Elektrotehnički fakultet
Univerzitet u Banjoj Luci

PREPOZNAVANJE GOVORA U SVRHU UPRAVLJANJA PAMETNOM KUĆOM

IZVJEŠTAJ PROJEKTNOG ZADATKA

iz predmeta

MULTIMEDIJALNI SIGNALI I SISTEMI

Student:
Milan Medić 1135/16

Mentori:
prof. dr Vladimir Risojević
dipl. inž. Vladan Stojnić

Jul 2020. godine

Sadržaj:

1. Uvod

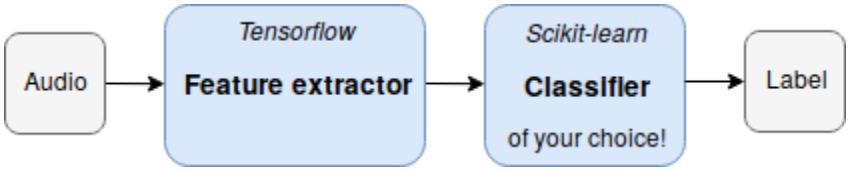
Prepoznavanje govora je oblast kojom se inženjeri bave od početka 50-ih godina prošlog vijeka. Još 1952 godine istraživači iz *Bell Telephone Laboratories* centra su uspjeli da naprave sistem koji prepoznaje izgovore cifara jednog govornika koristeći prepoznavanje formanta u spektrogramu snage izgovorene riječi. Od tada prepoznavanje govora se može podijeliti u tri epohe. Prva epoha obuhvata period do 1970. godine gdje je za prepoznavanje govora korišten algoritam dinamičkog vremenskog savijanja koji je upoređivao sličnosti između dva kratka frejma govora. Druga epoha zauzima period od 1970. do 1990. i tokom nje dolazi do većeg napretka na polju prepoznavanja govora. Sredinom 80-ih godina prošlog vijeka počinje se koristiti sakriveni Markovljev model (HMM – *Hidden Markov Model*) za prepoznavanje govora gdje se on implementira softverski ali i hardverski te omogućuje popriličan stepen prepoznavanja govora i početak praktičnih primjena. Treća epoha obuhvata period od 1990. pa do danas. Tokom ove epohe prepoznavanje govora je dobilo i komercijalni karakter, tokom 90-ih mnoge telefonske kompanije su nudile usluge glasovnog biranja broja ili glasovnog biranja korisnika. Početkom 2000-ih i ekspanzijom komercijalnih računara, prepoznavanje govora i glasovno upravljanje ponovo dobija na značaju. Sve do 2009. godine za prepoznavanje govora isključivo je korišten sakriveni Markovljev model, a od 2009. godine se počinju koristiti neuronske mreže i duboko učenje za prepoznavanje govora. Rezultati postignuti sa neuronskim mrežama su bili lošiji od rezultata dobijenih sa Markovljevim modelom sve do 2015. godine kada je *Google* objavio da je postigao skok u tačnosti od 46% (zbog ogromne količine glasovnih podataka na raspolaganju) nakon čega su neuronske mreže preuzele primar u oblasti prepoznavanja govora i potisnule HMM.

Trenutna rješenja za prepoznavanje govora (poput *Google assistent*-a) su nezavisna od govornika i nude veoma dobre rezultate, ali su ograničena jezikom govora (u *offline* režimu rada za sada su podržani samo veliki svjetski jezici) i u velikom broju slučajeva naplaćuju usluge po sekundama govora (*Google Cloud)*, te skladište vaše izgovorene riječi za buduće treniranje.

Predmet izrade ovog projektnog zadatka je sistem, u *offline* režimu rada, koji prepoznaje izgovorene komande na srpskom jeziku za upravljanje kućom. Broj komandi je ograničen i unaprijed definisan kao i njihova struktura koja se sastoji od tri riječi (jedne ključne riječi – „kućo“ i dvije riječi koje predstavljaju komandu). Za prepoznavanje govora u ovom projektnom zadatku korištene su unaprijed istrenirane *open-source* neuronske mreže za prepoznavanje muzičkih žanrova iz biblioteke *sklearn-audio-transfer* kao ekstraktori obilježja, te klasifikatori koji su obučavani na obilježjima dobijenim od datih mreža. Kao baza riječi (*dataset*) iskorišteni su glasovi 21 osobe (jedanaest muških i deset ženskih glasova) koji izgovaraju deset riječi po pet puta na ijekavskom izgovoru. Dati sistem je implementiran na *Jetson-nano* platformi, gdje se nakon pokretanja aplikacije izgovori komanda u eksterni mikrofon i na ekranu se ispišu riječi koje je sistem prepoznao. Rezultati postignuti sa ovako implementiranim sistemom su zadovoljavajući i daju oko 90% tačnosti na evaluacionim podacima korištenjem *OpenL3* ekstraktora obilježja.

