava rješavanje kompleksnijih problema, korištenje složenijih funkcija i otkrivanje novih značajki. Na kraju se nalazi izlazni sloj koji vraća gotovi rezultat predviđanja.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | **Plot** | **Function f(x)** | **Derivative of f, f'(x)** | **Range** | **Order of continuity** |
| **Sigmoid** |  |  | **f(x)(1-f(x)** | **{0,1}** |  |
| **Hyperbolic tangent** |  |  |  | **(1,1)** |  |
| **Rectified linear unit** | **Slika na kojoj se prikazuje crveno, jarko, raznobojno, popločano  Opis je automatski generiran** | **=max{0, x}** |  | **[0, ∞)** |  |

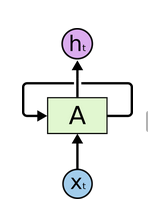
**Tablica 3.1.** Primjer nekih aktivacijskih funkcija i njihove karakteristike

****

**Slika 3.2.** Model neuronske mreže

## Ponavljajuće neuronske mreže

Neuronske mreže koje se koriste za procesuiranje sekvencijalnih podataka . Takve mreže spremno se skaliraju i na duže sekvence. Kako podaci dolaze jedan za drugim, potrebno je zapamtiti prethodne, a to je omogućeno dijeljenjem parametara kroz različite dijelove modela. Postoji petlja u njima, koja omogućava podacima da opstaju (slika 3.3.) .



**Slika 3.3.** Pojednostavljena ponavljajuća mreža

Ulazni podatak se obradi u neuronu te se šalje na izlaz, petlja dozvoljava da se podatak „zapamti“ i proslijedi na drugi dio obrade. Primjerice, kako bi predvidjeli sljedeću riječ u rečenici potrebno je znati kontekst rečenice prije. Petlja se može shvatiti kao više kopija jedne te iste mreže, što zapravo i funkcionira tako.

Slika na kojoj se prikazuje tekst, sat

Opis je automatski generiran

**Slika 3.4.** Ponavljajuća neuronska mreža

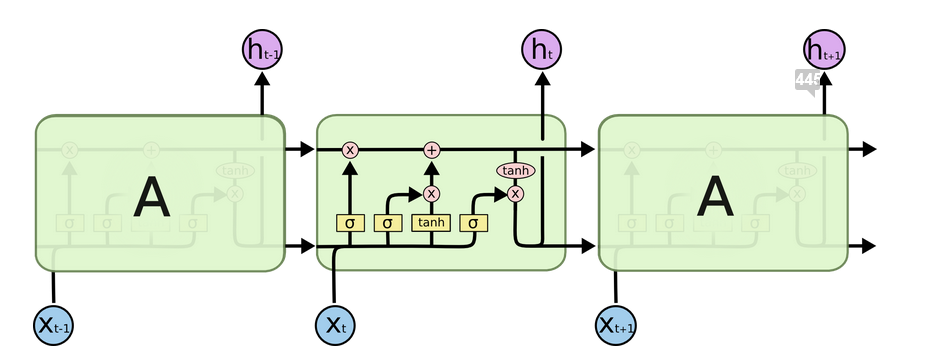
Ponavljajuće neuronske mreže imaju problem s pamćenjem podataka na duže staze. Nekad je potrebno znati kontekst rečenice koja je bila mnogo prije kako bi se predvidjela sljedeća riječ. Što je poveznica između relevantnih podatka veća to mreža sve teže može ispravno naučiti.   
Još jedan problem je i nestajući gradijent. Problem nastaje kada se *backpropagation* algoritam koji je zadužen za računanje gradijenta pomiče unatrag kroz različite slojeve neuronske mreže kako bi ažurirao težine neurona. S obzirom da se tu događa množenje, gradijent koji je izračunat u dubljim slojevima, smanjuje se kako se kreće natrag kroz mrežu. Primjerice, ako broj manji od 1 često množimo, približavamo se nuli. S obzirom da gradijent zajedno s *backpropagation* algoritmom i *cost* funkcijom utječe na težine neutrona, njegova mala vrijednost neće imati veliki utjecaj na težine i s time učenje postaje jako sporo. Ovaj problem donekle riješila je LSTM mreža.

