Upravljanje uređajem metodom prepoznavanja gesti prstiju

Čubaković, Lovro

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij

UPRAVLJANJE UREĐAJEM METODOM PREPOZNAVANJA GESTI PRSTIJU

Diplomski rad

Lovro Čubaković

Osijek, 2023.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac D1: Obrazac za imenovanje Povjerenstva za diplomski ispit

Osijek, 12.09.2023.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za diplomski ispit

Ime i prezime Pristupnika:	Lovro Čubaković
Studij, smjer:	Diplomski sveučilišni studij Elektrotehnika, smjer Komunikacije i informatika'
Mat. br. Pristupnika, godina upisa:	D-1362, 07.10.2021.
OIB studenta:	01088227472
Mentor:	prof. dr. sc. Marijan Herceg
Sumentor:	,
Sumentor iz tvrtke:	Zvonimir Kaprocki
Predsjednik Povjerenstva:	prof. dr. sc. Mario Vranješ
Član Povjerenstva 1:	prof. dr. sc. Marijan Herceg
Član Povjerenstva 2:	izv. prof. dr. sc. Ratko Grbić
Naslov diplomskog rada:	Upravljanje uređajem metodom prepoznavanja gesti prstiju
Znanstvena grana diplomskog rada:	**Telekomunikacije i informatika (zn. polje elektrotehnika)**
Zadatak diplomskog rada:	U zadnjem desetljeću intenzivno se radi na razvoju različitih metoda upravljanja uređajima bežičnim putem, kao što su upravljanje govorom, kapacitivnim ekranima, Bluetooth protokolom, itd. Jedan od takvih načina je i upravljanje prepoznavanjem gesti prstiju korištenjem kamere. U vozilima ova metoda omogućuje upravljanje različitim uređajima kao što su radio uređaj, infotainment uređaj, zaslon projiciran na vjetrobransko staklo (engl. head-up display), itd. U okviru ovog diplomskog rada potrebno je napraviti algoritam zasnovan na dubokom učenju koji će korištenjem kamere prepoznati različite geste prstiju, kao što su npr. naredba klizanja u lijevo, desno, gore i dolje (engl. slide left, right, up and down), odabir/poništi odabir (engl select/deselect), listanje
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 1 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene od strane mentora:	12.09.2023.

Potvrda mentora o predaji konačne verzije rada:	*Mentor elektronički potpisao predaju konačne verzije.*
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 26.09.2023.

Ime i prezime studenta:	Lovro Čubaković
Studij:	Diplomski sveučilišni studij Elektrotehnika, smjer Komunikacije i informatika'
Mat. br. studenta, godina upisa:	D-1362, 07.10.2021.
Turnitin podudaranje [%]:	4

Ovom izjavom izjavljujem da je rad pod nazivom**: Upravljanje uređajem metodom prepoznavanja gesti prstiju**

izrađen pod vodstvom mentora prof. dr. sc. Marijan Herceg

i sumentora ,

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD

S razvojem društva neverbalna se komunikacija počela koristiti u mnogim područjima. Govor tijela, izraz lica, ton glasa, geste, dodir i pogled, čine preko 65% ukupne komunikacije. Zbog brzog razvoja u području dubokog učenja i tehnologije računalnog vida, korištenje gestikulacije ljudske ruke i prstiju pridonosi napretku u ostvarivanju učinkovitije interakcije između čovjeka i računala. Kako je ruka najfleksibilniji dio ljudskog tijela, gestama ruku može se ostvariti raznovrstan oblik komunikacije između ljudi i strojeva. Ljudske geste variraju od jednostavnijih do složenijih. Širok je spektar korištenja gestikulacije prstima za komunikaciju između ljudi i računala ili drugih elektroničkih uređaja kao što su pametni telefoni, robotika (kontroliranje robotske ruke gestama ruke i prstiju), automobilski informacijski sustavi (engl. *automobile infotainment system*), i slično. Prepoznavanje gesta može zamijeniti interakciju između čovjeka i računala putem dodirnih ili žičano upravljanih uređaja. Dinamička detekcija gesti prstiju pruža korisnicima intuitivno iskustvo upravljanja, smanjujući potrebu za složenim korisničkim sučeljima i skraćujući vrijeme potrebno za izvršavanje naredbi.

Multimedijskim sustavima kao što su sustavi za reproduciranje audio i video zapisa može se upravljati putem gesti ruku: pomicanjem prsta gore ili dolje, lijevo ili desno, pokazivanjem otvorenog dlana ili stisnute šake, i slično. Ove geste mogu biti povezane s funkcijama kao što su smanjivanje i pojačavanje glasnoće, promjena radio postaje, pretraživanje glazbe, navigacija, upravljanje mobitelom, i slično.

Jedan od učinkovitijih načina upravljanja multimedijskim sustavima za reproduciranje audio/video sadržaja gestikulacija je prstima koja se ostvaruje primjenom strojnog učenja i umjetne inteligencije. Konvolucijska neuronska mreža (engl. *Convolutional Neural Network, CNN*) vrsta je neuronske mreže koja se koristi u dubokom učenju i koja se najviše primjenjuje za analizu objekata na slikama i videozapisima. CNN je algoritam dubokog učenja koji pripada podskupu strojnog učenja. To je vrsta neuronske mreže koja je proširena verzija umjetne neuronske mreže (engl. *Artificial Neural Netowrk, ANN*), a koja se pretežno koristi za izvlačenje značajki iz matričnog skupa podataka nalik mreži kao što su skupovi vizualnih podataka, odnosno slike ili videozapisi.

U ovom diplomskom radu bit će prikazana izrada algoritma zasnovanog na dubokom učenju, koji će korištenjem kamere prepoznati različite geste prstiju: naredba klizanja lijevo, desno, gore, dolje (engl. *slide left, right, up and down*), odaberi/poništi (engl. *select/deselect*), listanje

gore/dolje (engl. *scroll up/down*), itd. U diplomskom radu proučene su postojeće baze slika/video signala gesti prstiju na kojima se neuronska mreža trenira i kreirana je vlastita baza slika. Nakon treniranja mreže, implementiran je istrenirani model neuronske mreže visoke preciznosti i točnosti. U ovome se radu gestama prstiju upravlja *Spotify* web aplikacijom koja je namijenjena za strujanje glazbenog sadržaja. Izrađeni algoritam je implementiran na *Raspberry Pi 3* ugradbenu platformu uz izmjerene performanse sustava.

Rad je strukturiran na sljedeći način. U uvodnom poglavlju opisan je predmet rada, odnosno postavljeni su ciljevi. Na kraju uvodnog poglavlja objašnjena je struktura rada. U drugom su poglavlju opisana postojeća rješenja koja se tiču problematike rješavanja detekcije gesti prstiju što uključuje potrebne algoritme i metode strojnog učenja, a navedene su prednosti i nedostatci svake metode. U trećem je poglavlju opisano radno okruženje koje se koristilo za izradu i treniranje neuronske mreže i sve potrebne biblioteke te alati za treniranje. Opisane su geste koje će se koristiti za upravljanje aplikacijom i kreirana je baza video sekvenci za treniranje neuronske mreže. Dana je arhitektura modela neuronske mreže koja se koristila za treniranje i opisan je sam proces treniranja mreže. U četvrtom poglavlju napravljena je evaluacija rada neuronske mreže nad testnom skupu baze podataka *20BN-Jester* i u realnim uvjetima na deset osoba upotrebom aplikacije napisane u programskom jeziku *Python*. U petom poglavlju opisana je realizacija web aplikacije koja omogućuje upravljanje *Spotify* servisom za strujanje glazbe koristeći gestikulaciju prstima. U šestom poglavlju opisana je implementacija rješenja na ugradbenu platformu *Raspberry Pi 3*. Na kraju rada dan je zaključak.

2. PREGLED POSTOJEĆIH RJEŠENJA ZA DETEKCIJU GESTI PRSTIJU

Postoji nekoliko načina komunikacije između čovjeka i računala. Govorna naredba, naredba izdana gestikulacijom ili senzorima koji prate pokret tijela i sučelje između mozga i računala (engl. *brain-computer interface*), samo su neke od njih. Ljudske se geste smatraju jednim od efikasnijih načina komunikacije čovjeka i računala. One mogu zamijeniti uporabu miša, tipkovnice, džojstika kao i raznih dodirnih podloga za navigaciju. Geste prstiju trebaju biti prepoznate u stvarnom vremenu bez prevelikog opterećivanja računalnog hardvera. U tu svrhu razvijeni su i unaprijeđeni brojni algoritamski okviri i metode temeljene na različitim tehnologijama.

Detekcija gesta prstiju često se bazira na senzorima pokreta [1], ugrađenih u korisničke rukavice. Senzori pokreta mogu biti integrirani i u pametne telefone u koje se ugrađuju akcelerometri i žiroskopi zbog praćenja brzine pokreta ruke kao i orijentacije ruke tijekom samog pokreta. Geste prstiju detektiraju se i korištenjem tehnika baziranih na radio signalima [2] u kojima je spremljena informacija o gestama prstiju. U tom slučaju radar s frekvencijskim moduliranim kontinuiranim valovima (engl. *frequency-modulated continuos-wave, FMCW*) odašilje kontinuirani signal koji periodički mijenja frekvenciju. Signal se emitira prema ruci koja izvodi gestu. Emitirani signal odbijen od ruke ili prsta reflektira se natrag prema FMCW radaru te se iz promjene frekvencije emitiranog signala može detektirati o kojoj je gesti riječ. Ultrazvučna tehnika [3] za detekciju gesti prstiju, još je jedna od tehnologija za detektiranje gesti prstiju. U ultrazvučnoj tehnici, zvučnici i mikrofoni koriste se kao ultrazvučni ulazno/izlazni uređaji. Za detekciju se može koristiti i Dopplerov pomak ultrazvučnih valova reflektiranih od ljudskog tijela koje se kreće. Tijekom izvođenja geste u blizini izvora ultrazvuka (zvučnika), mahanje rukom ili prstima, uzrokuje pomak u frekvenciji zvuka kojeg detektira mikrofon. Uz pomoć karakterističnih pomaka može se u frekvenciji odrediti je li mahanje ruke ili prstiju bilo u smjeru mikrofona ili u smjeru od mikrofona.

Najzastupljenije metode detekcije ruku i prstiju temelje se na vizaulizaciji (engl. *vision-based*) ljudske ruke i prstiju. Znanstveno područje koje obuhvaća takve metode naziva se računalni vid (engl. *computer-vision*). Druga metoda, koja je zastupljenija od računalnog vida, a koja će se obraditi u ovom poglavlju, u vidu pregleda postojećih rješenja za detekciju prstiju, metoda je primjene algoritama strojnog učenja. Točnije, metoda dubokog učenja koja obuhvaća treniranje dubokih neuronskih mreža za dinamičku detekciju prstiju iz video sekvenci. Kako bi neuronska

mreža zapamtila raspored prstiju tijekom gestikulacije u video sekvenci koristi se najzastupljenija tehnika računalnog vida koja se temelji na CNN mrežama.

U radu [4] predložen je sustav prepoznavanja gesti prstiju koji se sastoji od snimanja videa uživo s USB kamere nakon čega slijedi segmentacija tog videa na okvire te faze inicijalizacije, kalibracije i detekcije pokreta. Uz ovakav sustav, korisnici mogu izdavati naredbe pokretima ruku koji se snimaju uživo korištenjem USB kamere i koje se obrađuju u stvarnom vremenu. Za svaku gestu ruku korišteno je 750 slika ruke i prstiju u istom položaju u svrhu treniranja. Sveukupno je moguće izdavati naredbe sa šest različitih gesti ruku. Arhitektura definirane CNN mreže u ovome radu sastoji se od 262 144 parametara. Ulaz u mrežu je slika u sivom tonu (engl. *grayscale*) rezolucije 128x128x1 piksela, dok je izlaz naziv geste koja se detektira. Istrenirani model postigao je točnost od 97,40% na testnom skupu. Prednosti ovakvog sustava su kratko vrijeme treniranja neuronske mreže i točna stvarno vremenska detekcija različitih gesti ruku. Nedostatci koji se pojavljuju u slici kod zahtjevnije pozadine su smetnje identificirane šumom. Smetnje nastaju zbog naglog osvjetljenja ili bljeskajućeg svjetla pri čemu se javljaju komplikacije prilikom detekcije gesti ruku. U radu [5] predložen je sličan mehanizam prepoznavanja gesti ruku kao i u radu [4] koji se temelji na istreniranome modelu CNN mreže. Rješenje koje predlaže rad [5] namijenjeno je kao komunikacijski alat za gluhe osobe i osobe s poremećajem iz spektra autizma. U radu [5] implementirano je pet gesti ruku. Istrenirani model je kroz 30 epoha postigao točnost od 99,13%. Ulaz u CNN mrežu u ovome rješenju su slike istoga formata kao i ulaz u rješenju [4], a izlaz je jedna od pet gesti prstiju koje model klasificira. Najveća prednost ovoga rješenja izuzetno je visoka točnost modela na testnom skupu. Nedostatak ovog rješenja smanjena je točnost prepoznavanja gesti ruku u situacijama u kojim se pojavljuje više od jedne ruke u kadru kamere. Postoji mogućnost da model neuronske mreže u tim situacijama detektira krivu gestu. U radu [6] uveden je novi koncept dubokog učenja za prepoznavanje dinamičkih gesti prstiju koji se temelji na kombinaciji trodimenzionalne konvolucijske neuronske mreže (3D-CNN) i dugotrajne kratkoročne memorije (engl. *Long Short-Term Memory*, *LSTM*). Arhitektura predložena u [6] izvlači prostorno-vremenske informacije iz ulaznih video sekvenci. Predložena arhitektura ovoga modela ima 3.7 milijuna parametara za trening. Ulaz u model su video sekvence koje sadrže 30 okvira rezolucije 112x112x3 piksela, a izlaz iz mreže jedna je od gesti. Implementirano je 15 gesti iz baze video sekvenci *20BN-Jester* [7]. Model je treniran na 2000 video sekvenci po svakoj klasi i postigao je točnost od 99% na trening skupu te 97% na testnom skupu podataka. Najveća prednost ovoga rješenja je dinamička detekcija gesti prstiju u stvarnom vremenu i vrlo učinkovita stopa treniranja po svakoj epohi koja osigurava visoku točnost modela. Nedostatak ovoga rješenja je

pojava pretreniranosti (engl. *overfitting*). U radu [8] predložen je model za prepoznavanje gesti ruku i prstiju razvijen za identifikaciju skupa video sekvenci gesti ruku. Geste ruku prepoznaju se iz dvodimenzionalnih RGB video zapisa, koristeći informacije koje se nalaze u videozapisu te prostorno-vremenske parametre uzastopnih okvira. U ovom rješenju uvedena je tehnika prijenosa učenja (engl. *transfer learning*) za fino podešavanje (engl. *fine-tuning*) arhitekture modela na relevantnim gestama ruku. Od 27 klasa iz baze video sekvenci *20BN-Jester*, u rješenju [8] izabrano je 5 klasa, i za svaku je klasu odabrano 2812 video sekvenci za trening. Sveukupno je prikupljeno 113 410 okvira za trening i 6784 okvira za validaciju. Ulaz u 3D-CNN LSTM mrežu u ovome rješenju su 32 okvira rezolucije 64x63x3 piksela, a izlaz je naziv geste koja se klasificira. Modelom je postignuta točnost od 93,95% kroz 64 epoha treniranja. Glavna prednost ovoga rada je arhitektura 3D-CNN LSTM modela male veličine. Model se sastoji od malog broja parametara i postiže visoku točnost kroz mali broj epoha treniranja. Model omogućuje rad u stvarnom vremenu i lakšu integraciju daljnjih klasa gesti prstiju. Zbog arhitekture modela male veličine, ovakav model moguće je implementirati u mobitele ili prijenosna računala kako bi se vršila detekcija gesti prstiju. U radu [9] predložena je nova mreža dubokog učenja (engl. *deep learning*) za prepoznavanje gesti ruku i prstiju koja je nazvana kao neuronska mreža za kratkoročno uzorkovanje (engl. *Short-Term Sampling Neural Network*, *STSNN*). Mreža zajedno integrira module za grupno kratkoročno uzorkovanje video sekvenci. Moduli podrazumijevaju spajanje značajki dvaju okvira (RGB okvira i *optical flow* okvira) dobivenih predobradom slike, CNN mrežu za izvlačenje prostornih značajki i LSTM jedinice za pamćenje vremenskih značajki. Ulaz u ovu mrežu je 37 video okvira za svaku video sekvencu. Svaki okvir iz tog videozapisa rezolucije je 176x100x3 piksela. Za treniranje modela izabrano je svih 27 klasa iz baze video sekvenci *20BN-Jester*. Model je treniran i na *Nvidia* bazi video sekvenci iz koje je izabrano 25 klasa. Predloženi STSNN model postigao je točnost od 95,73% na testnom skupu *20BN-Jester* baze video sekvenci, a na testnom je skupu *Nvidia* baze video sekvenci postigao točnost od 85,13%. Glavna prednost ovog rješenja je klasifikacija svih gesti iz bazi *20BN-Jester* i *Nvidia* te evaluacija istreniranog modela na tim bazama. Dana je i matrica zabune za svaki model i uspoređena je točnost ovoga modela s drugim modelima kao što su *MobileNet* [10], *ResNet* [11] te ostali modeli CNN mreža s kojima se može usporediti točnost modela definiranog u rješenju [9]. Ovo rješenje implementira i situacije u kojima korisnik gestu izvodi udaljeniji od USB kamere više od 50 centimetara kako bi riješio problem detekcije u slučaju kada je osoba koja izvodi gestu udaljenija od kamere nego što je to slučaj u video sekvencama iz baze *20BN-Jester*. Rješenje [9] radi samo u slučaju kada

korisnik izvodi gestu s jednom rukom bez drugih objekata ili kada je samo jedna osoba u kadru kamere. Pojavom drugih osoba u kadru kamere dolazi do pogrešne detekcije geste.

3. OPIS VLASTITOG RJEŠENJA ZA DETEKCIJU GESTI PRSTIJU KORIŠTENJEM WEB KAMERE

U ovom poglavlju opisan je izgled neuronske mreže kojoj je zadatak prepoznavanje gesti prstiju ljudske ruke. Na slici 3.1. prikazan je dijagram tijeka vlastitog rješenja. Ulaz u ovakav sustav rješenja video je sekvenca fiksnog broja okvira u sekundi (engl. *frames per second, FPS*), koja je uhvaćena s WEB kamere u trenutku kada osoba izvodi određenu gestu pred kamerom. Ta video sekvenca predaje se istreniranom modelu dugoročno rekurentne konvolucijske mreže (engl. *Long-term Recurrent Convolutional Network, LRCN*) koja vrši detekciju geste prstiju. Ako je detekcija geste uspješna, uz pomoć WEB aplikacije za upravljanje *Spotify* glazbenim servisom, izvodi se određena radnja nad *Spotify* aplikacijom.

Slika 3.1. Dijagram tijeka vlastitog rješenja za detekciju gesti prstiju korištenjem WEB kamere

Prvi korak u izradi rješenja odabir je gesti prstiju kojima će se upravljati *Spotify* aplikacijom. Odabir gesti ovisi o specifičnosti samoga sustava i primjeni tih gesti. Ključno je da se odaberu geste koje su relevantne za ovo rješenje i s kojima korisnik neće imati poteškoća prilikom njihovog izvođenja.