2. *Sklearn-audio-transfer-learning* biblioteka i ekstraktori obilježja

Kao baza ovog rada iskorištena je *sklear-audio-transfer-learning*[1] biblioteka. Ova biblioteka je *open-source* i predstavlja svojevrsan alat za jednostavno i efikasno modelovanje *proof-of-concept* modela za prepoznavanje muzičkih žanrova, te upoznavanje sa *Tensorflow* i *sklearn* bibliotekama. Princip rada ove biblioteke je prikazan na slici 2.1. Kao ekstraktor obilježja se koristi jedna od pretreniranih mreža *VGGish*[2], *OpenL3*[3] ili *musicnn*[4], kao klasifikator se može iskoristiti jedan od ponuđenih *sklearn* klasifikatora ili napraviti zasebni klasifikator (nekoliko slojeve potpuno povezane neuronske mreže npr.).



Slika 2.1 – *princip rada sklearn-audio-transfer biblioteke*

Tokom izrade ovog rada, kao ekstraktori obilježja korištene su *VGGish* i *OpenL3* mreže. *Musicnn* mreža nije korištena jer za ispravno funkcionisanje potražuje noviju verziju (2.0) *Tensorflow* biblioteke koja nije kompatibilna sa starijim verzijama (konkretno 1.13.1) koja je potrebna za funkcionisanje prve dvije mreže, osim tiga iz [5] se jasno može vidjeti da mreža *openl3* konzistentno daje bolje rezultate od *musicnn* nad istim podacima te je i to bio jedan od razloga zbog kojih se nije pristupilo rješavanju ovog ograničenja. Kao klasifikator je izabran SVM sa polinomskim kernelom. Detalji oko izbora klasifikatora i mreže za ekstrakciju obilježja biće detaljno objašnjeni u poglavlju 4.

2.1. VGGish mreža

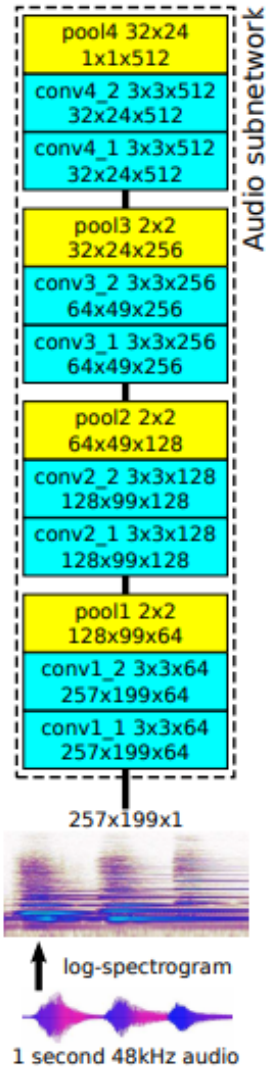
Ova mreža je trenirana nad skupom podataka od dva miliona audio isječaka trajanja 10 sekundi raspoređenih u 600 klasa. *AudioSet* koji je iskorišten za treniranje ove mreže datira iz 2017. godine i može se pronaći na [6]. Arhitektura ove mreže je prikazana na slici 2.2.

input (96 × 64)
conv3-64
maxpool
conv3-128
maxpool
conv3-256
maxpool
conv3-512
maxpool
FC-4096
FC-4096
FC-128

Slika 2.2 – *Arhitektura VGG-ish mreže*

2.2. OpenL3 mreža

OpenL3 mreža se sastoji od dvije podmreže i jednog sloja koji spaja obilježja iz datih podmreža. Jedna podmreža se koristi za ekstrakciju obilježja iz video ulaza, druga mreža za ekstrakciju obilježja iz audio izlaza te se zatim vrši upoređivanje da li audio zapis odgovara video zapisu. Ovakav pristup se naziva *Look, listen and learn*[7] a originalna mreža je trenirana nad dva skupa podataka *FlickerSound* koji se sastoji od 500 000 video klipova trajanja 10 sekundi i *Kinectic-Sound* koji se sastoji 15 000 video klipova trajanja 10 sekundi. Ulaz u audio ekstraktor obilježja je mel-spektrogram koji sadrži 257 frekvencijskih opsega i ima 199 preklapajućih prozora. U kasnijim verzijama, ovaj ekstraktor je treniran nad *AudioSet*-om i dodatnim skupovima podataka sa ambijentalnim zvukovima pri čemu je davao veoma dobre rezultate (oko 80%).