### LSTM

LSTM(eng. Long Short Term Memory) je posebna vrsta ponavljajuće neuronske mreže. Kreirana kako bi izbjegla probleme nedovoljno dugog pamćenja informacija. Kako je prikazano na slici 3.4., LSTM također ima oblik „lanca“ prilikom ponavljanja no ovdje se osim jednog sloja koriste četiri. (slika 3.5.). *Cell state* omogućuje informacijama da putuju kroz cijelu mrežu. Kako bi se regulirale informacije koje putuju, LSTM koristi strukture koje se nazivaju vrata. Uz pomoć njih on u svojoj memoriji može pisati, čitati i obrisati informaciju. Vrata se sastoje neuronskog sloja sa sigmoid(σ) aktivacijskom funkcijom. Sukladno tome, izlaz je 1 ili 0, što znači prolaze sve informacije ili ne prolazi nijedna.

Output vrata

Vrata



*Cell state*

*Forget* Vrata

Input Vrata

**Slika 3.5.** Shema LSTM neuronske mreže

Prva vrata na koja se nailazi su *Forget* vrata koja odlučuju koje će se informacije odbaciti prije nego se proslijede na *cell state*. Odluku radi sigmoid funkcija (1 ili 0). Sljedeća vrata su *Input*, odlučuje se koja nova informacija će se pohraniti u *cell state*. Sloj sa sigmoid funkcijom odlučuje koje informacije ažuriramo, a sloj s tanh funkcijom kreira vektor novih vrijednosti koji će ažurirati starije vrijednosti u *cell state*. Završno, treba odlučiti što će ići na izlaz. Cell state predaje vrijednost koja se filtrira kroz sloj s tahn funckijom kako bi vrijednosti bile između -1 i 1 te se pomnože sa izlazom sigmoid funkcije kako bi na izlaz izašli samo oni podaci koje želimo.   
S obzirom da vrata kontroliraju informacije koje prolaze, utječu i na gradijent te ga kontroliraju kako ne bi pobjegao previše nisko ili visoko.

# IZRADA PROJEKTA

Projekt je izrađen u programskom jeziku Python, koristeći biblioteke *opencv*, *tensorflow*, *mediapipe* i *sklearn* kao najbitnije. S obzirom da ne postoji ni jedan službeni skup podataka snimljenih izvedbi riječi znakovnog jezika, skup podatak je izrađen samostalno. Korištena je kamera laptopa kao izvor dohvaćanja video sekvenci. Problem detekcije riječi znakovnog jezika predstavlja klasifikacijski problem.

## Skupljanje podataka

MediaPipe biblioteka nudi izvlačenje ključnih točaka. Točke se očitavaju za lice, tijelo i šake, odnosno dlanove. S obzirom da se radi o znakovnom jeziku kao podaci koriste se samo točke šake, čime je i znatno poboljšanja preciznost.

Slika na kojoj se prikazuje na zatvorenom

Opis je automatski generiran Slika na kojoj se prikazuje na zatvorenom

Opis je automatski generiran

Slika na kojoj se prikazuje na zatvorenom

Opis je automatski generiran

**Slika 4.1** Prikaz točaka šake u različitim položajima

Skripta *keypoints\_extraction.py* za svrhu ima prikupiti podatke za treniranje. Kreira folder *Dataset* u kojem se nalaze folderi s nazivima riječi za koje će se mreža trenirati. Koristeći OpenCV i kameru podaci se izvlače u datoteke. Svaka riječ sastoji se od 50 sekvenci snimanja, što znači da osoba koja snima izvođenje pokreta za riječ mora to ponoviti 50 puta. Svaka sekvenca traje 60 sličica po mjeri koja stavljena u cv2.waitKey() funkciju, a u ovom slučaju je to 10ms. Svaka sličica sadrži ključne točke ruke u ovisnosti o njenom položaju. Ukoliko ruka nije očitana, polje se popunjava s nulama. Točke su spremljena kao *numpy* polje u eksterne datoteke. Znači za svaku riječ postoji 3000 polja, odnosno vektora s vrijednostima ključnih točaka. MediaPipe za ruke vraća 21 točku koja se sastoji od x, y i z vrijednosti (slika 4.2.).