Nakon odabira gesti, u izradi sustava slijedi kreiranje baze podataka te priprema skupa podataka za treniranje, validaciju i testiranje neuronske mreže. Potrebno je prikupiti dovoljan broj uzoraka skupa podataka gesti prstiju kako bi se neuronska mreža mogla istrenirati do zadovoljavajuće preciznosti i točnosti. Baza podataka mora sadržavati različite video sekvence gestikulacije prstiju koje će osigurati skoro sve funkcionalnosti za kontrolu *Spotify* aplikacije. Važno je osigurati dovoljan broj podataka za svaku definiranu gestu kako bi se postigla visoka točnost modela neuronske mreže na testnom skupu podataka i u realnim uvjetima.

Nakon prikupljanja skupa podataka, potrebno je definirati arhitekturu neuronske mreže. U ovome slučaju izabrana je kombinacija konvolucijske neuronske mreže i rekurentne neuronske mreže (engl. *Recurrent Neural Network, RNN*). Arhitektura modela treba biti potpuno prilagođena za obradu ulaznog skupa podataka video sekvenci koje sadržavaju geste prstiju. Model neuronske mreže mora biti dovoljno složen kako bi mogao naučiti relevantne obrasce iz tih podataka.

Nakon kreiranja baze podataka slijedi treniranje neuronske mreže. Za treniranje korištena je tehnika nadziranog učenja (engl. *supervised learning*) [12], gdje se modeli uče korištenjem ulaznih i izlaznih skupova podataka. Skupovi podataka prikupljeni su eksperimentalno ili iz već postojećih baza podataka. Na taj se načim može klasificirati ulazne signale. Ulazi u mrežu uspoređuju se s očekivanim izlazima i u procesu treniranja prilagođavaju se težine (engl. *weights*) veza između neurona kako bi se minimizirala pogreška. Postoje dva modela nadziranog učenja: klasifikacija i regresija. Kod klasifikacije mapiranje se objekata sa slike ili videozapisa vrši u unaprijed definirane klase. Klasifikacija uključuje mapiranje podataka u diskretne oznake klasa. Kod regresije mapiranje se objekata vrši u vrijednosti i pokušava se pronaći aproksimacija funkcije s minimalnim odstupanjem pogreške. Glavni cilj regresije je procjena funkcije mapiranja na temelju ulaznih i izlaznih varijabli. Treniranje se provodi iterativnim postupkom uz unaprijed definirane nazive klasa, a cilj je da se postigne najviša moguća točnost i preciznost modela.

U zadnjem koraku potrebno je izraditi WEB aplikaciju kojom će se upravljati *Spotify* glazbenim servisom ovisno o izvedenoj gesti pred WEB kamerom. U aplikaciju je potrebno implementirati autentikaciju korisnika od kojeg se očekuje da posjeduje *Premium Spotify* račun. Osim autentikacije, potrebno je povezati svaku gestu koju detektira LRCN model s odgovarajućom radnjom u *Spotify* aplikaciji.

3.1. Podešavanje radnog okruženja i instalacija potrebnih biblioteka

Izrada vlastitog rješenja za prepoznavanje gesti prstiju realizirana je u programskom jeziku *Python* koristeći *Anaconda Navigator* razvojno okruženje te programski alat *Jupyter Notebook* unutar web sučelja na operacijskom sustavu *Microsoft Windows 10*. Od ključnih biblioteka za treniranje i evaluaciju neuronske mreže korištene su *TensorFlow* [13], *Keras* [14], *NumPy* [15], *Scikit-learn* [16] te *OpenCV* biblioteka [17]. Unutar *Anaconda Navigator* razvojnog okruženja instalirano je virtualno okruženje nazvano „*tensorflow*“ u kojemu su preuzete i instalirane sve potrebne biblioteke za izradu rješenja ovog rada. Uz *Jupyter Notebook* instalirani su i programski

alati VSC (engl. *Visual Studio Code*) i *Spyder* za lakše uređivanje programskog koda i vizualizaciju podataka tokom treniranja i evaluacije neuronske mreže.

3.1.1. *TensorFlow* biblioteka

TesnorFlow je biblioteka otvorenog koda (engl. *open source*) namijenjena za strojno učenje i umjetnu inteligenciju. Može se koristiti za obavljanje različitih zadataka, no najviše se koristi za izgradnju, treniranje i evaluaciju dubokih neuronskih mreža. *TensorFlow* operacije mogu se izvoditi na uređajima kao što su CPU (engl. *Central Processing Unit*), GPU (engl. *Graphics Processing* Unit) ili TPU (engl. *Tensor Processing Unit*). *TensorFlow* omogućuje kolekciju različitih slojeva (engl. *layers*), aktivacijskih funkcija (engl. *activation functions*), optimizatora i metoda za izradu različitih arhitektura neuronskih mreža. *TensorFlow* omogućuje korisnicima jednostavan način izrade modela neuronske mreže za treniranje i kasnije obavljanje složenih zadataka. Za izradu rješenja ovog diplomskog rada korištena je verzija 2.11.0. *TensorFlow* biblioteke.

3.1.2. *Keras* biblioteka

Keras biblioteka namijenjena je za duboko učenje i dizajn jednostavnih i brzih neuronskih mreža. Biblioteka omogućuje apstrakciju različitih slojeva te modela neuronske mreže i olakšava njihovu konfiguraciju i izgradnju. Slično kao i *TesnorFlow* biblioteka, omogućuje definiranje slojeva, aktivacijskih funkcija, optimizatora i ostalih metoda pomoću jednostavnih API-ja (engl. *Applicaton Programming Interface*). Korisnici mogu stvarati i prilagođavati slojeve, funkcije gubitaka (engl. *loss function*) i razne metrike evaluacije prema svojem modelu. *Keras* je interoperabilan s drugim bibliotekama za duboko učenje kao što su *TensorFlow* i *Theano* [18] i najčešće se koristi u kombinaciji s tim bibliotekama. *Keras* nudi i alate za praćenje te vizualizaciju napretka prilikom treniranja model što uključuje različite metrike performansi (gubitak i točnost prilikom treniranja i validacije). Za izradu rješenja korištena je ista verzija kao i verzija *TensorFlowa*, odnosno verzija 2.11.0.

3.1.3. *NumPy* biblioteka

NumPy biblioteka koristi se u *Python* programskom jeziku kao biblioteka za kompleksne znanstvene proračune. Nudi učinkovite alate koji omogućuju rad s matricama i višedimenzionalnim poljima i nizovima kao i izvođenje kompleksnih matematičkih operacija nad njima. Jedna je od temeljnih biblioteka kada je potrebno u rješenje implementirati znanstveni račun i kompleksne operacije nad skupom podataka. *NumPy* biblioteka vrlo je učinkovita za učitavanje,

spremanje ili manipulaciju nad podacima različitog formata, što uključuje i *CSV* datoteke kao i tekstualne, binarne ili druge datoteke. Mnogi drugi alati i biblioteke za manipulaciju nad većim skupom podataka temelje se na *NumPy* biblioteci. U ovome radu korištena je verzija *NumPy* biblioteke 1.23.5.

3.1.4. *Scikit-learn* biblioteka

Scikit-learn biblioteka, poznatija kao *sklearn*, još je jedna od biblioteka koja se koristi za strojno učenje. Sadrži veliki skup algoritama za nadzirano (engl. *supervised*) i nenadzirano (engl. *unsupervised*) strojno učenje. Što se tiče nadziranog učenja, biblioteka pruža sve potrebne algoritme za klasifikaciju, regresiju i različite metode za klasifikaciju ili regresiju. Neke od tih metoda su slučajne šume (engl. *random forests*), pronalaženje najbližeg *k*-susjeda, pojačavanje gradijenta, itd. S druge strane, za nenadzirano učenje, biblioteka sadrži algoritme potrebne za grupiranje sličnih podataka ili detekciju anomalija u skupu podataka, smanjenje dimenzionalnosti, itd. *Sklearn* nudi i mogućnost unakrsne validacije (engl. *cross validation*) tijekom treniranja, podešavanje hiperparametara (engl. *hyperparameter tuning*) te ostalih metoda kako bi se postigla najbolja moguća optimalnost konfiguracije modela. Nudi i alate za obradu podataka kao što su normalizacija, skaliranje, izvlačenje značajki te pretvaranje varijabli. Navedeni alati koriste se za pripremu skupa podataka prije treniranja modela i za izvlačenje značajki. Za potrebe rješenja ovog rada instalirana je verzija 1.2.0. *Scikit-learn* biblioteke.

3.1.5. *OpenCV* biblioteka

OpenCV (engl. *Open Source Computer Vision Library*) je biblioteka otvorenog koda koja se koristi za potrebe računalnog vida (engl. *computer vision*) i strojnog učenja. Biblioteka sadrži više od 2500 optimiziranih algoritama koji služe za otkrivanje i prepoznavanje ljudskog lica, identifikaciju različitih objekata, klasifikaciju ljudskih aktivnosti tijekom video sekvenci, praćenje pokretnih objekata u videozapisima, itd. Također nudi podršku *Python* programskom jeziku u vidu rada s kamerama i pri obradi učitanog videa s kamere (učitavanje i manipulacija video okvira koji se dobivaju u stvarnom vremenu pomoću USB kamere). U okviru ovog rada korištena je verzija 4.7.0.68. *OpenCV* biblioteke.

3.2. Odabir gesti prstiju za upravljanje aplikacijom

Za potrebe ovog rada odabrano je devet gesti prstiju kojima će se kontrolirati *Spotify* aplikacija. Pri odabiru gesti vodilo se računa da se obuhvate sve potrebne radnje kako bi korisnik na

jednostavan način mogao upravljati *Spotify* glazbenim servisom. Temeljni pokreti kojima će se izvoditi geste su pokreti klizanja prstiju (engl. *swipe motions*).

Naredbama klizanja prstiju lijevo ili desno (engl. *swipe left/right*) predviđeno je mijenjanje pjesama u trenutnoj listi pjesama koje se reproduciraju, odnosno odabir sljedeće pjesme (klizanje desno) ili vraćanje na prethodnu (klizanje lijevo). Klizanjem prstima gore (engl. *swipe up)* postavlja se glasnoća zvuka na 100%, a klizanjem prstima dolje (engl. *swipe down*) postavlja se glasnoća na 50%. Za pokretanje ili zaustavljanje (engl. *play/pause*) reproduciranja glazbenog sadržaja korištena je gesta kojom se kameri pokazuje otvoreni dlan. Za smanjivanje glasnoće zvuka po 5% koristit će gesta nazvana „smanji“, a za pojačavanje glasnoće zvuka po 5% gesta „pojačaj“. Gesta „smanji“ izvodi se na način da se palac i kažiprst iz odvojenog položaja spoje zajedno, dok se kod geste „pojačaj“ iz spojenog položaja, plac i kažiprst odvoje. Za dodavanje pjesme u omiljeni sadržaj (engl. *liked songs*) *Spotify* aplikacije zamišljeno je da se koristi gesta koja simulira *like,* kod koje je palac ispružen i pokazuje prema gore, a za odbacivanje iste pjesme iz omiljenog sadržaja koristi se gesta „*dislike*“ kod koje se ispruženi palac pokazuje prema dolje. Kako bi se povećala točnost modela odlučeno je da se uz ovih 9 gesti kojima će se kontrolirati aplikacija, uvede i gesta nazvana „nema geste“ koja uči model da ne odrađuje nikakvu radnju kada je korisnik statičan pred kamerom, odnosno kada korisnik ne izvodi geste pred kamerom.

3.3. Kreiranje baze video signala za treniranje neuronske mreže

Javno su dostupni različiti skupovi video sekvenci za izvođenje gesti prstima. Za implementaciju gesti koje su navedene u potpoglavlju 3.2. koristila se baza *20BN-Jester*. Taj skup podataka sadrži preko 148 092 kratkih video sekvenci od kojih svaka traje tri sekunde. Svaki video sadrži 37 okvira koji su nakon preuzimanja baze podataka automatski raspodijeljeni po klasama. Baza ima sveukupno 27 klasa od kojih je u ovome radu korišteno 10 klasa navedenih u potpoglavlju 3.2. Skup video sekvenci *20BN-Jester* preuzet je s *Kaggle* web stranice [19] namijenjene za objavljivanje i uređivanje baza podataka potrebnih za strojno učenje.

Nakon preuzimanja cijele *20BN-Jester* baze bilo je potrebno izdvojiti klase, odnosno geste, koje se koriste za rješenje ovog diplomskog rada te ih spremiti lokano u datoteku na osobnom računalu. Iz svake klase izdvojeno je 700 video sekvenci po 37 okvira, što sveukupno čini 7000 video sekvenci, odnosno 259 000 okvira. Izdvajanja 700 video sekvenci za svih 10 klasa napravljeno je pomoću jednostavne skripte napisane u *Pythonu* koja služi za pretraživanje baze te izdvajanje određenog broja videa po klasi.

U tablici 3.1. navede su klase koje su se koristile za treniranje neuronske mreže, navedeni su njezini nazivi te opis radnje koja se ostvaruje svakom od prikazanih gesta.

Tablica 3.1. Klase koje su korištene za treniranje neuronske mreže

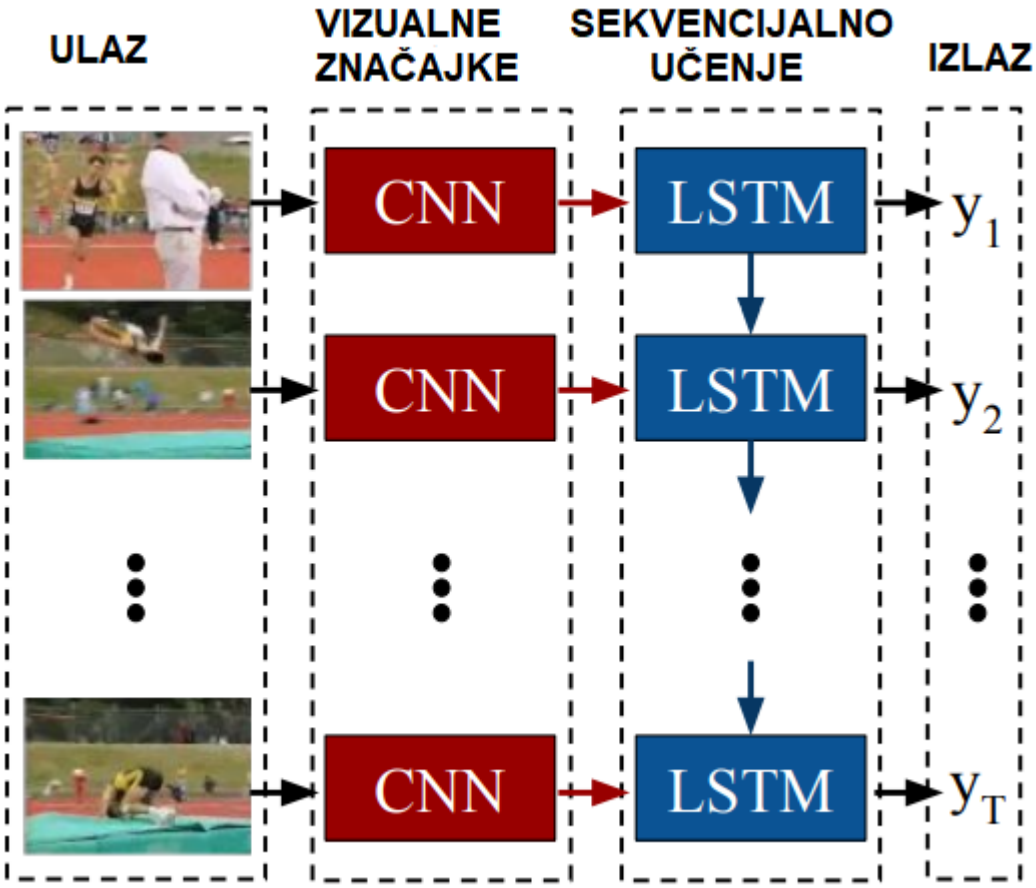
Naziv klase	Opis radnje koja se ostvaruje gestom
Dislike	Odbacivanje pjesme iz omiljenog sadržaja
Like	Dodavanje pjesme u omiljeni sadržaj
Nema geste	Korisnik je statičan pred kamerom, ne prikazuje ni jednu od ostalih 9 gesti
Pojačaj	Pojačavanje glasnoće zvuka za 5%
Smanji	Smanjivanje glasnoće zvuka za 5%
Swipe desno	Prebacivanje na sljedeću pjesmu u *playlisti*
Swipe dolje	Postavljanje glasnoće zvuka na 50%
Swipe gore	Postavljanje glasnoće zvuka na 100%
Swipe lijevo	Prebacivanje na prethodnu pjesmu u *playlisti*
Zaustavi/pokreni	Zaustavljanje/pokretanje reproduciranja trenutno odabrane pjesme

U prilogu P.3.1. prikazano je nekoliko video okvira prilikom izvođenja svake od gesti kako bi korisnik znao pravilno izvesti gestu i kako bi se na taj način povećala točnost prilikom detekcije gesti. Za kompletno shvaćanje izvođenja gesti potrebno je proučiti bazu podataka *20BN-Jester*.

3.4. Opis LRCN (engl. *Long-term Recurrent Convolutional Network*) mreže

U ovom radu koristila se 3D-CNN mreža i LSTM mreža. Kombinacija 3D-CNN i LSTM mreže naziva se LRCN mreža [20]. Ulaz LRCN mreže je video sekvenca promjenjive duljine koja dolazi na ulaz u 3D-CNN mrežu. Izlazi iz 3D-CNN mreže predaju se velikom broju modela rekurentnih sekvenci (LSTM-ovi) koji na kraju omogućuju predviđanje određene radnje iz video sekvence promjenjive duljine.

LRCN mreža funkcionira na način da svaki vizualni ulaz x_t (okvir iz video sekvence) prolazi kroz transformaciju značajki $\emptyset_V(.)$ s parametrima V, gdje se u većini slučajeva transformacija značajki $\emptyset_V(.)$ odvija u CNN-u, kako bi se proizvela vektorska reprezentacija fiksne duljine $\emptyset_V(x_t)$. Izlazi $\emptyset_V$ prosljeđuju se u modul učenja rekurentne sekvence, u ovome slučaju LSTM mreži. Na slici 3.2 prikazan je poopćeni dijagram LRCN mreže.