Slika 2.3 – *Arhitektura audio ekstraktora OpenL3 mreže*

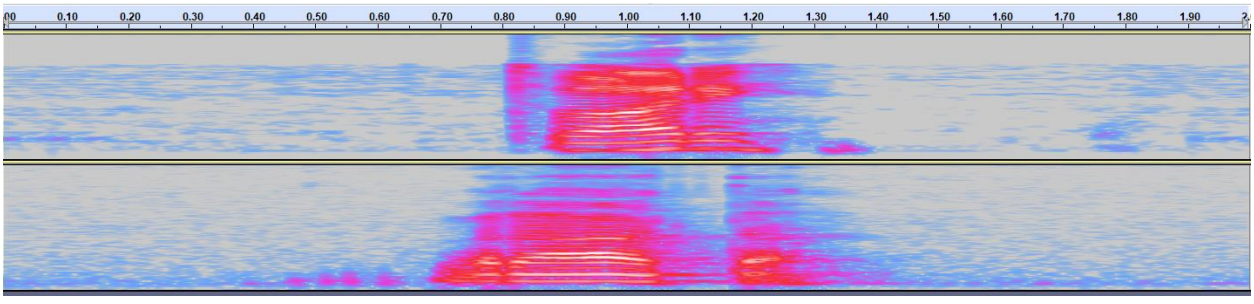
3. Baza riječi (*dataset*)

Kao baza riječi, odnosno *dataset* iskorišteni su glasovi 21 osobe (jedanaest muških i deset ženskih) koje su snimljene kako izgovaraju deset riječi prikazanih u tabeli 3.1 po pet puta. Svi glasovi su snimljeni na srpskom jeziku, sa ijekavskim naglaskom na različitim uređajima da bi se smanjio uticaj favorizacije određenog mikrofona, odnosno uređaja za snimanje. Nakon snimanja riječi, one su obrađene i standardizovane korištenjem programa *Audacity*. Riječi su isječene tako da traju dvije sekunde, sa frekvencijom odmjeravanja od 22050 Hz nakon čega su spremne za daljnu manipulaciju.

Tabela 3.1 – *Baza riječi*

Rd. broj	Riječ	Rd. broj	Riječ
1.	kućo	**6.**	vrata
2.	otvori	**7.**	prozor
3.	zatvori	**8.**	grijanje
4.	uključi	**9.**	hladjenje
5.	isključi	**10.**	svjetlo

Ekstrakcija obilježja iz ovako generisanih riječi, bez bilo kakve obrade se pokazala kao veoma loš izbor jer sve riječi imaju dugačke periode tišine, a kako su ulazi u ekstraktore obilježja Mel-spektrogrami koji zavise i od vremena a ne samo frekvencije signala, sličnost između dvije različite riječi bi bila veoma velika. Ovo se može vidjeti i na slici 3.1. gdje su prikazani mel-spektrogrami dvije različite riječi (otvori i kućo) kod kojih tišina zauzima preko 60%. Da bi se ovaj problem uklonio, vršene su različite manipulacije nad *dataset*-om nabrojane u tabeli 3.2 a detaljno objašnjene u narednom poglavlju kao i njihov uticaj na tačnost klasifikacije riječi.



Slika 3.1. – *Mel-spektrogram riječi „kućo“ i „otvori“*

Tabela 3.2. – *Vrste obrada riječi unutar dataset-a*[1]