Slika na kojoj se prikazuje tekst

Opis je automatski generiran

**Slika 4.2.** Ključne točke šake

Skup podataka sastoji se od 6 riječi: Pozdrav, Hvala, Oprosti, Ja, Naočale, Kuća.  
Ukupno dolazimo do brojke od 1,134,000 podataka u podatkovnom skupu.

## Treniranje modela

Treniranje se izvodi u skripti *training\_model.py*. Podaci se prvotno izvlače iz eksternih datoteka te se pretvaraju u numpy polje. Riječi se označavaju s brojevima od 0 do 5. Podaci se dijele na skup za treniranje i skup za testiranje u omjeru 0.75:0.25 te se nasumično miješaju (miješanje je u ovom slučaju određeno za sekvence, ne za podatke unutar njih tako da izvođenje pokreta za riječ i dalje ide redom).

Kreira se sekvencijalni model te se dodaje *dropout* u iznosu od 0.2. *Dropout* je metoda regularizacije neuronske mreže gdje se određeni broj neurona ignorira tijekom treninga. Razlog tome je što mreža može postati osjetljiva na specifične težine određenih neurona te dolazi do *overfittinga.* Zatim se dodaju dva LSTM sloja. Oba imaju po 24 jedinice, koriste tahn kao aktivacijsku funkciju (default). Prvi sloj ima input u obliku 60,126 što je i broj naših ključnih točaka lijeve i desne šake (126) te broj sličica u kojima je snimljen pokret. Posljednji LSTM sloj parametar *return\_sequences* ima postavljen na *false* jer njegov izlaz želimo preusmjeriti samo na sljedeći sloj. Zadnji sloj *dense* koristimo kako bi dobili željeni izlaz. Za to nam pomaže funkcija *softmax. Softmax* omogućuje da se određeni vektor realnih vrijednosti pretvori u vektor realnih vrijednosti ali sa sumom vrijednosti jednako 1.

Slika na kojoj se prikazuje tekst, sat

Opis je automatski generiran

**Slika 4.3.** Formula *softmax* funkcije

Formula (slika 4.3.) gdje predstavlja sve ulazne vrijednosti koji mogu biti bilo koji realan broj i izraz u nazivniku koji je normalizacija koja osigurava da će se sve izlazne vrijednosti zbrojiti u 1.

Riječi su označene kao string tip podatka te ih je bilo potrebno pretvoriti u numerički tip podatka, u ovom slučaju koristeći *one-hot* kodiranje. Ono se koristi kad nema nekog rednog odnosa među kategorijama, što je i slučaj ovdje gdje su riječi jednostavno nasumično odabrane. S obzirom da imamo 6 kategorija, potrebno je 6 binarnih vrijednosti. Na mjesto jedne riječi ide vrijednost 1, a na ostale 0. No, kako mreža vrši predviđanje za svaku sekvencu, a njih ima 50 po riječi, znači da nam treba 300 binarnih vrijednosti.

Slika na kojoj se prikazuje elektronički, tipkovnica

Opis je automatski generiran

**Slika 4.4.** *One-hot* kodiranje

S obzirom da imamo *one-hot* kategoričko kodiranje za parametar gubitka uzimamo *categorical crossentropy* funkciju i *categorical\_accuracy*. Gubitak prikazuje rezultat usporedbe predviđene klase s onom koja je zapravo na izlazu. Rezultat se izračunava tako da se „kažnjava“ vjerojatnost na osnovu udaljenosti od očekivane vrijednosti izlaza, odnosno sumiraju se pogreške svakog podatka. Također je minimiziran, što znači da manje vrijednosti predstavljaju bolji model. Najčešće ga koristimo za evaluaciju modela na osnovu kako uči. *Categorical accuracy* daje rezultat koliko često model pretpostavi klasu točno.

Za optimizaciju se koristi Adam. Optimizacijski algoritam koji koristi metodu prilagođavanja stupnja učenja kako bi pronašao individualnu stopu učenja za svaki parametar. Za treniranje koristimo 200 epoha, te model kad završi s treniranjem spremamo kao h5 datoteku.