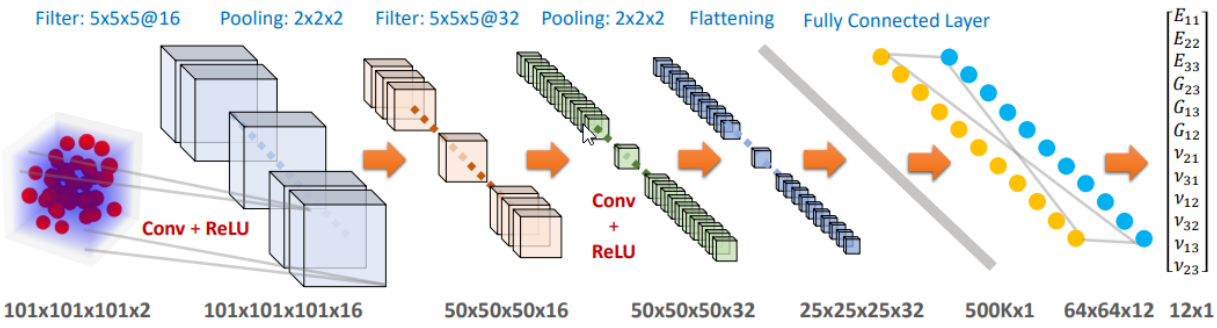
Slika 3.2. Dijagram LRCN mreže koja uz prostorne značajke sa slike pamti i vremenski kontinuirane radnje iz sekvence video okvira [20]

3.4.1. Značajke CNN-a

Konvolucijska neuronska mreža je vrsta modela dubokog učenja za obradu podataka koji imaju uzorak mreže (engl. *grid pattern*), kao što su slike. CNN mreže dizajnirane su za automatsko i adaptivno učenje prostornih hijerarhija značajki, od niskih značajki pa do uzoraka visoke razine. CNN mreža obično se sastoji od tri vrste slojeva ili građevnih blokova (engl. *building blocks*): konvolucija (engl. *convolution*), udruživanje (engl. *pooling*) te potpuno povezani sloj (engl. *fully connected layer).* Prva dva sloja, konvolucijski slojevi te slojevi udruživanja, izvode izvlačenje značajki iz slika, dok treći, potpuno povezani sloj, preslikava izdvojene značajke iz slika u konačni izlaz. Ključnu ulogu u CNN-u ima konvolucijski sloj koji se sastoji od niza matematičkih operacija, odnosno konvolucije. U digitalnim slikama, vrijednosti piksela svake slike pohranjene su u dvodimenzionalnoj (2D) slici, tj. nizu brojeva. Filter koji se koristi za izvlačenje značajki iz slika naziva se kernel i primjenjuje se na svakoj poziciji slike. Uporaba kernela CNN čini vrlo

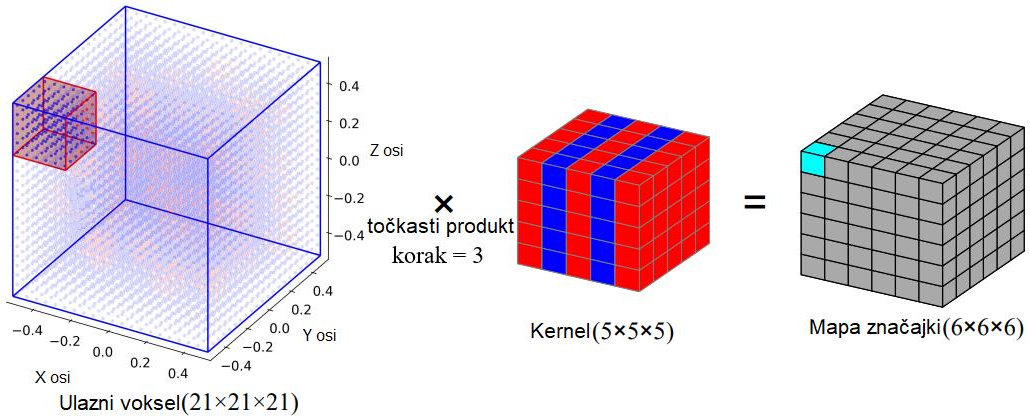
učinkovitim za obradu slike, jer se značajka može pojaviti bilo gdje na slici. Kako jedan sloj prenosi svoj izlaz na ulaz sljedećeg sloja, izvučene značajke mogu hijerarhijski i progresivno postati složenije. Postupak optimizacije parametara kao što su koeficijenti kernela, naziva se trening. Optimizacija se primjenjuje kako bi se minimizirala razlika između izlaza i temeljnih istinitih oznaka (engl. *ground truth labels*). Neki od algoritama optimizacije su povratno širenje (engl. *back propagation)* te gradijentno opadanje (engl *gradient descent*).

Fokus u ovome radu je na 3D-CNN mreži koja će se koristiti u kombinaciji s LSTM mrežom. Razlika između trodimenzionalne (3D) i dvodimenzionalne (2D) konvolucijske neuronske mreže je ta, što se kod dvodimenzionalne mreže kerneli pomiču samo u dvije dimenzije (npr. visina i širina slike, odnosno X i Y os), a kod trodimenzionalne mreže kerneli se pomiču u tri smjera, osim visine i širine, dodaje se i dubina same slike, odnosno Z os. Arhitektura jedne 3D-CNN mreže u primjeni homogenizacije heterogenih elemenata [21] dana je na slici 3.3.



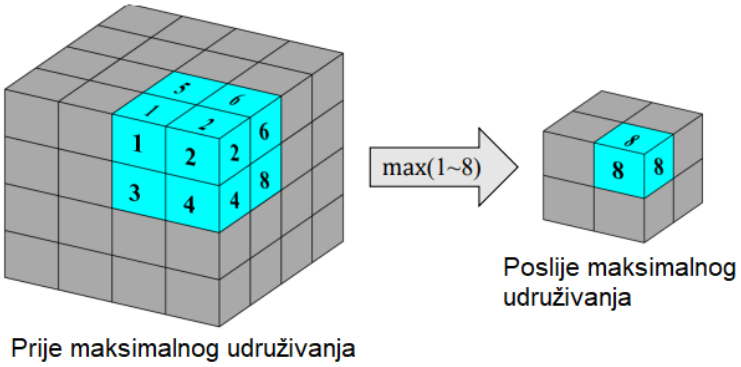
Slika 3.3. Primjer arhitekture 3D-CNN mreže za homogenizaciju heterogenih elemenata [21]

Pomoću slike 3.4. objašnjen je proces konvolucije unutar 3D-CNN modela. Recimo da je na ulazu trodimenzionalni prostor polja (engl. *voxel*) veličine $(21x21x21)$. 3D filter, odnosno kernel veličine $(5x5x5)$ i koraka (engl. *stride*) 3, prelazi preko ulaznog trodimenzionalnog polja i primjenjuje operacije konvolucije nad ulazom [21]. Time nastaje točkasti produkt tenzora koji predstavlja prostor značajki, a naziva se i mapa značajki (engl. *feature map*). Težine (engl. *weights*) i pomak (engl. *biases*) svakog kernela treniraju se za izdavanja istaknutih značajki iz ulaza. Korak, proširenje (engl. *padding*) i veličina kernela neki su od uobičajenih hiperparametara koji definiraju konvolucijske operacije. Korak označava veličinu za koju se kernel pomiče svaki put kada prelazi preko trodimenzionalnog prostora polja. Npr. korak od 1 znači da kernel prelazi preko slike piksel po piksel. Kako bi se očuvala prostorna veličina izlaza, potrebno je ispuniti ulazne vrijednosti polja nulama. Na taj način će ulaz i izlaz na slici 3.4. imati identične veličine $(21x21x21)$.



Slika 3.4. Proces konvolucije unutar 3D-CNN modela [21]

Slojevi udruživanja dodaju se između uzastopnih konvolucijskih slojeva u CNN-u. Sloj udruživanja progresivno smanjuje prostornu veličinu podataka smanjivanjem uzorkovanja vrijednosti voksela. Operacije udruživanja služe za izračunavanje maksimalne ili prosječne vrijednosti unutar volumena. Slika 3.5. prikazuje kako operacija maksimalnog udruživanja radi s veličinom volumena $(2x2x2)$.



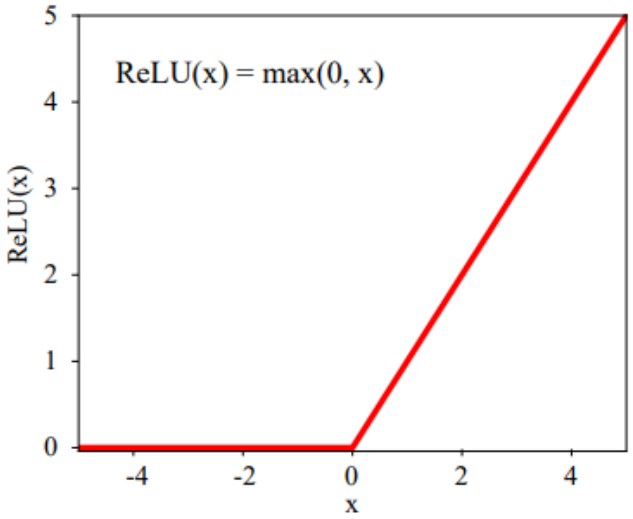
Slika 3.5. Operacija maksimalnog udruživanja [21]

Aktivacijski slojevi koriste se za uvođenje nelinearnosti u CNN mrežu. Funkcioniraju na način da uzmu jedan broj (vrijednost piksela) i izvedu određenu fiksnu matematičku funkciju nad njime. Neke od aktivacijskih funkcija koje se koriste u strojnom učenju su ReLU (engl. *Rectified Linear Unit*) $f(x) = \max(0, x)$, funkcija *sigmoid* $f(x) = 1/(1 + e^{-x})$ te tangens hiperbolni $f(x) = \tanh(x)$. Među ovim nelinearnim funkcijama, preferira se aktivacijska funkcija ReLU,

koja je prikazana na slici 3.6. Funkcija se koristi zbog svoje jednostavne aritmetičke operacije i izvrsnog svojstva konvergencije na SGD (engl. *Stochastic Gradient Descent)* [19] algoritmu u usporedbi sa *sigmoid* funkcijom ili *tanh* funkcijom. Matematički izraz izlazne vrijednosti γ na poziciji (x, y, z) na j-toj mapi značajki u i-tom trodimenzionalnom konvolucijskom sloju [21] može se napisati kao:

$$\gamma_{j,xyz}^{(i)} = ReLU\left(b_j^{(i)} + \sum_{m=1}^{M^{(i-1)}} \sum_{p=0}^{P^{(i)}-1} \sum_{q=0}^{Q^{(i)}-1} \sum_{r=0}^{R^{(i)}-1} w_{jm,pqr}^{(i)} \gamma_{m,(x+p)(y+q)(z+r)}^{(i-1)}\right) \tag{3-1}$$

gdje $ReLU(.)$ predstavlja ReLU funkciju nad elementima; $b_j^{(i)}$ je zajednički pomak za j-tu mapu značajki; $w_{jm,pqr}^{(i)}$ je (p, q, r)-ta vrijednost trodimenzionalnog kernela za j-tu mapu na i-tom sloju povezanom s m-tom mapom značajki u $(i-1)$-tom sloju; $M^{(i-1)}$ predstavlja broj mapi značajki na $(i-1)$-tom sloju; $P^{(i)}$, $Q^{(i)}$ te $R^{(i)}$ predstavljaju veličinu trodimenzionalnog kernela na i-tom sloju.

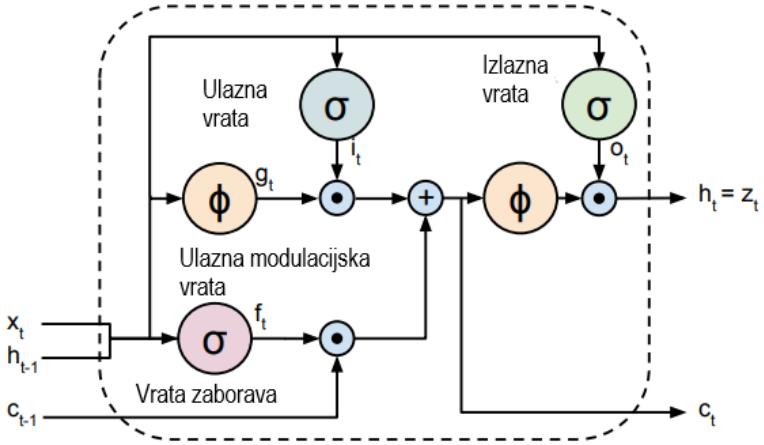


Slika 3.6. Prikaz ReLU funkcije [21]

3.4.2. LSTM jedinica

Struktura LSTM jedinice sastoji se od ulazno/izlaznih te zaboravi/pamti ćelijskih logičkih vrata koja kontroliraju proces učenja kao što je prikazano na slici 3.7. Sva logička vrata se podešavaju pomoću sigmoidnih funkcija za kontrolu „otvaranja“ i „zatvaranja“ tijekom procesa učenja. Dugoročno pamćenje u LSTM-u poznato je kao stanje ćelije. Stanje ćelije kontrolira

podatke koji se pohranjuju unutar LSTM ćelije iz prethodnih intervala. Ako je izlazno stanje vrata zaborava 0, govori vratima ćelije da zaborave informaciju, a ako je 1, govori vratima ćelije da je zadrže u stanju ćelije.



Slika 3.7. Primjer LSTM jedinice [20]

Jednadžbe od (3-2) do (3-7) [20] opisuju LSTM jedinice.

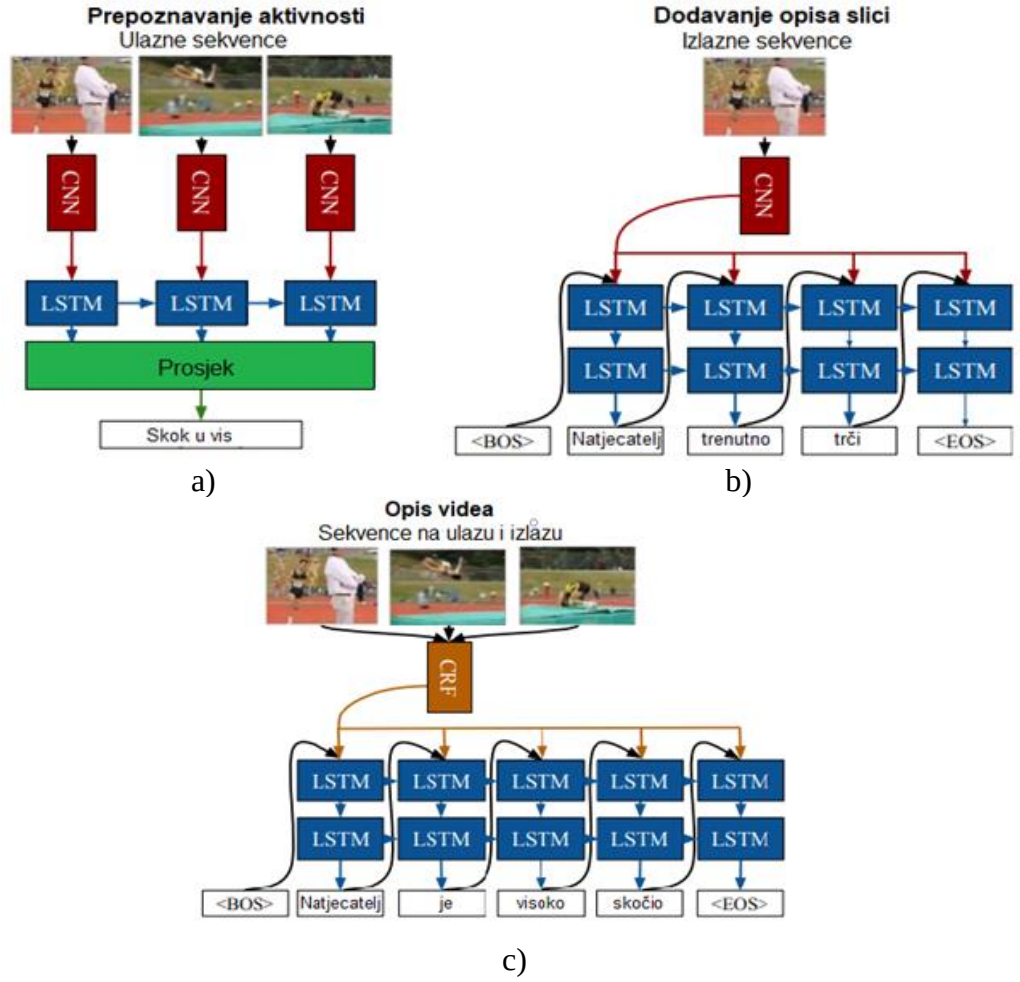
$$i_t = \sigma(x_t w_{xi} + h_{t-1} w_{hi} + c_{t-1} w_{ci} + w_{ibais}), \quad (3\text{-}2)$$
$$f_t = \sigma(x_t w_{xf} + h_{t-1} w_{hf} + c_{t-1} w_{cf} + w_{fbais}), \quad (3\text{-}3)$$
$$z_t = tanh(x_t w_{xz} + h_{t-1} w_{hz} + w_{zbais}), \quad (3\text{-}4)$$
$$c_t = z_t \otimes i_t + c_{t-1} \otimes f_t\,, \quad (3\text{-}5)$$
$$o_t = \sigma(x_t w_{xo} + h_{t-1} w_{ho} + c_{t-1} w_{co} + w_{obais}), \quad (3\text{-}6)$$
$$h_t = o_t + \tanh(c_t), \quad (3\text{-}7)$$

gdje izraz i_t predstavlja ulazna vrata, f_t predstavlja vrata koja zaboravljaju informaciju, o_t su izlazna vrata, a z_t su vrata ćelije. Izrazi c_t i h_t predstavljaju izlazne memorijske aktivacijske funkcije pri vremenu t. Izrazi (3-3), (3-4), (3-6) i (3-7) predstavljaju jednadžbe za izračun vrijednosti vrata zaborava, izlaznih vrata te skrivenog stanja (engl. *hidden state*).

3.4.3. LRCN model

Na slici 3.8. prikazani su primjeri primjene LRCN modela (prepoznavanje aktivnosti, dodavanje opisa slici i dodavanje opisa videozapisu) [20]. BOS (engl. *beginning of a sequence*) predstavlja početni okvir video sekvence, a EOS (engl. *ending of a sequence*) predstavlja završni okvir video sekvence. Postoje tri glavne primjene LRCN modela u ovisnosti o ulazu i izlazu. Prva primjena odnosi se na prepoznavanje aktivnosti koja je prikazana videozapisom s ciljem

predviđanja jedne oznake poput skakanja ili trčanja (slika 3.8., a). Druga primjena je dodavanje opisa jednom okviru iz videozapisa kako bi se opisala radnja prikazana okvirom (slika 3.8., b). Kada su dodani opisi svakom okviru videozapisa prelazi se na treću i glavnu primjenu. Treća primjena je dodavanje opisa cijelom videozapisu koji se sastoji od više okvira, u svrhu opisivanja radnje ili scene koja je prikazana videozapisom (slika 3.8., c).