Rd. broj	Naziv	Opis
1.	cutoff_1s	Riječ je odsječena da je govor centriran na sredini fajla, dopunjen nulama sa obe strane.
2.	cutoff_2s	Riječ je odsječena da je govor centriran na sredini fajla, dopunjen nulama sa obe strane. Ovo je originalni *dataset*.
3.	repeat_1s	Dio sa govorom je odsječen i zatim konkateniran da se popuni trajanje od jedne sekunde.
4.	repeat_2s	Dio sa govorom je odsječen i zatim konkateniran da se popuni trajanje od dvije sekunde.
5.	repeat_4s	Dio sa govorom je odsječen i zatim konkateniran da se popuni trajanje od četiri sekunde.
6.	filtered_cutoff_2s	*Dataset* cutoff_2s je filtriran niskopropusnim Čebiševljevim filtrom sa graničnom frekvencijom 5 kHz.
7.	filtered_repeat_1s	*Dataset* repeat_1s je filtriran niskopropusnim Čebiševljevim filtrom sa graničnom frekvencijom 5 kHz.
8.	filtered_repeat_2s	*Dataset* repeat_2s je filtriran niskopropusnim Čebiševljevim filtrom sa graničnom frekvencijom 5 kHz..

[1] Unutar tabele su samo nabrojane promjene nad *detaset*-om, jer ovo je jedan od hiperparametara mreže te detaljniji opis kao i rezultati se nalaze u poglavlju 4

4. Izbor hiperparametara i testiranje mreže

U ovom poglavlju će detaljno biti objašnjen izbor svih hiperparametara mreže, njihov uticaj na tačnost prepoznavanja govora i zaključci dobijeni iz tih rezultata.

4.1. Izbor ekstraktora obilježja, klasifikatora i *dataset*-a

Kao što je u prethodnim poglavljima rečeno, biblioteka *skleran-audio-transfer-learning* nudi mogućnost izbora tri već istrenirane mreže kao ekstraktore obilježja. Zbog nekompatibilnosti *Tensorflow 2.x* verzija sa prethnodnim *1.x* verzijama, mreža *musicnn* nije korištena u ovom radu. Na slikama 4.1 i 4.2 se mogu vidjeti vrijednosti kros-validacije dobijene koristeći *VGGish* i *OpenL3* ekstraktore respektivno, nad *cutoff_2s dataset*-om pri čemu su iskorišteni svi klasifikatori i rađena je redukcija dimenzionalnosti obilježja uz pomoć analize glavnih komponenti (PCA[2]). Korištena je ugrađena funkcija *cross-val-score* iz biblioteke *sklearn* sa desetorostrukim preklapanjem za računanje preciznosti i standardne devijacije rezultata.

Ekstraktor	Klasifikator	Dataset	PCA	Acc. ± (std.)
vggish	kNN	cutoff_2s	64	0.5 ± (0.06)
vggish	kNN	cutoff_2s	0	0.44 ± (0.06)
vggish	SVM	cutoff_2s	64	0.43 ± (0.07)
vggish	SVM	cutoff_2s	0	0.36 ± (0.08)
vggish	SVM	cutoff_2s	128	0.35 ± (0.07)
vggish	kNN	cutoff_2s	128	0.35 ± (0.05)
vggish	linearSVM	cutoff_2s	64	0.34 ± (0.04)
vggish	perceptron	cutoff_2s	0	0.34 ± (0.05)
vggish	linearSVM	cutoff_2s	0	0.34 ± (0.05)
vggish	linearSVM	cutoff_2s	128	0.33 ± (0.03)
vggish	perceptron	cutoff_2s	128	0.33 ± (0.04)
vggish	MLP	cutoff_2s	128	0.33 ± (0.04)
vggish	perceptron	cutoff_2s	64	0.32 ± (0.04)
vggish	MLP	cutoff_2s	64	0.32 ± (0.03)
vggish	MLP	cutoff_2s	0	0.24 ± (0.05)

Slika 4.1 – *Vrijednosti kros-validacije korištenjem VGGish ekstraktora obilježja*

Ekstraktor	Klasifikator	Dataset	PCA	Acc. ± (std.)
openl3	kNN	cutoff_2s	128	0.45 ± (0.06)
openl3	kNN	cutoff_2s	0	0.44 ± (0.05)
openl3	kNN	cutoff_2s	64	0.43 ± (0.05)
openl3	SVM	cutoff_2s	64	0.42 ± (0.03)
openl3	SVM	cutoff_2s	128	0.42 ± (0.05)
openl3	perceptron	cutoff_2s	128	0.41 ± (0.04)
openl3	linearSVM	cutoff_2s	128	0.4 ± (0.03)
openl3	MLP	cutoff_2s	128	0.4 ± (0.04)
openl3	MLP	cutoff_2s	64	0.39 ± (0.03)
openl3	perceptron	cutoff_2s	0	0.39 ± (0.03)
openl3	linearSVM	cutoff_2s	64	0.38 ± (0.04)
openl3	perceptron	cutoff_2s	64	0.38 ± (0.03)
openl3	SVM	cutoff_2s	0	0.37 ± (0.03)
openl3	linearSVM	cutoff_2s	0	0.35 ± (0.06)
openl3	MLP	cutoff_2s	0	0.26 ± (0.22)