Slika na kojoj se prikazuje tekst

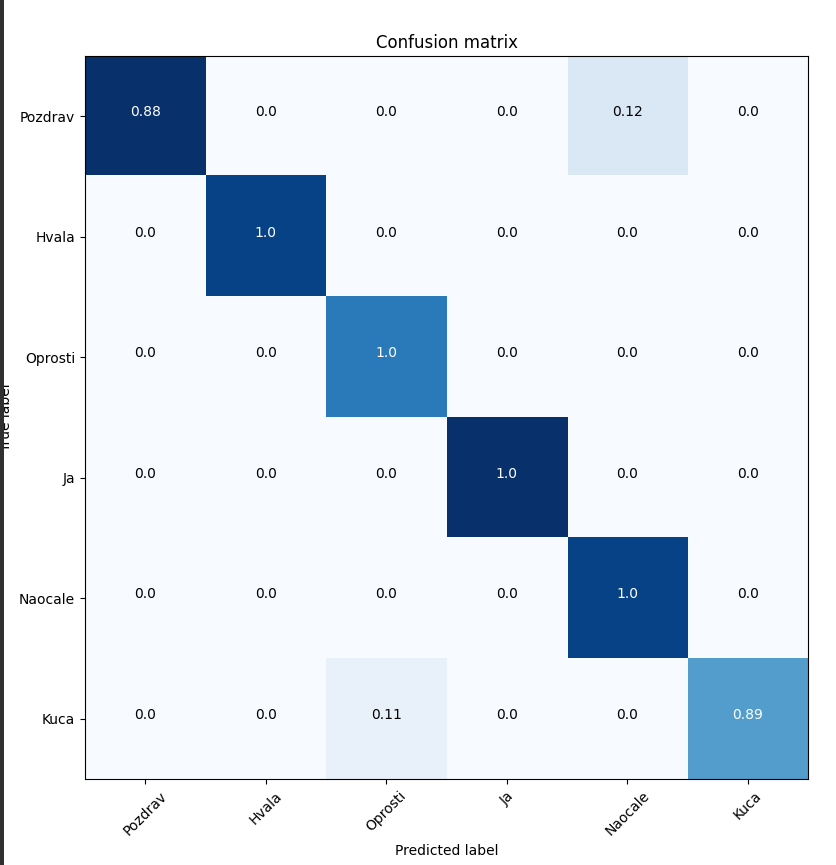
Opis je automatski generiran

**Slika 4.5.** Praćenje treniranja mreže

## Evaluacija modela

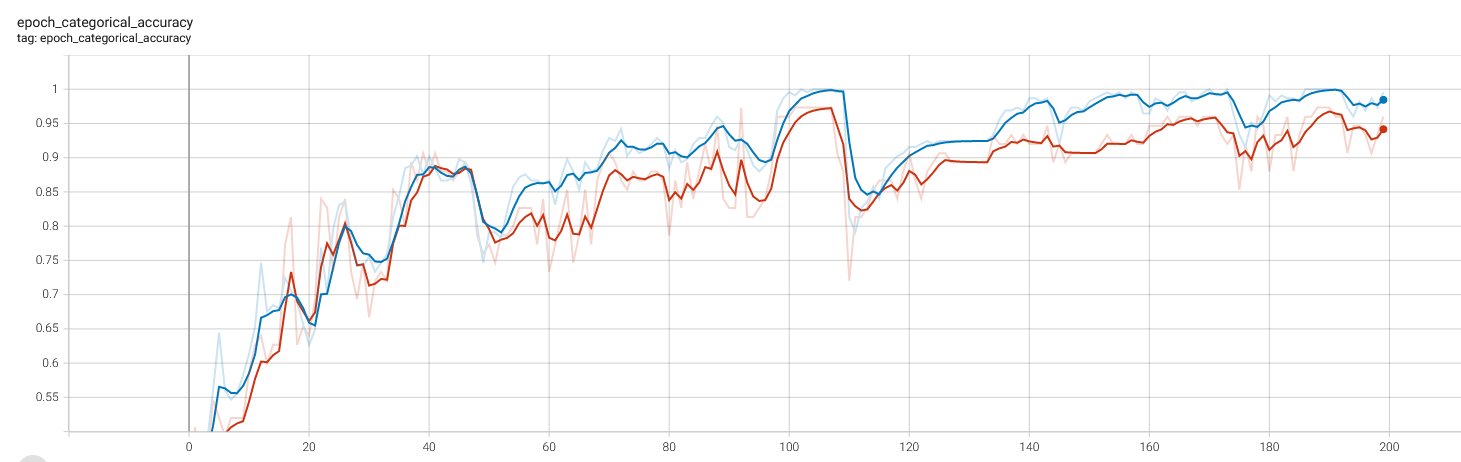
Nakon što smo model trenirali potrebno ga je evaluirati te provjeriti moraju li se hiperparametri namještati kako bismo postigli bolji rezultat. Treniranje je pokrenuo više puta te je najbolje rezultate model dao za dva LSTM sloja s 24 jedinice u svakom i treniranje kroz 200 epoha. Konačan rezultat je preciznost od 0.96 odnosno 96% te gubitak od 0.1809. Bitno je gledati validacijske rezultate jer oni najbolje pokazuju kako se mreža ponaša na novim podacima. Isto tako F1 score pokazuje da mreža ima malu količinu *false positives* i *false negatives*. To se može vidjeti i iz matrice zabune koja je kreirana u Tensorboard.