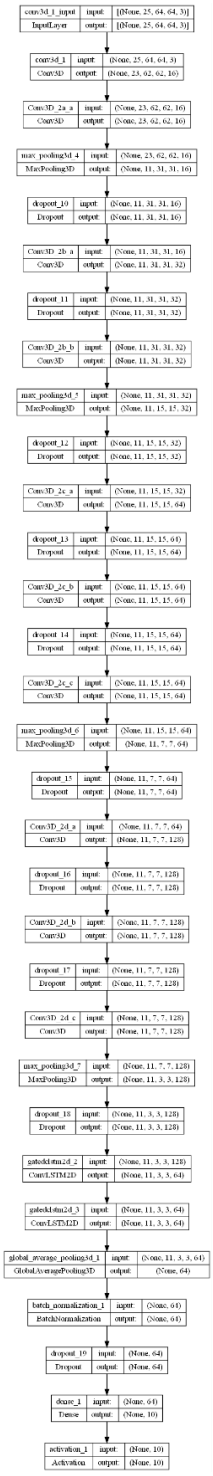


Slika 3.8. Primjeri primjene LRCN modela: a) prepoznavanje određenih radnji, b) dodavanja opisa slike, c) dodavanje opisa video sekvence [20]

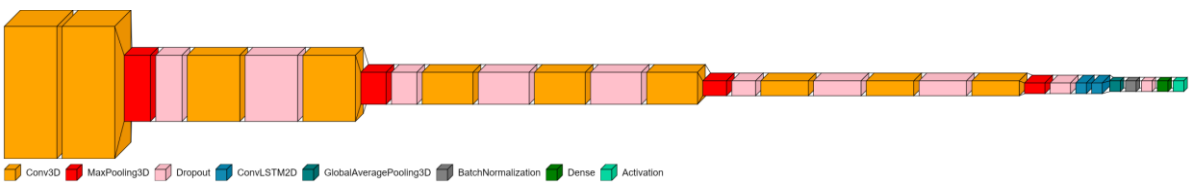
3.5. Arhitektura dugoročne rekurentne konvolucijske mreže (LRCN mreža)

Na slici 3.9. prikazana je arhitektura modela neuronske mreže koja je kreirana kao rješenje ovog rada. Na slici su prikazane dimenzije ulaza u svaki sloj mreže te dimenzije izlaza iz svakog sloja što je i vidljivo danim brojevima u zagradama. Ulaz u konvolucijsku neuronsku mrežu na

slici 3.9. je peterodimenzionalno (5D) polje. Prvi izraz s desna na lijevo u svim zagradama na slici 3.9. označava prvu veličinu u polju, a to je veličinu serije (engl. *batch size*). Veličina serije u svim zagradama je *None*, odnosno ništa, zbog toga što model unaprijed ne zna kolika je veličina serije. Model u prvu dimenziju polja sprema veličinu serije tek prilikom treniranja kada učitava varijablu *batch_size* iz funkcije *fit*. Druga veličina, koja se naziva *patch_size*, označava broj video okvira za svaku video sekvencu u kojoj je prikazana jedna gesta, a taj broj video okvira je postavljen na 25. Treća i četvrta veličina u polju označavaju rezoluciju svake slike u pikselima, koja je 64x64. Svakoj originalnoj slici je prilikom učitavanja u polje promijenjena veličina (engl. *resize*) rezolucije s 176x100 piksela na 64x64 piksela. Zadnja veličina u zagradama označava dubinu svake slike, pa tako u ovom slučaju vrijednost dubine iznosi 3 zbog RGB vrijednosti. Izlaz iz mreže je naziv geste koja se detektira. Slika 3.10. prikazuje vizualizaciju arhitekture neuronske mreže. Ovaj model sastoji se od 3D konvolucijskih slojeva. Kada se radi obrada video podataka, uobičajeni pristup pretvaranju 2D video okvira u 3D podatke je slaganje uzastopnih okvira u novu dimenziju koja predstavlja vrijeme. Zbog toga je ulaz u ovaj model koji je prikazan na slici 3.9. 5D polje, jer je u 4D polje, koje se koristi kao ulaz u konvolucijske neuronske mreže za obradu slike, potrebno dodati novu dimenziju, a to je broj okvira koji čine video sekvencu. Osnovna arhitektura koja je korištena za sekvencijalni model neuronske mreže sastoji se od četiri 3D-CNN konvolucijska dijela mreže u kojima su definirani konvolucijski slojevi te od dva LSTM sloja. Prvi konvolucijski dio mreže sastoji se od dva trodimenzionalna konvolucijska sloja definirana sa 16 kernela veličine (3,3,3) te korakom od (1,1,1). Nakon toga je definiran sloj maksimalnog udruživanja (engl. *Max Pooling Layer*) kojemu je veličina udruživanja (engl. *pool size*) (2,2,2). Ostali konvolucijski slojevi imaju istu veličinu kernela – (3,3,3) te isti korak – (1,1,1). Razlika je u tome što je u svakom od tih konvolucijskih slojeva promijenjen broj kernela kojim se odrađuje konvolucija na video okvirima, a koji prolaze kroz neuronsku mrežu prilikom treniranja. U drugom konvolucijskom dijelu koriste se 32 kernela, u trećem 64 kernela, a u četvrtom je konvolucijskom sloju broj kernela povećan na 128. U trećem i četvrtom konvolucijskom dijelu mreže koriste se tri trodimenzionalna konvolucijska sloja.



Slika 3.9. Rješenje arhitekture neuronske mreže



Slika 3.10. Vizualizacija rješenja arhitekture neuronske mreže

U svakom od slojeva korištena je i aktivacijska funkcija ReLU, regularizator kernela (engl. *weight decay*) koji je postavljen na vrijednost 12 i razina dilatacije (engl. *dilation rate*) od (1,1,1). Sloj maksimalnog udruživanja u drugom, trećem i četvrtom konvolucijskom dijelu mreže postavljen je na (1,2,2). Nakon konvolucijskih slojeva uvedena su dva LSTM sloja mreže od 64 kernela veličine (3,3) uz korak od (1,1). Nakon LSTM slojeva dodan je sloj globalnog srednjeg udruživanja (engl. *global average pooling*) i sloj normalizacije serije (engl. *batch normalization* [22]). U zadnjem dijelu arhitekture dodan je sloj gustoće (engl. *dense layer*) od 10 jedinica jer je zadatak ove neuronske mreže klasifikacija deset gesti prstiju. Na kraju je dodana *softmax* aktivacijska funkcija. Poslije svakog konvolucijskog dijela uveden je sloj gubitka (engl. *Dropout Layer*), odnosno isključivanja neurona razine 0.25 kako bi se smanjila vrijednost funkcije gubitaka na validacijskom skupu podataka prilikom treniranja, tj. kako bi se smanjila pretreniranost. Pretreniranost [23] se odnosi na činjenicu kada model nauči predobro pamtiti podatke za treniranje do te mjere da izgubi mogućnost prilagođavanja općenitim uzorcima, odnosno model odgovara samo specifičnim uzorcima iz skupa podataka na kojem se trenira. Pretreniranost često nastaje kod podataka koji su vrlo slični. Kao rezultat pretreniranosti, model može raditi vrlo dobro na podacima za treniranje, a jako loše na skupovima podataka za validaciju ili testiranje. Prilikom treniranja koristi se kategorizacijska unakrsna entropija (engl. *categorical crossentropy*) za praćenje gubitaka točnosti i validacije tijekom treniranja mreže, a za optimizaciju se koristi Adamov optimizator [24] uz brzinu učenja (engl. *learning rate*) od 0.0001. Brzina učenja [25] ima ključnu ulogu prilikom treniranja neuronske mreže. Određuje veličinu ažuriranja parametara modela na temelju gradijenta funkcije gubitaka te na taj način pokušava minimizirati funkciju gubitaka. Ovaj definirani model neuronske mreže se sveukupno sastoji od 2 170 794 parametara, od čega su 2 170 666 parametara predviđeni za treniranje, a 128 parametara ne prolaze proces treniranja. U prilogu P.3.2 nalazi se programski kod kojim se definira model neuronske mreže u programskom jeziku *Python*.

3.6. Treniranje dugoročne rekurentne konvolucijske mreže (LRCN mreža)

Treniranje neuronske mreže izvršeno je na bazi podataka opisanoj u potpoglavlju 3.3. Prije početka treniranja neuronske mreže, bazu video sekvenci bilo je potrebno prilagoditi za treniranje. Zbog ograničenosti virtualne memorije grafičke kartice poslužitelja na kojemu je trenirana LRCN mreža, bilo je potrebno pripremiti ulazni skup podataka za ulaz u mrežu. Broj okvira za svaku video sekvencu sveden je na 25 okvira od ukupno 37 okvira kako bi *numpy* polje u koje se spremaju svi podaci prilikom treniranja bilo moguće učitati u virtualnu memoriju grafičke kartice

te kako bi ostala memorija bila dostupna za matematičke operacije koje neuronska mreža izvodi nad okvirima. Svaki okvir iz video sekvenci smanjen je na rezoluciju 64x64 piksela i svako se polje od 25 okvira sprema u *numpy* polje. Nakon normalizacije RGB vrijednosti, svaki piksel u *numpy* polju ima vrijednost od 0 do 1. Prilikom obrade svakog okvira, prostor boja svakog od okvira transformira se iz BGR prostora boja u RGB prostor boja zbog toga što *OpenCV* biblioteka automatski učitava slike u BGR formatu te je svaku učitanu sliku potrebno ponovno transformirati u RGB format. Nakon pripreme ulaznih podataka izgled trening podataka na ulazu u mrežu je sljedeći: (25, 64, 64, 3). Prvi broj u zapisu označava broj video okvira svake video sekvence, drugi i treći broj označavaju rezoluciju svakog okvira, a zadnji broj označava dubinu (engl. *depth*) svakog okvira, koja je zapisana pomoću 3 bajta, od kojih svaki označava boju, odnosno kanal RGB standarda.

Svakoj video sekvenci dodana je oznaka klase (engl. *label*) što znači da svaka klasa može nositi oznake od 0 do 9, npr. prva klasa koja obuhvaća geste od 0. do 700. videa ima vrijednost oznake 0, sljedeća klasa koja obuhvaća geste od 700. do 1400. videa ima vrijednost oznake 1, i tako sve do zadnje oznake. U navedenim intervalima svake oznake klase sadržane su video sekvence istih gesti prstiju.

Prije početka treniranja, baza video okvira je podijeljena na dio za treniranje, dio za validaciju i na dio za testiranje neuronske mreže. Za treniranje je korišteno 80% baze podataka (5600 video sekvenci), za validaciju 10% (700 video sekvenci) te za testiranje preostalih 10% (700 video sekvenci). Prilikom podjele baze podataka korištena je funkcija *train_test_split* te joj je predana *stratify* funkcija u argumentu kako bi se uzorci baze video sekvenci jednako podijelili na deset definiranih gesti. U argumentu je predana i *shuffle* funkcija koja prilikom podjele baze podataka nasumično uzima uzorke iz svih klasa. Varijabla *random_state* označava koliko se uzoraka odjednom uzima tijekom *shuffling* procesa. Da bi se baza podataka podijelila na ovaj način potrebno je pozvati funkciju *train_test_split* iz *sklearn* biblioteke sljedećim naredbama:

X_train_new, X_rem, y_train_new, y_rem = train_test_split(train_set, Y_train , train_size=0.8, random_state=42, shuffle=True, stratify=Y_train)

X_val_new, X_test_new, y_val_new, y_test_new = train_test_split(X_rem,y_rem, test_size=0.5, random_state=42, shuffle=True ,stratify=y_rem)

Odlučeno je da se trening svede na 45 epoha te je izabrana veličina serije vrijednosti 8 jer se pokazalo da ta veličina daje najbolje rezultate te smanjuje vrijednosti funkcije gubitaka na validacijskom skupu podataka. U rješenju ovog diplomskog rada pretreniranost je smanjena

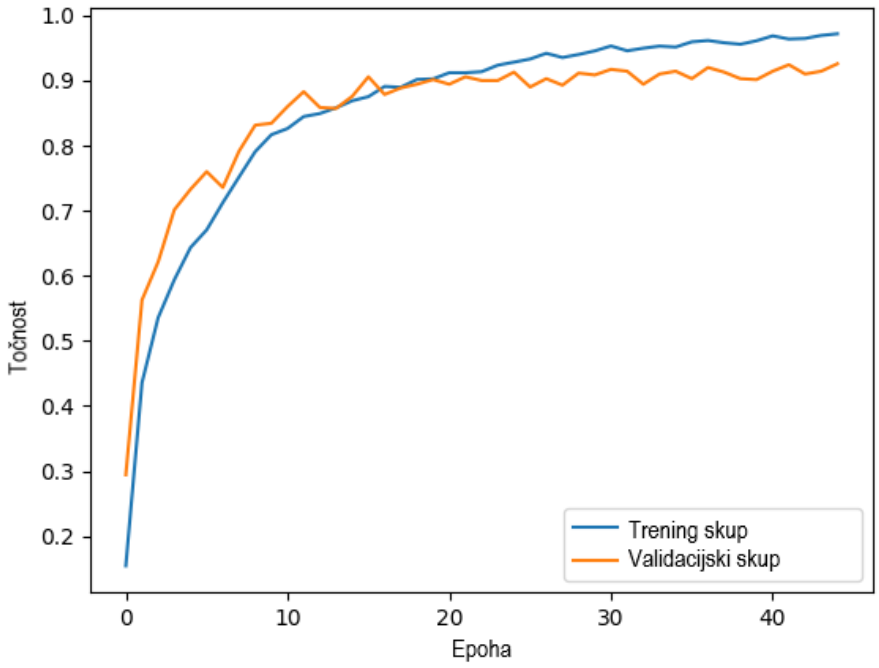
povećanjem *dropout* slojeva nakon svakog maksimalnog sloja udruživanja te povećanjem uzoraka baze podataka.

Za treniranje neuronske mreže korišten je poslužitelj (engl. *server*) koji ima ugrađenu grafičku karticu NVIDIA RTX 3080 Ti s 12 GB virtualne memorije, na taj se način značajno smanjilo vrijeme treniranja neuronske mreže.

Za treniranje definiranog modela LRCN mreže prikazane na slici 3.9. na 20BN-Jester bazi video sekvenci korišteni su sljedeći trening parametri:

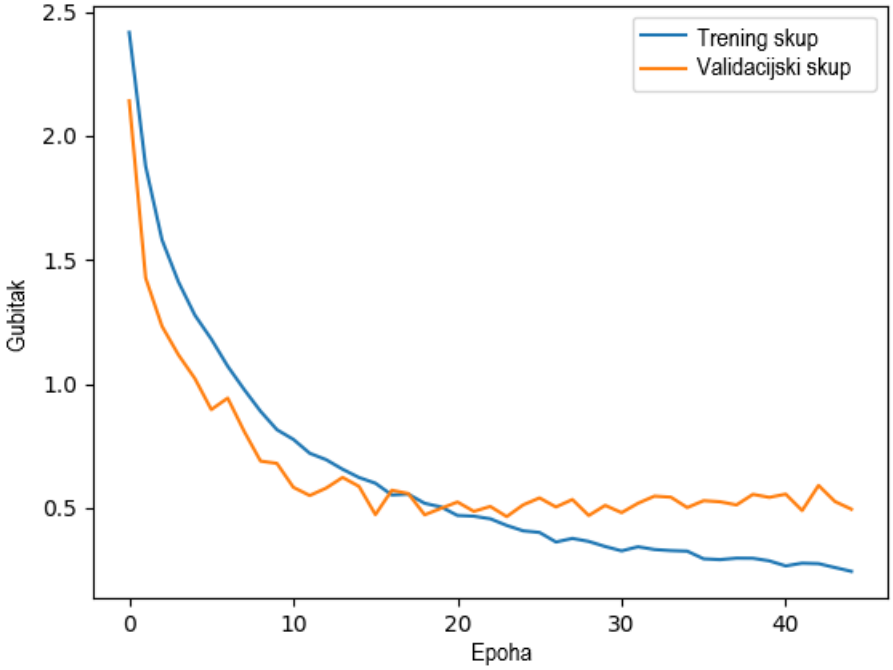
- veličina serije (engl. *batch size*) – 8;
- broj epoha – 45;
- brzina učenja – 0.0001;
- *shuffle*;
- Adamov optimizator;
- isključivanje neurona (engl. *dropout*) – 25%.

Na slici 3.11. prikazana je krivulja točnosti na trening i validacijskom skupu podataka tijekom treniranja LRCN mreže na *20BN-Jester* bazi video sekvenci.



Slika 3.11. Krivulja točnosti na trening i validacijskog skupu podataka kroz 45 epoha treniranja LRCN mreže na *20BN-Jester* bazi video sekvenci

Na slici 3.12. je pirkazana krivulja gubitaka na trening i validacijskom skupu podataka tijekom treniranja LRCN mreže na *20BN-Jester* bazi video sekvenci. Korištena je kategorička unakrsna entropija (engl. *categorical crossentropy*) kao funkcija gubitaka.



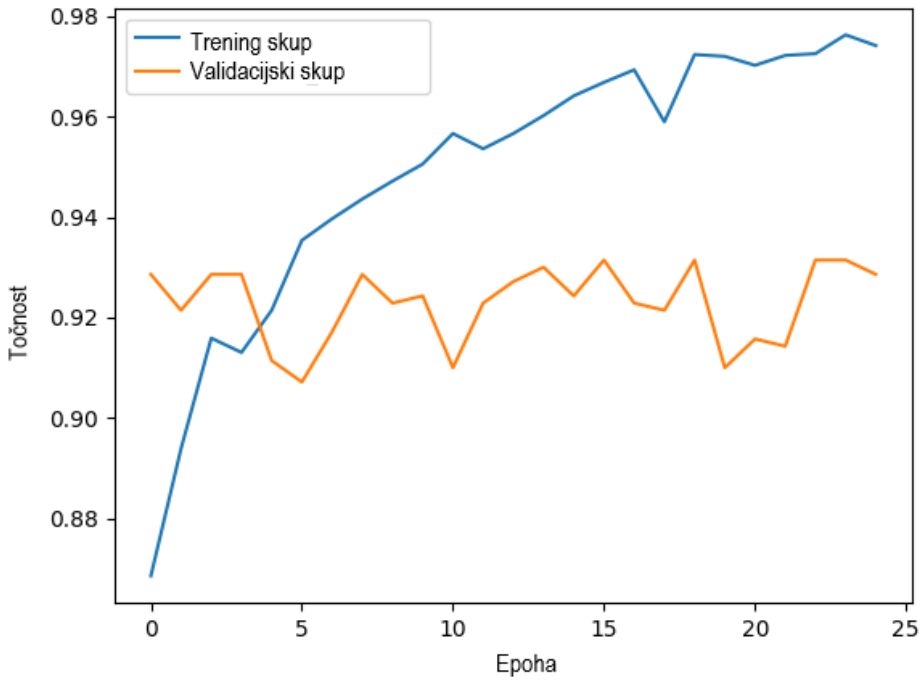
Slika 3.12. Krivulja gubitaka na trening i validacijskog skupu podataka kroz 45 epoha treniranja LRCN mreže na *20BN-Jester* bazi video sekvenci

Nakon treniranja LRCN mreže uzeti su parametri dobiveni nakon 45. epohe. Postotak točnosti na testnom skupu baze *20BN-Jester* iznosi 89,00%. Kako bi se povećala točnost modela odlučeno je da se spremljeni model ponovno istrenira na bazi *20BN-Jester*, ali ovaj put sa različitim video sekvencama za svaku klasu iz baze *20BN-Jester*. Zbog toga je bilo potrebno napraviti novu bazu od 7000 video sekvenci za treniranje mreže. Ovaj postupak ponovnog treniranja modela koji je već naučen da klasificira određene radnje, objekte ili aktivnosti naziva se prijenos učenja (engl. *transfer learning*) [26]. Metoda prijenosa učenja implementirana je kako bi se povećala točnost modela prilikom evaluacije na testnom skupu.

Za ponovno treniranja modela na novom skupu *20BN-Jester* video sekvenci korišteni su sljedeći trening parametri:

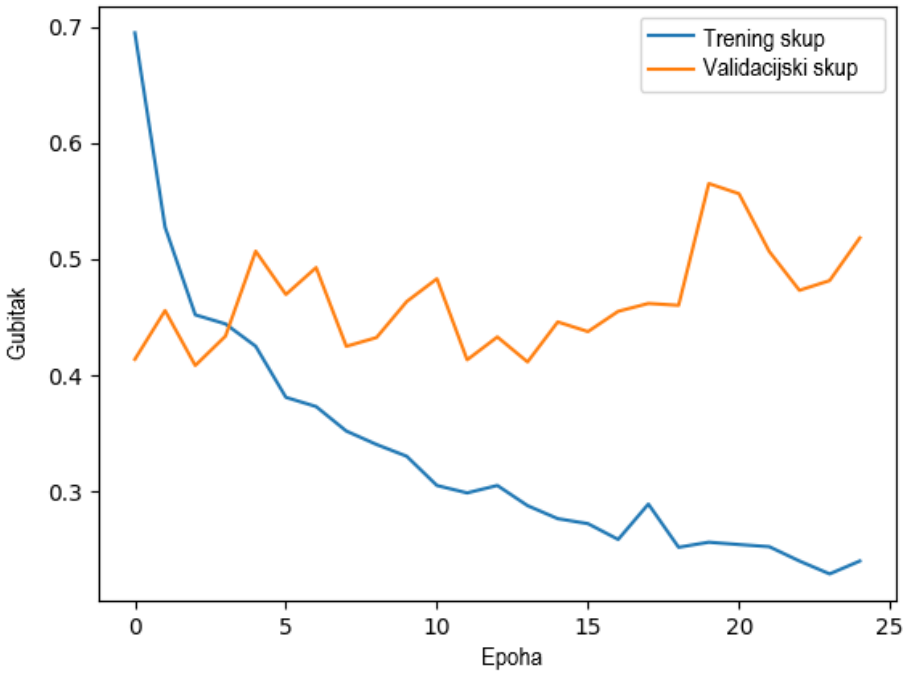
- veličina serije (engl. *batch size*) – 8;
- broj epoha – 25;
- brzina učenja – 0.0001;
- *shuffle*;
- Adamov optimizator;
- isključivanje neurona (engl. *dropout*) – 25%.

Na slici 3.13. prikazana je krivulja točnosti na trening i validacijskom skupu podataka tijekom ponovnog treniranja LRCN modela na novom skuup *20BN-Jester* video sekvenci.



Slika 3.13. Krivulja točnosti na novom trening i validacijskog skupu podataka kroz 25 epoha ponovnog treniranja prethodno istreniranog LRCN modela na *20BN-Jester* bazi video sekvenci

Na slici 3.14. prikazana je krivulja gubitaka na trening i validacijskom skupu podataka tijekom ponovnog treniranja LRCN modela na novom skupu *20BN-Jester* video sekvenci.



Slika 3.14. Krivulja gubitaka na novom trening i validacijskog skupu podataka kroz 25 epoha ponovnog treniranja prethodno istreniranog LRCN modela na *20BN-Jester* bazi video sekvenci

Nakon primjene *transfer learning* metode, model je postigao točnost od 92,14% na novom testnom skupu *20BN-Jester* video sekvenci, što je za 3,14% više u odnosu na prvotno istrenirani model. Zbog postignute visoke razine točnosti, ovaj će se model koristiti za detekciju gesti prstiju kako bi se upravljalo *Spotify* aplikacijom.

Prilikom treniranja oba modela vidljiva je pojava pretreniranosti. Na slici 3.11. pretreniranost se pojavljuje na otprilike 25. epohi. Primjećuje se stagnacija točnosti na validacijskom skupu podataka. Točnost na trening skupu još uvijek raste do 45. epohe, nakon koje je moguće primijetiti i stagnaciju na trening skupu. Vidljivo je kako točnost na validacijskom skupu ne raste, već varira od 85% do 88%. Prilikom izrade ovoga rješenja vodilo se računa da se pretreniranost smanji na najmanju moguću razinu primjenom različitih metoda koje su navedene prethodno u ovome poglavlju (dodavanje *dropout* slojeva prilikom definiranja arhitekture modela, povećanje uzoraka baze video sekvenci, smanjivanje veličine serije, itd.).

3.7. Realizacija web aplikacije za upravljanje *Spotify* servisom

Model neuronske mreže integriran u *Spotify* aplikaciju osposobljen je za prepoznavanje i tumačenje implementiranih gesti ruku i prstiju. Putem USB kamere, snima se videozapis od 25 okvira u kojem korisnik aplikacije izvodi gestu. Model analizira taj videozapis u stvarnom vremenu kako bi prepoznao određenu gestu. Korisnik pomoću gesta može upravljati *Spotify* glazbenim servisom za *streaming* glazbenog sadržaja. Izabran je model na kojem je primijenjena *transfer learning* metoda iz 3. poglavlja koji je postignuo točnost od 92,14% na testnoj bazi.

3.7.1. Radno okruženje i korištene tehnologije za izradu *Fingerfy* web aplikacije

Prilikom izrade aplikacije za kontrolu *Spotify* glazbenog servisa korišteno je VSC (engl. *Visual Studio Code*) razvojno okruženje. Cijela aplikacija napisana je u *Python* programskom jeziku. Osim biblioteka koje su navedene u 3. poglavlju koje se odnose na strojno učenje, bilo je potrebno koristiti sljedeće biblioteke: *tkinter* [27], *spotipy* [28] te *requests* [29].

3.7.1.1. *Tkinter* biblioteka

Tkinter je *Python* biblioteka koja se koristi za stvaranje grafičkih korisničkih sučelja (engl. *Graphical User Interface,* GUI). Omogućuje izradu interaktivnih i efikasnih aplikacija. *Tkinter* nudi širok raspon *widgeta*, uključujući oznake, tekstne okvire, gumbe i izbornike. Podržava različite radnje uzrokovane događajima (engl. *event-driven paradigms*), time omogućujući da radnje budu pokrenute interakcijom korisnika.

3.7.1.2. *Spotipy* biblioteka

Spotipy je *Python* biblioteka koja nudi sučelje za interakciju sa *Spotify* web API-jem (engl. *Application Programming Interface*). Programerima omogućuje autentikaciju korisnika, traženje pjesama, albuma i popisa za reprodukciju, dohvaćanje korisničkih podataka i izvođenje drugih operacija. Uz *Spotipy,* programeri mogu jednostavno integrirati funkcionalnosti *Spotify* servisa u svoje aplikacije, bilo da se radi o izgradnji sustava za preporuku glazbe, stvaranja upravitelja popisa pjesama, analizi navika slušanja korisnika ili u ovome slučaju upravljanja *Spotify* servisa gestama prstiju.

3.7.1.3. *Requests* biblioteka

Ova biblioteka koristi se kao alat za izradu HTTP (engl. *Hypertext Transfer Protocol*) zahtjeva. Omogućuje jednostavno sučelje prilagođeno korisniku za interakciju s web uslugama i API-jima. Pomoću zahtjeva (engl. *requests*) programeri mogu s lakoćom slati GET, POST, PUT, DELETE i druge vrste HTTP zahtjeva. HTTP protokol služi za rukovanje zaglavljima (engl. *headers*), autentikacijom, kolačićima (engl. *cookies*) i upravljanje sesijama, pojednostavljujući proces izrade HTTP zahtjeva. Ova biblioteka podržava i rukovanje JSON (engl. *JavaScript Object Notation*) podacima, učitavanje podataka i SSL (engl. *Secure Sockets Layer)* provjeru.

3.7.2. *Spotify* web API i izrada aplikacije

Spotify web API je alat koji korisnicima omogućuje interakciju sa *Spotify* platformom i pristup širokom rasponu podataka povezanih s glazbom. Omogućuje *RESTful* sučelje za dohvaćanje informacija o pjesmama, albumima, izvođačima, popisima za reprodukciju i korisničkim profilima.

Prvi korak pri izradi aplikacije je autentikacija korisnika. Za početak korisnik mora napraviti račun na *Spotify for Developers* [30] web stranici kako bi mogao koristiti *Spotify* API. Nakon izrade računa korisnik dobiva svoj *Client ID* i *Client secret* identifikacijske podatke koje je potrebno postaviti prilikom autentikacije. Potrebno je postaviti i URL adresu korisničke web aplikacije. U samome kodu potrebno je postaviti područje (engl. *scope*) u kojem će se koristiti API. Sve od navedenog moguće je implementirati u aplikaciju na način kao što je opisano u prilogu P.3.3.

Nakon napisane logike za autentikaciju, potrebno je implementirati *requests* biblioteku kako bi se mogao pravilno koristiti API, tj. slanje zahtjeva HTTP serveru te dobivanje odgovora (engl. *response*). Nakon toga potrebno je definirati koje će se akcije izvesti u *Spotify* aplikaciji ovisno o detektiranoj gesti pomoću modela LRCN neuronske mreže. Primjer definiranja dvije funkcije za zvanje servera u ovisnosti o pokazanoj gesti i odrađivanje te akcije nakon detektirane geste dan je u prilogu P.3.4. Sva logika za korištenje API-ja i autentikacija napisana je unutar klase SpotifyAPI.

Prije ulaska svakog video okvira u neuronsku mrežu potrebno je izvesti predobradu podataka. Obrada podataka izvršava se pomoću funkcije *normaliz_data*. Svih 25 okvira tijekom kojih se radi detekcija sprema se u *numpy* polje. Svaka se slika, odnosno matrica, rezolucije 64x64 piksela normalizira. Zatim se računa srednja vrijednost i maksimalna vrijednost unutar svakog polja od 25

video okvira koje predstavlja jednu gestu. Funkcija vraća normalizirano polje od 25 video okvira. Primjer koda za normalizaciju podataka dan je u prilogu P.3.5.

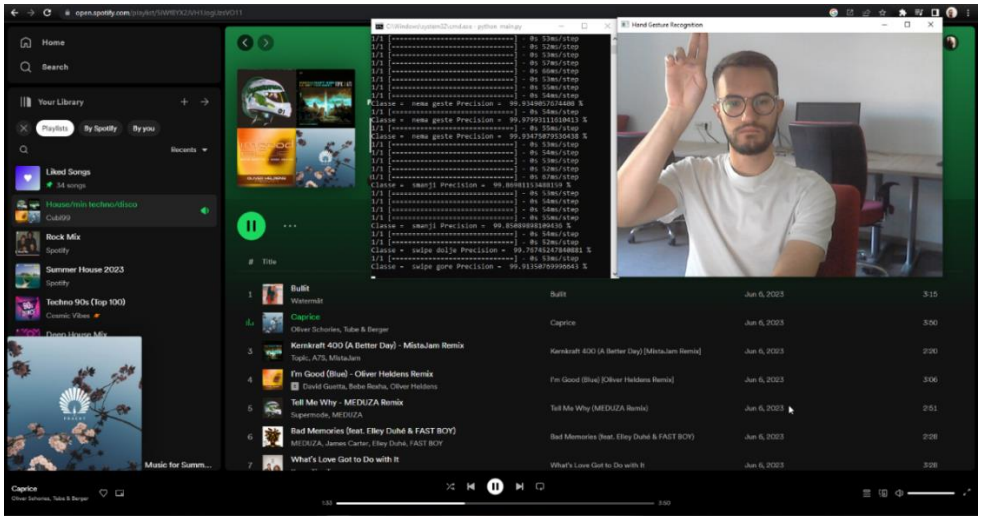
3.7.3. Izgled korisničkog sučelja i interakcija s korisnikom

Korisničko sučelje kreirano je korištenjem funkcija *tkinter* biblioteke. Kreirano sučelje omogućuje autentikaciju korisnika klikom miša na *Authenticate* gumb nakon čega se otvara prozor kamere gdje korisnik izvodi gestikulacije. Za kontrolu *Spotify* servisa potrebno je imati u isto vrijeme otvorenu *Spotify* aplikaciju u web pretraživaču (engl. *web browser*). U ovome radu korišten je *Google Chrome* pretraživač. U svakome trenutku korisnik može izaći iz aplikacije pritiskom na tipku *q* na tipkovnici ili pritiskom na gumb *Exit* u prozoru *Fingery* aplikacije. U postavkama *Spotify* aplikacije moguće je podesiti da korisnik gestama kontrolira neki drugi uređaj, npr. mobitel ili tablet. Izgled korisničkom sučelja aplikacije za kontrolu *Spotify* servisa dan je na slici 3.15.

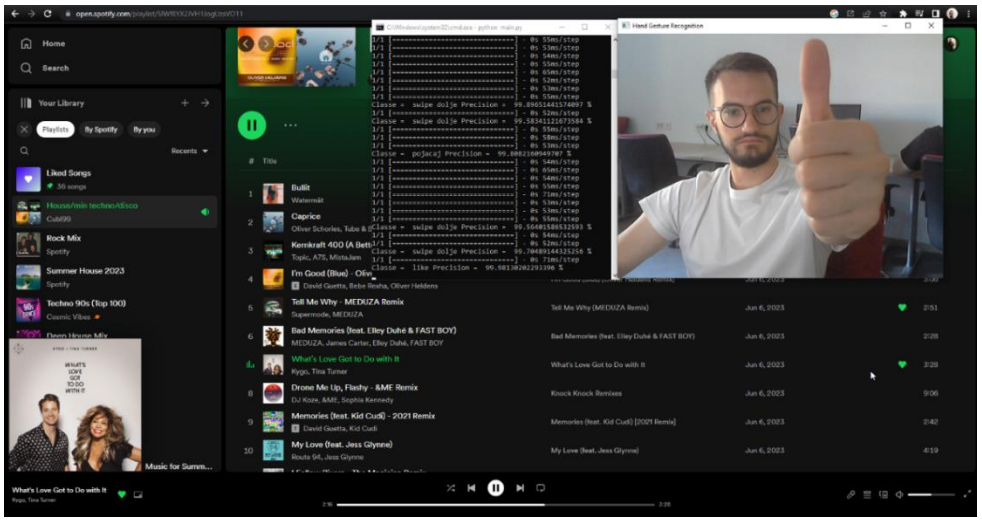
Slika 3.15. Izgled korisničkog sučelja kreirane aplikacije

Na slici 3.16. i 3.17. dani su primjeri upravljanja *Spotify* aplikacijom. Na slici 3.16. izvedena je „*swipe* gore“ gesta koja postavlja glasnoću zvuka na 100%. U svakom trenutku moguće je u

prozoru od naredbenog retka (engl. *command prompt*) iščitati gestu koja je detektirana te postotak preciznosti detektirane geste. Na slici 3.17. izvedena je gesta „*like*“ koja automatski pjesmu koja se trenutno reproducira stavlja u favorite, odnosno u omiljenu *playlistu*.



Slika 3.16. Primjer izvedbe *swipe* gore geste kojom se postavlja glasnoća na 100%



Slika 3.17.. Primjer izvedbe *like* geste kojom se trenutno reproducirana pjesma postavlja u favorite

3.8. Implementacija rješenja na *Raspberry PI 3* ugradbenu platformu

U ovom poglavlju opisana je implementacija rješenja na *Raspberry Pi 3* ugradbenu platformu. Za implementaciju se koristila *Raspberry Pi 3 Model B Vr.2* ugradbena platforma koja

sadrži 1.2 GHz četverojezgreni *ARM Cortex-A53* 64-bitni procesor, 1 GB LPDDR2 RAM memoriju, ugrađenu 2.4 GHz Wi-Fi i Bluetooth vezu. Od priključaka posjeduje HDMI priključak, četiri USB priključka i 40-pinskih GPIO (engl *General Purpose Input/Output*) priključaka za uspostavljanje komunikacije s različitim senzorima, modulima i ostalim hardverskim komponentama. Ovaj model pruža podršku za različite operacijske sustave, uključujući poznate *Raspbian* i *Debian* operacijske sustave temeljene na *Linux* kernelu. Služi kao platforma za razvoj različitih projekata u području ugradbenih sustava (engl. *embedded systems*).

Za potrebe ovoga rada na *Raspberry Pi 3* platformu instaliran je *Debian 11* operacijski sustav. Prvi korak u implementaciji rješenja nakon instaliranja operacijskog sustava prebacivanje je datoteke u kojem se nalazi gotovi projekt. Ovo se može obaviti „fizičkim" prebacivanje datoteke u kojem se nalazi rješenje pomoću SD kartice ili kloniranjem projekta s *GitHuba.* Kao alternativa *Anaconda Navigator* razvojnom okruženju koristi se *Miniconda* [31], okruženje koje je namijenjeno za ARM 64 procesore te za *Linux* sustave jer zauzima puno manje resursa. Nakon instaliranja operacijskog sustava i postavljanja datoteke s rješenjem, instalirana je *Python* verzija 3.10.10. Prije pokretanja projekta potrebno je napraviti virtualno okruženje i instalirati sve potrebne biblioteke koje se nalaze u *requirements.txt* datoteci sljedećim naredbama:

$ conda create –name <env> --file <file with requirements>

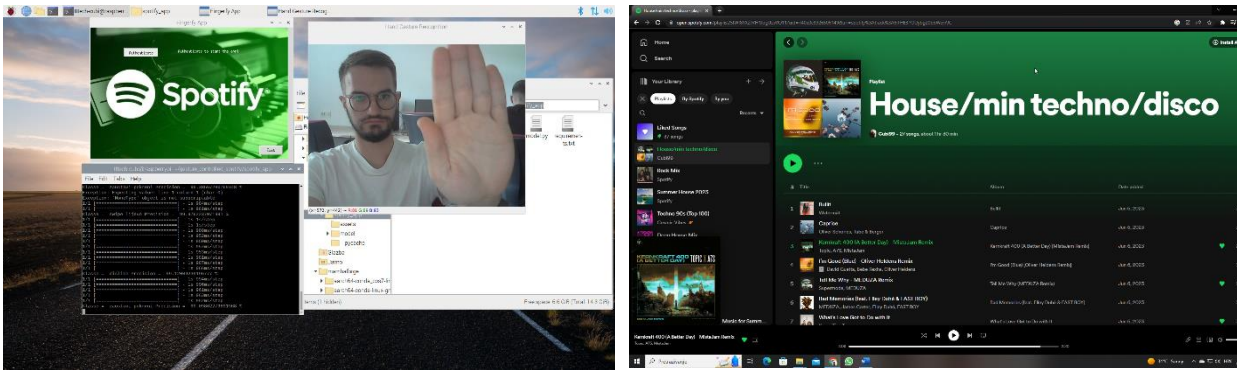
Nakon što se u naredbenom bloku navigira u datoteku s rješenjem naredbom

$ cd /home/tttech-cubi/gesture_controlled_spotify/spotify_app,

projekt se pokreće sljedećom naredbom:

$ python main.py

Rezultati testiranja rada neuronske mreže na *Raspberry Pi 3* ugradbenoj platformi dani su u 4. poglavlju. Uz ovakav sustav moguće je pokrenuti detekciju gesti na *Raspberry Pi 3* platformi uz spojenu USB kameru, a *Spotify* aplikaciju imati otvorenu na osobnom računalu u web pretraživaču jer se cijelo rješenje zasniva na *Spotify* API-ju te HTTP protokolu. Primjer detektiranja gesti na *Raspberyy Pi 3* platformi te izvedbi određene naredbe na osobnom računalu nakon uspješne detekcije prikazan je na slici 3.18.