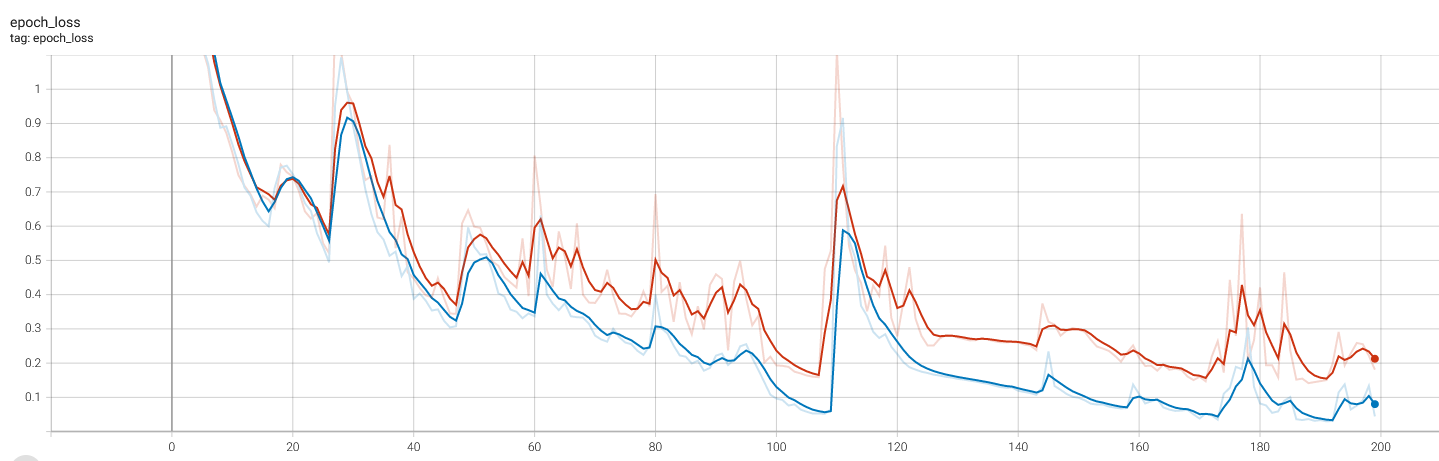
Promatrajući matricu zabune (slika 4.6.) za višeklasni problem vidimo kako riječi naočale i pozdrav te oprosti i kuća imaju neke krivo predviđene sekvence. To je i za očekivati s obzirom da su pokreti, odnosno ključne točke slični i isprepliću se. Isti problem bi se stvorio i kod više riječi sa sličnim pokretima.



**Slika 4.6.** Matrica zabune

Na grafovima preciznosti i gubitka (slika 4.7.) plava linija je validacijski skup, a crvena skup za treniranje. Promatrajući ih zaključujemo par stvari. Treniranje je napredovalo u dobrom pravcu, kako se gubitak smanjivao, povećavala se preciznost i taj trend se zadržao i prilikom naglih skokova. Postoje određeni skokovi koji su možda nastali zbog *overfittinga* u pojedinim trenutcima, greškama u algoritmu koje ne podižu nikakvu iznimku ili pak pogrešnim značajkama koje mreža očekuje, pa čak je i model postigao bolji rezultat oko 108 epohe gdje su validacijski rezultati bili bolji nego krajnji. Ipak model na kraju daje i više nego zadovoljavajuće rezultate.