(a) (b)

Slika 3.18. (a) detekcija geste na *Raspberry Pi 3* ugradbenoj platformi, (b) izvedba radnje na osobnom računalu putem HTTP protokola

4. EVALUACIJA NAPRAVLJENOG SUSTAVA ZA DETEKCIJU GESTI PRSTIJU

Evaluacija neuronskih mreža ključni je korak u procjeni njihovog rada i razumijevanju njihovih mogućnosti. Tijekom procesa evaluacije razmatrano je nekoliko ključnih stavki zbog ocjenjivanja učinkovitosti i pouzdanosti modela neuronskih mreža kao što su točnost, preciznost, opoziv (engl. *recall*), F1 rezultat (engl. *F1 score*), ROC (engl. *receiver operating characteristic*) krivulja, AUC (engl. *area under the curve*) krivulja te matrica zabune (engl. *confusion matrix*).

U prethodnom poglavlju opisana je točnost modela kao temeljna metrika koja mjeri ukupnu točnost predviđanja neuronske mreže. Točnost predstavlja omjer ispravno klasificiranih instanci u usporedbi s ukupnim brojem instanci u skupu podataka. Međutim, točnost sama po sebi ne može dati potpunu sliku o procjeni rada neuronske mreže. Postoji nekoliko razloga zašto točnost nije dovoljna za evaluaciju rada neuronske mreže:

- kod neuravnoteženih skupova podataka, model može postići visoku točnost predviđanjem klase koja ima najveći broj uzoraka za treniranje, a može biti loš za klase koje imaju značajno manji broj uzoraka;
- neuronske mreže mogu biti osjetljive na male promjene u ulaznim podacima, model može postići visoku točnost na trening skupu, a malu točnost na novom (testnom) skupu podataka;
- u slučajevima koji uključuju više klasa i više zadataka koje treba izvoditi model, točnost neće pružiti dovoljno informacija o tome kako će model raditi u realnim uvjetima na svakoj klasi ili zadatku, neovisno.

Matrica zabune koristi se kao jedna od važnijih metoda za procjenu izvedbe klasifikacijskog modela u strojnom učenju. Matrica zabune služi za procjenu performansi klasifikacijskog modela neuronske mreže na način, da prikazuje odnos između stvarnih i predviđenih oznaka svake klase. Matrica zabune je matrica dimenzije $n \times m$, gdje n i m označavaju redni broj pojedine klase pri čemu se n i m kreću u rasponu brojeva kojeg određuje broj klasa. Svaki *n*–ti redak odgovara stvarnim oznakama klase, a svaki *m*–ti stupac odgovara predviđenim oznakama klase. Elementi koji se ne nalaze na dijagonali predstavljaju pogrešne klasifikacije. Iz matrice se može izvesti nekoliko metrika procjene [32]:

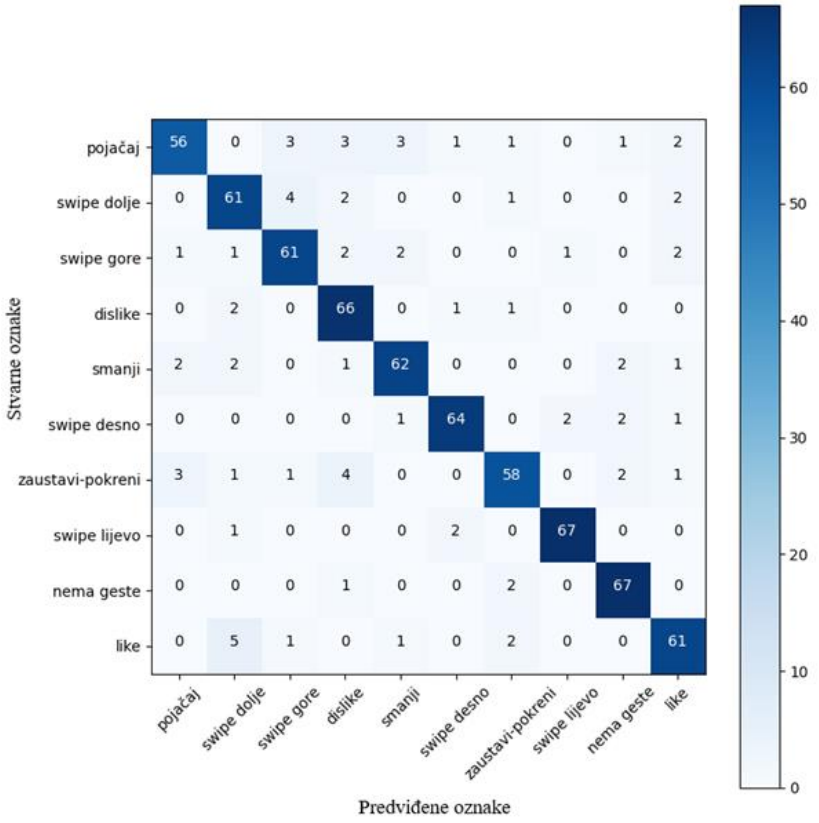
- ispravno pozitivne (engl. *true positive, TP*) video sekvence – označava da je neuronska mreža prepoznala da se na video sekvenci nalazi *n*-ta gesta te da je to točno;
- neispravno pozitivne (engl. *false positive, FP*) video sekvence – označava da je neuronska mreža prepoznala da se na video sekvenci nalazi *n*-ta gesta, ali je to netočno;
- neispravno negativne (engl. *false negative, FN*) video sekvence – označava da je neuronska mreža prepoznala da se na video sekvenci ne nalazi *n*-ta gesta, ali je to netočno.
- ispravno negativne (engl. *true negative, TN*) video sekvence – označava da je neuronska mreža prepoznala da se na video sekvenci ne nalazi *n*-ta gesta te da je to točno.

Kao druga metoda za evaluaciju performansi istrenirane neuronske mreže, napisana je *Python* aplikacija za testiranje rada neuronske mreže pomoću koje korisnici pokazuje gestu USB kameri. Model detektira gestu u stvarnom vremenu, a rezultat detekcije se ispisuje na prozor kamere prikazan na ekranu. U svrhu ovoga rada koristila se Logitech C270 HD kamera.

4.4. Evaluacija rada neuronske mreže na testnom skupu video sekvenci *20bn-jester*

Na slici 4.1. prikazana je matrica zabune LRCN modela kreirana na testnom skupu *20BN-Jester* video sekvenci. Za svaku se od deset klasa na istreniranome modelu testiralo 70 video sekvenci. Matrica zabune pokazuje da je većina predviđanja točna. Na primjer, za osmu gestu, „*swipe* lijevo“, model je prepoznao 67 video sekvenci kao ispravno pozitivne (96%), dok je samo tri video sekvence prepoznao kao neispravno pozitivne (4%). Model daje najviše neispravno pozitivnih vrijednosti za geste „*swipe* dolje“ i „*swipe* gore“ što je očekivano po prirodi gestikulacije i pokreta ljudske ruke tijekom izvođenja tih gesti. Sami pokreti klizanja (engl. *swipe*) prema gore ili dolje su vrlo slični te zbog toga dolazi do pogrešne detekcije tijekom testiranja. Također, istrenirani model gestu „zaustavi/pokreni“ najviše pogrešno detektira kao gestu „*dislike*“. Gesta „*like*“ najviše se pogrešno detektira kao gesta „*swipe* dolje“. Ostale geste se podjednako detektiraju kao ispravno negativne video sekvence, a model najviše dolazi do zabune u slučaju kada se detektiraju geste „*swipe* dolje“ i „*swipe* gore“. Najviše ispravno pozitivnih vrijednosti

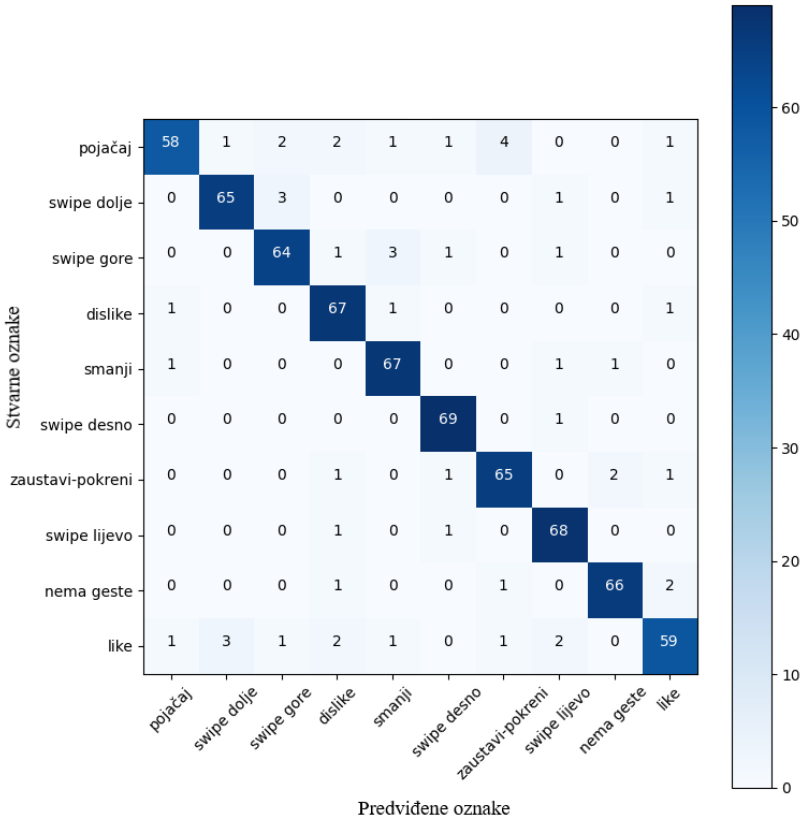
imaju geste „*swipe* lijevo“ (96%), „nema geste“ (96%) i gesta „*dislike*“ (94%). Najmanje ispravno pozitivnih vrijednosti imaju geste „pojačaj“ (80%) i „zaustavi/pokreni“ (83%). Postotak ispravno pozitivnih vrijednosti gesta je sljedeći: „*swipe* lijevo“ (96%), „nema geste“ (96%), „*dislike*“ (94%), „*swipe* desno“ (92%), „smanji“ (89%), „*like*“ (87%), „*swipe* dolje“ (87%), „*swipe* gore“ (87%), „zaustavi/pokreni“ (83%), „pojačaj“ (80%). Ukupna točnost LRCN modela na testnom skupu *20BN-Jester* video sekvenci iznosi 89,00%.



Slika 4.1. Matrica zabune istreniranog LRCN modela koja je kreirana na testnom skupu *20BN-Jester* video sekvenci

Na slici 4.2. dana je matrica zabune LRCN modela kreirana na validacijskom skupu *20BN-Jester* video sekvenci. Za svaku klasu testiralo se predviđanje za 70 video sekvenci. Pomoću matrice zabune na slici 4.2. može se zaključiti da model bolje predviđa geste na validacijskom skupu, nego što je to slučaj na testnom skupu, što je i prikazano na slici 4.1. Model je sveukupno prepoznao 92,57% ispravno pozitivnih vrijednosti gesta na validacijskom skupu u odnosu na 89,00% ispravno pozitivnih vrijednosti gesta na testnom skupu. Na temelju matrice zabune na slici 4.2. vidi se da je model najviše ispravno pozitivnih vrijednosti prepoznao za gestu „*swipe* desno“, točnije 69, što je 99%. Uz „*swipe* desno“, ostale geste koje prednjače prema broju ispravno

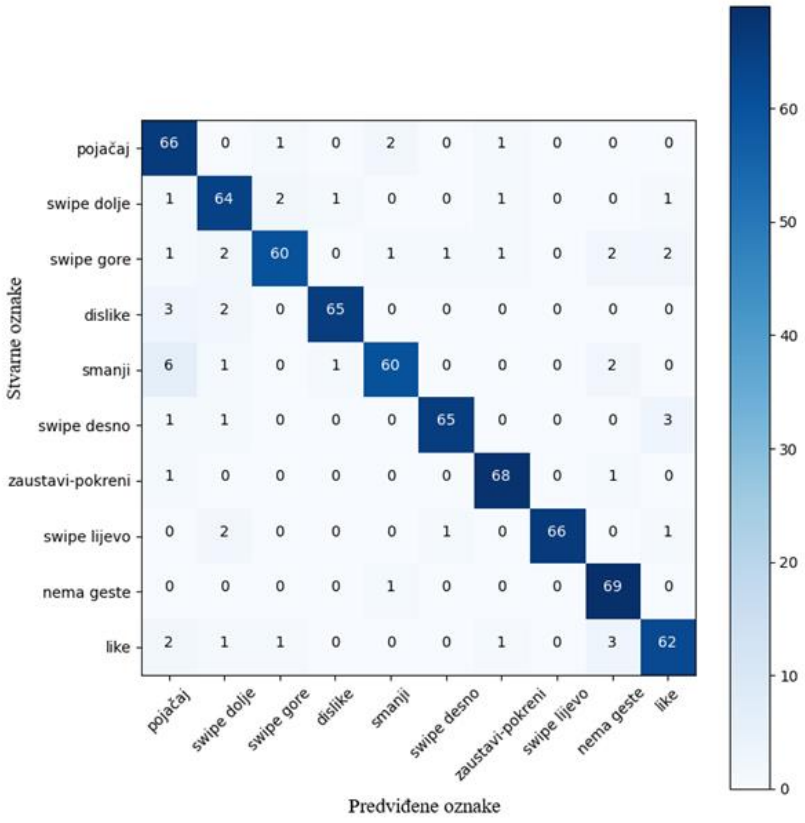
pozitivnih vrijednosti su „*swipe* desno“ (99%), „*swipe* lijevo“ (97%), „*dislike*“ (96%) te „smanji“ (96%). Najmanje ispravno pozitivnih vrijednosti imaju geste „pojačaj“ (83%) i „*like*“ (84%). Postotak ispravno pozitivnih vrijednosti gesta je sljedeći: „*swipe* desno“ (99%), „*swipe* lijevo“ (97%), „smanji“ (96%), „*dislike*“ (96%), „nema geste“ (94%), „*swipe* dolje“ (93%), „zaustavi/pokreni“ (93%), „*swipe* gore“ (91%), „*like*“ (84%), „pojačaj“ (83%). Ukupna točnost LRCN modela na validacijskom skupu *20BN-Jester* video sekvenci iznosi 92,57%.



Slika 4.2. Matrica zabune istreniranog LRCN modela koja je kreirana na validacijskom skupu *20BN-Jester* video sekvenci

Na slici 4.3. dana je matrica zabune ponovno istreniranog LRCN modela kreirana na testnom skupu *20BN-Jester* video sekvenci. Kao i za matricu zabune na slici 4.1., za svaku se od deset klasa na ponovo istreniranome modelu testiralo 70 novih video sekvenci. U ovome slučaju može se uočiti da model bolje predviđa geste poslije primjene metode prijenosa učenja. Na primjer, za sedmu gestu, „zaustavi/pokreni“, model je prepoznao 68 video sekvenci kao ispravno pozitivne (97%), dok je samo dvije video sekvence prepoznao kao ispravno negativne (3%). Model daje najviše neispravno pozitivnih vrijednosti za geste „*swipe* gore“ i „smanji“. Kao i u prethodnom

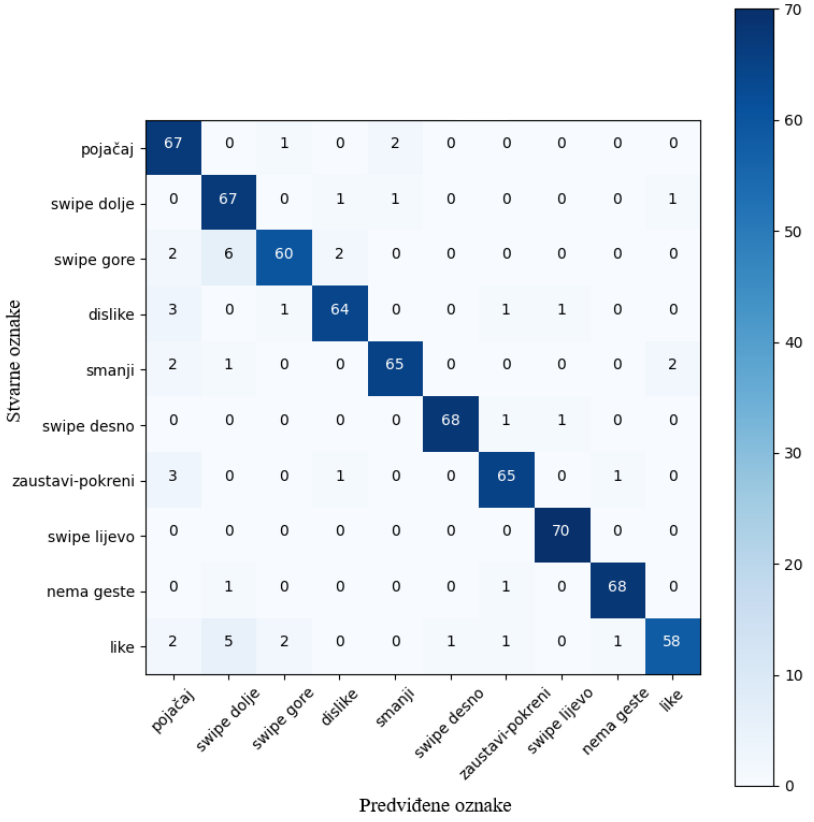
slučaju, model dolazi do zabune kada su u pitanju geste „*swipe* dolje“ i „*swipe* gore“ te geste „pojačaj“ i smanji“. Pošto su po prirodi gestikulacije ovi parovi gesti slični, prilikom detekcije dolazi do zabune jer treba uzeti u obzir da nisu sve geste iz baze 2*0BN-Jester* pravilno izvedene od strane ljudi koji su sudjelovali u izradi baze video sekvenci. Najviše ispravno pozitivnih vrijednosti imaju geste „nema geste“ (99%) te „zaustavi/pokreni“ (97%). Najmanje ispravno pozitivnih vrijednosti imaju geste „*swipe* gore“ (86%) te „smanji“ (86%). Postotak ispravno pozitivnih vrijednosti gesta je sljedeći: „nema geste“ (99%), „zaustavi/pokreni“ (97%), „*swipe* lijevo“ (94%), „pojačaj“ (94%), „*dislike*“ (93%), „*swipe* desno“ (93%), „*swipe* dolje“ (92%), „*like*“ (89%), „*swipe* gore“ (86%), „smanji“ (86%). Ukupna točnost ponovno istreniranog LRCN modela na testnom skupu *20BN-Jester* video sekvenci iznosi 92,14%.



Slika 4.3. Matrica zabune ponovno istreniranog LRCN modela (primijenjena *transfer learning* metoda) koja je kreirana na testnom skupu *20BN-Jester* video sekvenci

Na slici 4.4. dana je matrica zabune ponovno istreniranog LRCN modela kreirana na validacijskom skupu *20BN-Jester* video sekvenci. Za svaku od 10 klasa provelo se testiranje za 70 video sekvenci na validacijskom skupu podataka. Slično kao i na slici 4.2., model predviđa geste

bolje na validacijskom skupu, nego što je to slučaj na testnom skupu. Model je sveukupno prepoznao 93,14% ispravno pozitivnih vrijednosti na validacijskom skupu u odnosu na 92,14% ispravno pozitivnih vrijednosti gesta na testnom skupu. Ovaj omjer prepoznatih ispravno pozitivnih vrijednosti na testnom skupu u odnosu na validacijski skup puno je manji nego što je to u slučaju bez primijenjene metode prijenosa učenja. Na temelju matrice zabune na slici 4.4. vidi se da je model za gestu „*swipe* lijevo“ sve vrijednosti prepoznao kao ispravno pozitivne. Uz „*swipe* lijevo“, ostale geste koje prednjače prema broju ispravno pozitivnih vrijednosti su „nema geste“ (97%), „*swipe* desno“ (97%), „pojačaj“ (96%) i „*swipe* dolje“ (96%). Najmanje ispravno pozitivnih vrijednosti imaju geste „*like*“ (83%) i „*swipe* gore“ (86%). Postotak ispravno pozitivnih vrijednosti gesta je sljedeći: „*swipe* lijevo“ (100%), „nema geste“ (97%), „*swipe* desno“ (97%), „pojačaj“ (96%), „*swipe* dolje“ (96%), „smanji“ (93%), „zaustavi/pokreni“ (93%), „*dislike*“ (91%), „*swipe* gore“ (86%), „*like*“ (83%). Ukupna točnost ponovno istreniranog LRCN modela na validacijskom skupu *20BN-Jester* video sekvenci iznosi 93,14%.



Slika 4.4. Matrica zabune ponovno istreniranog LRCN modela (primijenjena *transfer learning* metoda) koja je kreirana na validacijskom skupu *20BN-Jester* video sekvenci

4.5. Rezultati testiranja rada neuronske mreže u realnim uvjetima

Na slici 4.5. prikazana je matrica zabune ponovno istreniranog LRCN modela kojeg je testiralo deset nasumičnih korisnika aplikacije napisane u programskom jeziku *Python*. Svaki od deset korisnika imao je zadatak svaku gestu napraviti pet puta, što znači da je svaki korisnik napravio 50 gesti, a to je sveukupno 500 prikazanih gesti.

Sama aplikacija, nazvana *test_app*, učitava istrenirani model neuronske mreže koji se kasnije koristi za detekciju gesti. Model je moguće učitati i kompajlirati sljedećim naredbama:

```
model.load_weights("C:/Users/lovro/Desktop/jester_evaluacija/7000, nema geste, retrained/model/model/")
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
```

Videozapis se snima putem USB kamere spojene s računalom. Detekcija gesti vrši se u stvarnom vremenu svakih 25 video okvira. Nakon što je gesta učinjena od strane korisnika i detektirana, na prozoru kamere, prikazanog na monitoru osobnog računala, ispisuje se naziv detektirane geste. Ovo rješenje je moguće implementirati u testnoj aplikaciji *test_app* nakon što se otvori kamera pomoću *OpenCV* biblioteke, na način opisan u prilogu P.4.1.

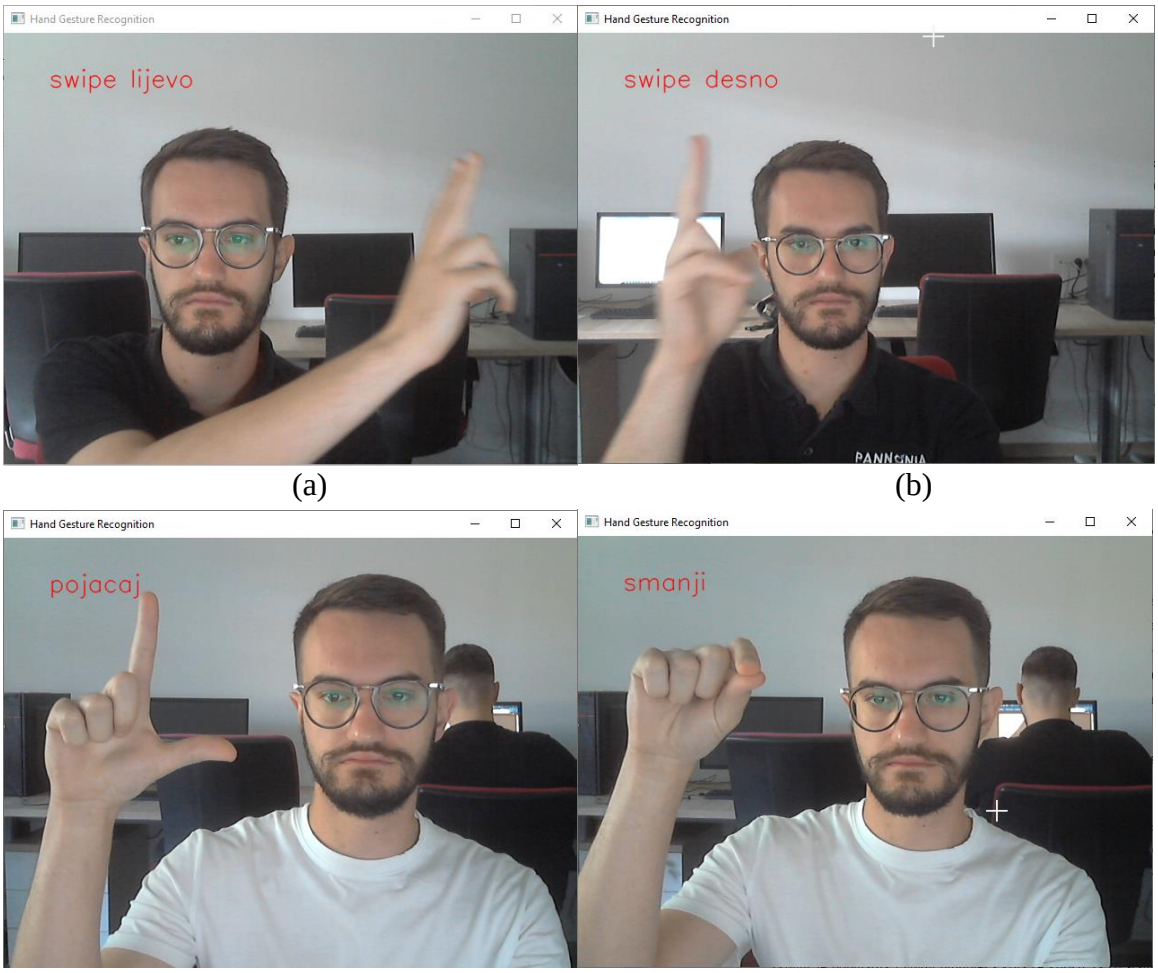
Pomoću matrice zabune na slici 4.5. može se razumjeti koje geste model teže detektira nego ostale kao i kod kojih gesti najviše dolazi do zabune. Postotak ispravno pozitivnih vrijednosti gesti je sljedeći: „*dislike*“ (92%), „nema geste“ (92%), „*swipe* lijevo“ (90%), „*swipe* desno“ (86%), „*like*“ (82%), „zaustavi/pokreni“ (80%), „pojačaj“ (76%), „*swipe* gore“ (72%), „smanji“ (70%), „*swipe* dolje“ (62%).

UKUPNO	nema geste	zaustavi/pokreni	swipe dolje	swipe gore	swipe lijevo	swipe desno	like	dislike	pojačaj	smanji
nema geste	**46**	2	0	0	0	2	1	0	1	1
zaustavi/pokreni	0	**40**	0	3	0	0	0	2	3	1
swipe dolje	0	0	**31**	8	0	0	2	0	3	1
swipe gore	0	1	1	**36**	0	0	1	0	1	0
swipe lijevo	0	0	0	0	**45**	0	0	0	0	0
swipe desno	0	0	0	0	0	**43**	0	1	0	0
like	0	0	0	0	0	0	**41**	0	0	0
dislike	2	4	13	1	0	1	0	**46**	0	4
pojačaj	1	1	1	1	0	1	3	1	**38**	8
smanji	1	2	4	1	5	3	2	0	4	**35**
∑	50	50	50	50	50	50	50	50	50	50

Slika 4.5. Matrica zabune ponovno istreniranog LRCN modela kojeg je testiralo deset nasumičnih korisnika *test_app* aplikacije

Slično kao i kod evaluacije ovoga modela na testnoj bazi *20BN-Jester*, model najviše zamjenjuje gestu „*swipe* gore“ s gestom „*swipe* dolje“ i obrnuto. Do toga dolazi jer korisnik pri izvedbi geste „*swipe* gore“ automatski spušta ruku u prirodnu poziciju pri čemu model detektira gestu „swipe dolje“. Zato je pri izvedbi svake geste potrebno usporeno maknuti ruku u desno ili lijevo kako bi prozor kamere bio oslobođen te spreman za izvedbu sljedeće geste. Ovaj problem riješen je na način, da se u dijelu koda *test_app* aplikacije, u kojemu je otvoren prozor kamere pomoću *OpenCV* biblioteke, preskače 25 video okvira. Detekcija se vrši u prvih 25 okvira, zatim sljedećih 25 okvira model ne detektira gestu, itd. Način detekcije geste s „praznim hodom“ od 25 okvira moguće je ostvariti sljedećim naredbama:

```
skip_frame_number = 25
cap.set(cv2.CAP_PROP_POS_FRAMES, skip_frame_number)
success, frame = cap.read()
```



Slika 4.6. Primjer ispisivanja klase u *test_app* aplikaciji nakon učinjene i pravilno detektirane geste: (a) „*swipe* lijevo“, (b) „*swipe* desno“, c) „pojačaj“, d) „smanji“

Na slici 4.6. prikazani su primjeri ispitivanja gesti i ispisivanja imena klase nakon pravilno detektirane geste. Sve ostale geste izvode se na isti način kao i video sekvence u *20BN-Jester* skupu podataka. Što se tiče rezultata testiranja aplikacije pomoću deset nasumičnih korisnika, točnost modela pomoću ovakvog testiranja iznosi 80,2%. Model je točno detektirao 402 geste od ukupno 500 gesti koje su korisnici izveli, što je vidljivo na slici 4.5. Točnost pri testiranju ove aplikacije pomoću korisnika manja je nego evaluacija točnosti samoga modela na testnom skupu, zbog toga što u ovome slučaju postoji faktor ljudske greške. Kod ovakvog načina testiranja javljaju se dodatni problemi kod detekcije gesti u stvarnome vremenu: okolina kadra kamere u kojoj neki objekti ili radnje mogu zasmetati prilikom detekcije gesti. Preporuča se da je okolina kamere slobodna dok se izvodi gestikulacija, odnosno da u kadru kamere tijekom izvođenja nema drugih osoba ili pokreta.

4.6. Rezultati testiranja rada neuronske mreže u realnim uvjetima na ugradbenoj platformi *Raspberry Pi 3*

Na slici 4.7. prikazana je matrica zabune ponovno istreniranog LRCN modela kojeg je testiralo deset nasumičnih korisnika na *Raspberry Pi 3* ugradbenoj platformi. Slično kao i kod testiranja modela koristeći *test_app* aplikaciju, svaki je korisnik imao zadatak svaku gestu izvesti pet puta, što je sveukupno 500 prikazanih gesti. Točnost modela nakon testiranja iznosi 73,6%, što znači da je model točno detektirao 368 gesti od sveukupno 500 izvedenih gesti. Za razliku od testiranja modela pomoću *test_app* aplikacije na osobnom računalu, testiranje na ugradbenoj platformi *Raspberry Pi 3* pokazalo je da model lošije vrši detektiranje gesti uključujući i faktor ljudske greške prilikom pokazivanja gesti. Razlog lošijeg detektiranja gesti na *Raspberry Pi 3* ugradbenoj platformi je taj što *Raspberry Pi 3* raspolaže s puno manje resursa (RAM memorija, brzina procesora, itd.) nego osobno računalno. Znatno ograničeni računalni resursi *Raspberry Pi 3* ugradbene platforme u usporedbi s osobnim računalom, rezultiraju sporijom obradom okvira iz videozapisa koji se snima u stvarnom vremenu. Zbog toga dolazi do manje preciznosti prilikom detekcije gesti. Za razliku od 402 točno detektirane geste u realnim uvjetima na osobnom računalu, na *Raspberry Pi 3* ugradbenoj platformi model je točno detektirao 368 gesti od ukupno 500, što je 34 pogrešno detektiranih gesti više nego na osobnom računalu.

Pomoću matrice zabune na slici 4.7. vidi se kako je model prepoznao najviše ispravno pozitivnih vrijednosti za geste „*swipe* lijevo“ (88%), „*swipe* desno“ (88%) i „*dislike*“ (88%). Najviše neispravno pozitivnih vrijednosti imaju geste „*like*“ (48%), „*swipe* gore“ (60%), „*swipe* dolje“ (64%). Slično kao i kod testiranja na osobnom računalu, kod gesti „swipe *gore*“ i „*swipe* dolje“ dolazi do zabune jer je način pokreta ruke i prstiju, nakon što se izvrši jedna od ovih dviju gesti, sličan. Najviše neispravno pozitivnih vrijednosti ima gesta „*like*“. Od sveukupno 50 ponavljanja te geste, model je točno prepoznao gestu „*like*“ samo u 24 slučaja. Ostale geste imaju sličan broj ispravno pozitivnih vrijednosti kao i u slučaju testiranja na osobnom računalu. Postotak ispravno pozitivnih vrijednosti gesta je sljedeći: „*dislike*“ (88%), „*swipe* lijevo“ (88%), „*swipe* desno“ (88%), „zaustavi/pokreni“ (80%), „pojačaj“ (80%), „smanji“ (72%), „nema geste“ (68%), „*swipe* dolje“ (62%), „*swipe* gore“ (60%), „*like*“ (48%),

UKUPNO	nema geste	zaustavi/pokreni	swipe dolje	swipe gore	swipe lijevo	swipe desno	like	dislike	pojačaj	smanji
nema geste	**34**	0	0	0	0	0	1	0	0	1
zaustavi/pokreni	2	**40**	1	3	0	0	11	0	1	2
swipe dolje	2	2	**32**	8	0	0	1	0	1	2
swipe gore	0	0	2	**30**	0	0	2	0	0	0
swipe lijevo	2	0	1	0	**44**	0	3	5	1	1
swipe desno	0	0	0	0	0	**44**	0	0	0	0
like	0	0	0	1	1	1	**24**	0	0	0
dislike	2	2	11	6	1	0	2	**44**	2	2
pojačaj	6	4	1	0	1	3	4	1	**40**	6
smanji	2	2	2	2	3	2	2	0	5	**36**
∑	50	50	50	50	50	50	50	50	50	50

Slika 4.7. Matrica zabune ponovno istreniranog LRCN modela kojeg je testiralo deset nasumičnih korisnika na *Raspberry Pi 3* ugradbenoj platformi

5. ZAKLJUČAK

Kontroliranje *Spotify* aplikacije za reproduciranje glazbe pomoću pokreta ruku i gesti prstiju inovativan je način na koji korisnici komuniciraju s uslugom strujanja (engl. *streaming*) glazbe. Da bi se postiglo ovo rješenje primijenjena je metoda nadziranog strojnog učenja kojom je istrenirana dugoročna rekurentna konvolucijska mreža (engl. *Long-term Recurrent Convolutional Network, LRCN*) na bazi *20BN-Jester* video sekvenci. Istrenirana LRCN mreža omogućava dinamičku detekciju gesti prstiju. Implementirano je deset klasa (gesti) kojima korisnik može upravljati *Spotify* glazbenim servisom. Model je nakon primjene prijenosa učenja (engl. *transfer learning*) postigao točnost od 92,14% na testnom skupu video sekvenci. Evaluacija same mreže provedena je pomoću matrica zabune. Za dodatnu evaluaciju napisana je testna aplikacija u *Python* programskom jeziku, u kojoj osoba može testirati model u stvarnom vremenu. Nakon detekcije gesta se ispisuje na prozor kamere pred kojom osoba izvodi gestu. Za potrebe evaluacije rada modela LRCN mreže pozvano je 10 nasumičnih osoba za testiraje rada neuronske mreže. Nakon testiranja model je postigao točnost od 80,2%. Svaka je osoba izvela sve geste pet puta. Od 500 izvedenih gesti, točno je detektirana 401 gesta. Pri detekciji gesti u stvarnom vremenu javljaju se neki od sljedećih problema koji mogu uzorkovati pogrešnu detekciju: pojavljivanje drugih pokreta u kadru kamere kao što su pokreti ostalih ljudi koji se pojavljuju u kadru ili spontani pokreti osobe koja izvodi gestu pri čemu taj pokret može omesti pravilnu detekciju geste. Integracijom modela LRCN neuronske mreže sa *Spotify* web API-jem stvorena je aplikacija koja omogućuje bežično upravljanje *Spotify* servisom izvedbom gesti prstiju ispred USB kamere. Rješenje je implementirano na ugradbenu platformu *Raspberry Pi 3* na kojoj je model nakon testiranja u realnim uvjetima postigao točnost od 73,6%.

SAŽETAK

U ovome radu predložen je sustav za upravljanje *Spotify* aplikacijom gestama prstiju. Za prepoznavanje gesti prstiju korištena je dugoročna rekurentna konvolucijska mreža (LRCN) istrenirana na skupu *20BN-Jester* video sekvenci. LRCN model kombinira konvolucijsku neuronsku mrežu (CNN) i rekurentnu neuronsku mrežu (RNN), u ovome slučaju dugotrajno kratko pamćenje (LSTM jedinice), u svrhu učinkovite analize prostornih i vremenskih značajki u sekvencama pokreta. Implementirano je prepoznavanje sljedećih gesti koje korisnici mogu izvoditi: „*swipe* gore“ za postavljanje glasnoće zvuka na 100%, „*swipe* dolje“ za postavljanje glasnoće zvuka na 50% , „*swipe* lijevo“ za prebacivanje na prethodnu pjesmu u *playlisti*, „*swipe* desno“ za prebacivanje na sljedeću pjesmu u *playlisti*, „*like*“ za dodavanje pjesme u omiljeni sadržaj, „*dislike*“ za odbacivanje pjesme iz omiljenog sadržaja, „zaustavi/pokreni“ za zaustaviti ili pokrenuti pjesmu, „pojačaj“ za pojačavanje glasnoće zvuka za 5% te gesta „smanji“ za smanjivanje glasnoće zvuka za 5%. Osim ovih 9 gesti implementirana je gesta „nema geste“ kako bi model naučio situacije u kojima je osoba statična pred kamerom. Točnost LRCN modela na kojem je primijenjena metoda prijenosa učenja iznosi 92,14%. Evaluacija LRCN modela te evaluacija rada aplikacije za kontrolu *Spotify* glazbenog servisa temelji se na matricama zabune. Matrica zabune testne aplikacije za kontrolu *Spotfiy* servisa stvorena je na temelju rezultata prepoznavanja gesti koje je izvelo deset nasumičnih korisnika aplikacije. Točnost pri prepoznavanja gesti u stvarnom vremenu testirana na deset korisnika na osobnom računalu iznosi 80,2%, dok na ugradbenoj platformi *Raspberry Pi 3* točnost iznosi 73,6%.