****



**Slika 4.7.** Grafovi gubitka i preciznosti na trening i test podacima

## Test u stvarnosti

Skripta detection\_test.py sadrži jednostavan test modela u stvarnom vremenu. Na isti način na koji su se skupljali podaci, kroz kameru i ključne točke detektirane MediaPipe metodama, sada se ti podaci prosljeđuju u model za predikciju. Prvotno se skupi 60 sličica i svaka sličica ima 136 točaka čime je zadovoljen ulaz u model. Zbog *softmax* funkcije podaci na izlazu daju zbroj 1, pa s obzirom da jedan od podataka uvijek ima najveću vrijednost, lako se pronalazi njegova pozicija te se pozicija prosljeđuje polju s riječima, a s obzirom da je oblik izlaza iz modela i polja s riječima isti, natrag dobivamo predviđenu riječ. Riječ se prikazuje na zaslonu zajedno s vrijednošću vjerojatnosti da je to ta riječ prema detekciji modela(slika 4.8.).

Slika na kojoj se prikazuje tekst, osoba, zid, na zatvorenom

Opis je automatski generiran

**Slika 4.8**. Testiranje modela u stvarnom vremenu

## Problemi u implementaciji i mogućnosti poboljšanja

Glavni problem je sam skup podataka i način na koji se on prikuplja. Ovdje bi se uvijek moglo izvesti poboljšanje. Za veći skup podatak gdje postoje riječi s sličnim pokretima, ključne točke bi padale u približno iste regije. Točnost takvog modela bi bila zasigurno manja. To se moglo primijetiti i na ovom projektu kod riječi „pozdrav“ i „naočale“ te „kuća“ i „oprosti“. Poanta je u kvalitetnom skupljanju podataka, a da bi se to ostvarilo potrebna je bolja kamera, više sekvenci za svaku riječ, izvođenje pokreta na način na koji bi to izvela osoba koja ga razumije i koristi što znači da je za skupljanje ovakvog skupa najbolje koristi osobe koje znaju znakovni jezik. Ako bi se htio napraviti model za korištenje u stvarnom svijetu ne možemo se samo zadržati na ključnim točkama MediaPipe ili neke druge biblioteke. Potrebno je uključiti i druge značajke. Jedno od poboljšanja je i korištenje *bounding* *boxa* oko tijela osobe kako bi se točke uvijek zadržale na istim relativnim pozicijama neovisno je li osoba u gornjem ili donjem kutu te je su li podaci sakupljeni kao u ovom projektu gdje su sve točke oko sredine kamere.

# ZAKLJUČAK

Ovaj projekt je poslužio kao vježba i učenje funkcioniranja dubokih mreža s naglaskom na ponavljajuće duboke neuronske mreže te se kroz programsku implementaciju uspješno došlo do rješenja. Osim dubokih neuronskih mreža naučilo se i ponešto znakovnog jezika te spoznala činjenica koliko je potreban gluhim ljudima i kako bi aplikacija koja provodi detekciju znakovnog jezika bila od velike pomoći kako njima tako i drugim ljudima koji ih ne razumiju.

# LITERATURA

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

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

[3] N. Buduma, , str.7 i str.245-248, O'Reilly Media Inc., Sjedinjene Američke Države, 2017.

[4] How deep learning works, <https://www.mathworks.com/discovery/deep-learning.html>, posjećeno 10.07.2020.

[5] Aktivacijska funkcija, <https://en.wikipedia.org/wiki/Activation_function>

[6] LSTM, <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>

[7] N. Buduma, , str.168 O'Reilly Media Inc., Sjedinjene Američke Države, 2017.

[8] LSTM, <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>

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

[10] Funkcija gubitka, <https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/>

[11] One-hot kodiranje, <https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/>

1. različiti semiotički izvori (vizualna sredstva, boje, geste i pokreti, pogled, glas, glazba, djelatnosti ili predmeti) [↑](#footnote-ref-1)