Ključne riječi: Spotify**,** dinamička detekcija gesti prstiju, strojno učenje, klasifikacija, LRCN model, 20BN-Jester, Raspberry Pi

CONTROLLING THE DEVICE USING THE METHOD OF RECOGNIZING FINGER GESTURES

ABSTRACT

In this work, a system for controlling the Spotify application using finger gestures is proposed. A Long Short-Term Memory Recurrent Convolutional Network (LRCN) trained on the 20BN-Jester video sequences dataset was employed for finger gesture recognition. The LRCN model combines Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN), specifically Long Short-Term Memory (LSTM) units, for efficient analysis of spatial and temporal features in motion sequences. The implemented system recognizes the following gestures that users can perform: „swipe up“ to set the volume to 100%, „swipe down“ to set the volume to 50%, „swipe left“ to switch to the previous song in the playlist, „swipe right“ to switch to the next song in the playlist, „like“ to add a song to favorites, „dislike“ to remove a song from favorites, „pause-play“ to pause or play the song, „volume up“ to increase the volume by 5% and „volume down“ to decrease the volume by 5%. In addition to these 9 gestures, a „no gesture“ gesture has been implemented to allow the model to learn situations in which a person remains static in front of the camera. The accuracy of the LRCN model with transfer learning applied is 92,14%. The evaluation of the LRCN model and the evaluation of the Spotify control application are based on confusion matrices. The confusion matrix for Spotify control application was created based on the gesture recognition results performed by ten random application users. Real-time gesture recognition accuracy, tested on ten users on a personal computer, is 80,2%, while on the Raspberry Pi 3 embedded platform, the accuracy is 73,6%.

Keywords: Spotify, dynamic detection of finger gestures, machine learning, classification, LRCN model, 20BN-Jester, Raspberry Pi

LITERATURA

[1] X. Chu, J. Liu, i S. Shimamoto, „A Sensor-Based Hand Gesture Recognition System for Japanese Sign Language“, u *2021 IEEE 3rd Global Conference on Life Sciences and Technologies (LifeTech)*, Nara, Japan: IEEE, ožu. 2021, str. 311–312. doi: 10.1109/LifeTech52111.2021.9391981.

[2] D. Rodrigues i C. Li, „Hand Gesture Recognition Using FMCW Radar in Multi-Person Scenario“, u *2021 IEEE Topical Conference on Wireless Sensors and Sensor Networks (WiSNeT)*, San Diego, CA, USA: IEEE, sij. 2021, str. 50–52. doi: 10.1109/WiSNeT51848.2021.9413794.

[3] E. A. Ibrahim, M. Geilen, J. Huisken, M. Li, i J. P. De Gyvez, „Low Complexity Multi-directional In-Air Ultrasonic Gesture Recognition Using a TCN“, u *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Grenoble, France: IEEE, ožu. 2020, str. 1259–1264. doi: 10.23919/DATE48585.2020.9116482.

[4] H.-K. Chen i C.-C. Chuang, „A Gesture Controlled Music Playback System Using Convolutional Neural Network“, u *2020 International Symposium on Computer, Consumer and Control (IS3C)*, Taichung City, Taiwan: IEEE, stu. 2020, str. 13–16. doi: 10.1109/IS3C50286.2020.00010.

[5] I. Dhall, S. Vashisth, i G. Aggarwal, „Automated Hand Gesture Recognition using a Deep Convolutional Neural Network model“, u *2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, Noida, India: IEEE, sij. 2020, str. 811–816. doi: 10.1109/Confluence47617.2020.9057853.

[6] M. Ur Rehman *i ostali*, „Dynamic Hand Gesture Recognition Using 3D-CNN and LSTM Networks“, *Comput. Mater. Contin.*, sv. 70, izd. 3, str. 4675–4690, 2022, doi: 10.32604/cmc.2022.019586.

[7] J. Materzynska, G. Berger, I. Bax, i R. Memisevic, „The Jester Dataset: A Large-Scale Video Dataset of Human Gestures“, u *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, Seoul, Korea (South): IEEE, lis. 2019, str. 2874–2882. doi: 10.1109/ICCVW.2019.00349.

[8] L. Gionfrida, W. M. R. Rusli, A. E. Kedgley, i A. A. Bharath, „A 3DCNN-LSTM Multi-Class Temporal Segmentation for Hand Gesture Recognition“, *Electronics*, sv. 11, izd. 15, str. 2427, kol. 2022, doi: 10.3390/electronics11152427.

[9] W. Zhang, J. Wang, i F. Lan, „Dynamic hand gesture recognition based on short-term sampling neural networks“, *IEEECAA J. Autom. Sin.*, sv. 8, izd. 1, str. 110–120, sij. 2021, doi: 10.1109/JAS.2020.1003465.

[10] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, i L.-C. Chen, „MobileNetV2: Inverted Residuals and Linear Bottlenecks“. arXiv, 21. ožujak 2019. Pristupljeno: 28. srpanj 2023. [Na internetu]. Dostupno na: http://arxiv.org/abs/1801.04381

[11] K. He, X. Zhang, S. Ren, i J. Sun, „Deep Residual Learning for Image Recognition“. arXiv, 10. prosinac 2015. Pristupljeno: 02. rujan 2023. [Na internetu]. Dostupno na: http://arxiv.org/abs/1512.03385

[12] „N. Bolf, i Bolf (ur.), N. (2021). Osvježimo znanje: Strojno učenje. Kemija u industriji, 70 (9-10), 591-593, dostupno na: https://hrcak.srce.hr/263495 [24.6.2023.]“.

[13] „„Create production-grade machine learning models with TensorFlow‘, dostupno na: https://www.tensorflow.org/ [25.6.2023.]“.

[14] „„Keras: Deep Learning for humans‘, dostupno na: https://www.tensorflow.org/ [25.6.2023.]“.

[15] „„NumPy‘, dostupno na: https://numpy.org/ [25.6.2023.]“.

[16] „‚Scikit-learn: machine learning in Python‘, dostupno na: https://scikit-learn.org/stable/ [25.6.2023.]“.
[17] „‚OpenCV - Open Computer Vision Library‘, dostupno na: https://scikit-learn.org/stable/ [25.6.2023.]“.
[18] „Theano Python biblioteka, dostupno na: https://scikit-learn.org/stable/ [25.6.2023.]“.
[19] „‚20bn-jester, Jester Dataset V1 for Hand Gesture Recognition‘, dostupno na: https://www.kaggle.com/datasets/toxicmender/20bn-jester [26.6.2023.]“.
[20] J. Donahue *i ostali*, „Long-term Recurrent Convolutional Networks for Visual Recognition and Description“. arXiv, 31. svibanj 2016. Pristupljeno: 28. srpanj 2023. [Na internetu]. Dostupno na: http://arxiv.org/abs/1411.4389
[21] C. Rao i Y. Liu, „Three-dimensional convolutional neural network (3D-CNN) for heterogeneous material homogenization“. arXiv, 14. veljača 2020. Pristupljeno: 28. srpanj 2023. [Na internetu]. Dostupno na: http://arxiv.org/abs/2002.07600
[22] S. Ioffe i C. Szegedy, „Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift“. arXiv, 02. ožujak 2015. Pristupljeno: 02. rujan 2023. [Na internetu]. Dostupno na: http://arxiv.org/abs/1502.03167
[23] A. Ray i H. Ray, „Study of Overfitting through Activation Functions as a Hyper-parameter for Image Clothing Classification using Neural Network“, u *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, Kharagpur, India: IEEE, srp. 2021, str. 1–5. doi: 10.1109/ICCCNT51525.2021.9580022.
[24] Z. Zhang, „Improved Adam Optimizer for Deep Neural Networks“, u *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*, Banff, AB, Canada: IEEE, lip. 2018, str. 1–2. doi: 10.1109/IWQoS.2018.8624183.
[25] J. Konar, P. Khandelwal, i R. Tripathi, „Comparison of Various Learning Rate Scheduling Techniques on Convolutional Neural Network“, u *2020 IEEE International Students' Conference on Electrical,Electronics and Computer Science (SCEECS)*, Bhopal, India: IEEE, velj. 2020, str. 1–5. doi: 10.1109/SCEECS48394.2020.94.
[26] J. N. Saeed, A. M. Abdulazeez, i D. A. Ibrahim, „2D Facial Images Attractiveness Assessment Based on Transfer Learning of Deep Convolutional Neural Networks“, u *2022 4th International Conference on Advanced Science and Engineering (ICOASE)*, Zakho, Iraq: IEEE, ruj. 2022, str. 13–18. doi: 10.1109/ICOASE56293.2022.10075585.
[27] „‚tkinter – Python interface to Tcl/Tk‘, dostupno na: https://docs.python.org/3/library/tkinter.html [7.7.2023.]“.
[28] „‚Welcome to Spotipy!‘ , dostupno na: https://spotipy.readthedocs.io/en/2.22.1/ [7.7.2023.]“.
[29] „‚Find, install and publish Python packages with the Python Package Index‘, dostupno na: https://pypi.org/project/requests/ [7.7.2023.]“.
[30] „‚Spotify for Developers: Home‘ , dostupno na: https://developer.spotify.com/ [7.7.2023.]“.
[31] „"Miniconda – conda documentation“, dostupno na: https://docs.conda.io/en/latest/miniconda.html [10.7.2023.]“.
[32] „N. Beheshti, ‚Guide to Confusion Matrices & Classification Performance Metrics‘, Towards Data Science, 2022, dostupno na: https://towardsdatascience.com/guide-to-confusion-matrices-classification-performance-metrics-a0ebfc08408e [5.7.2023.]“.

ŽIVOTOPIS

Lovro Čubaković rođen je u Zagrebu 21. siječnja 1999. Osnovnu školu Davorina Trstenjaka završava u Hrvatskoj Kostajnici 2013., a Srednju školu Ivana Trnskoga u Hrvatskoj Kostajnici, smjer Opća gimnazija, završava 2017. Nakon srednje škole upisuje sveučilišni preddiplomski studij elektrotehnike na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku. Na 2. godini studija upisuje smjer Komunikacije i informatika. Preddiplomski studij elektrotehnike završava 2021. te upisuje sveučilišni diplomski studij Elektrotehnika, smjer Komunikacije i informatika, izborni blok Mrežne tehnologije. Od 2021. volontira za međunarodnu studentsku udrugu IAESTE, gdje se kao koordinator aktivno služi engleskim jezikom. Na završnoj godini diplomskog studija postaje stipendist tvrtke TTTechAuto d.o.o.

PRILOZI

Prilog P.3.1. Tablica video okvira u kojima osobe prikazuju način izvođenja svake implementirane geste (preuzeto iz baze *20BN-Jester)*

Naziv geste	Način izvođenja geste
Dislike	
Like	

Nema geste

Pojačaj

Smanji

Swipe desno

Swipe dolje

Swipe gore

Swipe lijevo

Zaustavi/pokreni

Prilog P.3.2. *Python* programski kod kojim je definiran model neuronske mreže

```
model = Sequential()
model.add(Conv3D(16,(3,3,3),input_shape=(patch_size, img_cols, img_rows,
3),activation='relu'))
model.add(Conv3D(16,(3,3,3),strides=(1,1,1),padding='same',dilation_rate=(1
,1,1), kernel_initializer='he_normal',kernel_regularizer=l2(weight_decay),
use_bias=False,name='Conv3D_2a_a', activation = 'relu'))
model.add(MaxPooling3D(pool_size=(2,2,2)))
model.add(Dropout(0.25))
model.add(Conv3D(32,(3,3,3),strides=(1,1,1),padding='same',dilation_rate=(1
,1,1), kernel_initializer='he_normal',kernel_regularizer=l2(weight_decay),
use_bias=False,name='Conv3D_2b_a', activation = 'relu'))
model.add(Dropout(0.25))
model.add(Conv3D(32,(3,3,3),strides=(1,1,1),padding='same',dilation_rate=(1
,1,1), kernel_initializer='he_normal',kernel_regularizer=l2(weight_decay),
use_bias=False,name='Conv3D_2b_b', activation = 'relu'))
model.add(MaxPooling3D(pool_size=(1, 2,2)))
model.add(Dropout(0.25))
model.add(Conv3D(64,(3,3,3),strides=(1,1,1),padding='same',dilation_rate=(1
,1,1), kernel_initializer='he_normal',kernel_regularizer=l2(weight_decay),
use_bias=False, name='Conv3D_2c_a', activation = 'relu'))
model.add(Dropout(0.25))
model.add(Conv3D(64,(3,3,3),strides=(1,1,1),padding='same',dilation_rate=(1
,1,1), kernel_initializer='he_normal',kernel_regularizer=l2(weight_decay),
use_bias=False,name='Conv3D_2c_b', activation = 'relu'))
model.add(Dropout(0.25))
model.add(Conv3D(64,(3,3,3),strides=(1,1,1),padding='same',dilation_rate=(1
,1,1), kernel_initializer='he_normal',kernel_regularizer=l2(weight_decay),
use_bias=False,name='Conv3D_2c_c', activation = 'relu'))
model.add(MaxPooling3D(pool_size=(1, 2,2)))
model.add(Dropout(0.25))
model.add(Conv3D(128,(3,3,3),strides=(1,1,1),padding='same',dilation_rate=(
1,1,1), kernel_initializer='he_normal',kernel_regularizer=l2(weight_decay),
use_bias=False,name='Conv3D_2d_a', activation = 'relu'))
model.add(Dropout(0.25))
model.add(Conv3D(128,(3,3,3),strides=(1,1,1),padding='same',dilation_rate=(
1,1,1), kernel_initializer='he_normal',kernel_regularizer=l2(weight_decay),
use_bias=False,name='Conv3D_2d_b', activation = 'relu'))
model.add(Dropout(0.25))
model.add(Conv3D(128,(3,3,3),strides=(1,1,1),padding='same',dilation_rate=(
1,1,1), kernel_initializer='he_normal',kernel_regularizer=l2(weight_decay),
use_bias=False,name='Conv3D_2d_c', activation = 'relu'))
model.add(MaxPooling3D(pool_size=(1, 2, 2)))
model.add(Dropout(0.25))
model.add(ConvLSTM2D(filters=64,kernel_size=(3,3),strides=(1,1),padding='sa
me',kernel_initializer='he_normal',
recurrent_initializer='he_normal',kernel_regularizer=l2(weight_decay),
recurrent_regularizer=l2(weight_decay),return_sequences=True,name='gatedcls
tm2d_2'))
model.add(ConvLSTM2D(filters=64,kernel_size=(3,3),strides=(1,1),padding='sa
me',kernel_initializer='he_normal',recurrent_initializer='he_normal',kernel
_regularizer=l2(weight_decay),recurrent_regularizer=l2(weight_decay),return
_sequences=True, name='gatedclstm2d_3'))
model.add(GlobalAveragePooling3D())
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(nb_classes,kernel_initializer='normal'))
model.add(Activation('softmax'))
```

Prilog P.3.3. Primjer koda za autentikaciju i postavljanje područja unutar kojeg je korištena *Spotify* aplikacija

```
CLIENT_ID = 'xxxxxxxxxxxxxxxxxxxxxxxxxx'
CLIENT_SECRET = 'xxxxxxxxxxxxxxxxxxxxxxxxx'
SPOTIPY_REDIRECT_URI = 'http://localhost:8888/spotify-api/callback/'


SCOPE = 'user-modify-playback-state user-read-playback-state user-library-
read user-library-modify user-read-currently-playing'
class SpotifyAPI:
    """konekcija"""

    def __init__(self, token: str = "") -> None:
        self.token = ""
        self.client_id = CLIENT_ID
        self.client_secret = CLIENT_SECRET
        self.redirect_uri = "http://localhost:8888/spotify-api/callback/"
        self.token = token
        self.headers = {
            "Authorization": f"Bearer {self.token}"
        }

    def get_token(self) -> str:
        """dohvati token i autoriziraj"""
        print("Waiting...")
        try:
            oauth = SpotifyOAuth(client_id=self.client_id,
client_secret=self.client_secret,
                                 redirect_uri=self.redirect_uri,
scope=SCOPE, show_dialog=True)
            self.token = oauth.get_access_token(as_dict=False,
check_cache=False)
            print(self.token)
            print("Successful, let's use this app!")
            return self.token
        except Exception as e:
            print(f"Exception: {e}")
            print("Authentication failed, please try again.")
```

Prilog P.3.4. Primjer koda za definiranje i pozivanje funkcija za prebacivanje pjesama u slučaju uspješne detekcije geste

```
def skip_to_previous(self) -> None:
        try:
            endpoint = "https://api.spotify.com/v1/me/player/previous"
            requests.post(endpoint, headers=self.headers)
        except Exception as e:
            print(f"Exception: {e}")

def skip_to_next(self) -> None:
        try:
            endpoint = "https://api.spotify.com/v1/me/player/next"
            requests.post(endpoint, headers=self.headers)
        except Exception as e:
            print(f"Exception: {e}")


def gesture_action(self, gesture: str) -> None:
        """
        pozvati prilikom detekcije
        ovisno o gesti napravi akciju
        """
        if gesture == "swipe lijevo":
            self.skip_to_next()
        elif gesture == "swipe desno":
            self.skip_to_previous()
        else:
            pass
```

Prilog P.3.5. Primjer koda za obradu video okvira pri detekciji geste

```
def normaliz_data(np_data):
    input = np.array(np_data)
    X_tr = []
    X_train = X_tr.append(np_data)
    X_tr.append(input)
    X_train = np.array(X_tr)
    train_set = np.zeros((1, 25, img_cols, img_rows, 3))
    train_set[0][:][:][:][:] = X_train[0, :, :, :, :]
    train_set = train_set.astype('float32')
    train_set -= np.mean(train_set)
    train_set /= np.max(train_set)
    return train set
```

Prilog P.4.1. Primjer koda za otvaranje prozora USB kamere, detekciju geste te ispisivanje geste nakon uspješne detekcije geste

```
to_predict = []
num_frames = 0
cap = cv2.VideoCapture(0)
classe =''

while(True):
    ret, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
if (len(to_predict) == 25):
        frame_to_predict = np.array(to_predict, dtype=np.float32)
        frame_to_predict = normaliz_data(frame_to_predict)
        predict = model.predict(frame_to_predict)
        classe = classes[np.argmax(predict)]
         to_predict = []
        if(np.amax(predict)*100 >= 99):
            print('Classe = ',classe, 'Precision =
'np.amax(predict)*100,'%')
    if(np.amax(predict)*100 >= 99):
    cv2.putText(frame, classe, (50, 60), cv2.FONT_HERSHEY_SIMPLEX, 0.9,
(0, 0, 250),1,cv2.LINE_AA)
    cv2.imshow('Hand Gesture Recognition',frame)
     skip_frame_number = 25
    cap.set(cv2.CAP_PROP_POS_FRAMES, skip_frame_number)
    success, frame = cap.read()
     if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```