Slika 4.2. – *Vrijednosti kros-validacije korištenjem OpenL3 ekstraktora obilježja*

Sa slika iznad jasno se vidi da se najveća preciznost kros-validacije dobija za kNN *(k-Nearest Neighbors)* i SVM *(Support Vector Machines)* klasifikatore te se izbor klasifikatora svodi na jedan od ta dva, dok su ostali klasifikatori davali dosta lošije rezultate (preko 10% lošiji) pri čemu je MLP *(Multi-layerd perceptron)* davao najlošije rezultate.

U poglavlju 3 su navedene vrste modifikacija rađenih nad *dataset*-om za dobijanje većih performansi. Cilj ovih modifikacija je da se utvrdi optimalan oblik riječi za kojeg će ekstraktori

[2] Vrijednost 0 u koloni PCA označava da PCA nije iskorišten

obilježja i klasifikator dati najveću tačnost prepoznavanja izgovorenih riječi. Na slikama 4.3 i 4.4 se jasno može vidjeti uticaj različitih oblika riječi na tačnost prepoznavanja, gdje se potvrđuje teza iz poglavlja 3 da će se najlošiji rezultati dobijati za osnovni *dataset*, a bolji za podatke gdje se izgovor ponavlja neprekidno.

Najboljih 10 vrijednosti:

Ekstraktor	Klasifikator	Dataset	PCA	Acc. ± (std.)
openl3	SVM	repeat_4s	128	0.95 ± (0.02)
openl3	SVM	repeat_4s	64	0.94 ± (0.02)
openl3	kNN	repeat_4s	128	0.92 ± (0.02)
openl3	SVM	filtered_repeat_1s	64	0.92 ± (0.04)
openl3	SVM	repeat_2s	128	0.92 ± (0.02)
openl3	SVM	repeat_1s	128	0.92 ± (0.03)
openl3	SVM	filtered_repeat_2s	128	0.92 ± (0.03)
openl3	SVM	filtered_repeat_1s	128	0.92 ± (0.03)
openl3	SVM	repeat_1s	64	0.92 ± (0.04)
openl3	SVM	filtered_repeat_2s	64	0.91 ± (0.03)

Najlošijih 10 vrijednosti:

Ekstraktor	Klasifikator	Dataset	PCA	Acc. ± (std.)
openl3	SVM	filtered_2s	0	0.36 ± (0.03)
openl3	SVM	cutoff_2s	0	0.37 ± (0.03)
openl3	SVM	cutoff_2s	128	0.42 ± (0.05)
openl3	SVM	cutoff_2s	64	0.42 ± (0.03)
openl3	kNN	cutoff_2s	64	0.43 ± (0.05)
openl3	kNN	cutoff_2s	0	0.44 ± (0.05)
openl3	kNN	cutoff_2s	128	0.45 ± (0.06)
openl3	SVM	filtered_2s	64	0.46 ± (0.04)
openl3	SVM	filtered_2s	128	0.49 ± (0.03)
openl3	kNN	filtered_2s	0	0.54 ± (0.04)

Slika 4.3 – *Rezultati dobijeni OpenL3 mrežom*

Najboljih 10 vrijednosti:

Ekstraktor	Klasifikator	Dataset	PCA	Acc. ± (std.)
vggish	SVM	repeat_4s	64	0.66 ± (0.04)
vggish	SVM	repeat_4s	0	0.63 ± (0.06)
vggish	kNN	repeat_4s	64	0.62 ± (0.07)
vggish	SVM	repeat_4s	128	0.58 ± (0.04)
vggish	kNN	cutoff_1s	64	0.58 ± (0.07)
vggish	kNN	repeat_4s	0	0.57 ± (0.04)
vggish	SVM	repeat_2s	64	0.57 ± (0.04)
vggish	SVM	repeat_1s	64	0.56 ± (0.08)
vggish	kNN	repeat_1s	64	0.54 ± (0.09)
vggish	SVM	repeat_2s	0	0.54 ± (0.06)

Najlošijih 10 vrijednosti:

Ekstraktor	Klasifikator	Dataset	PCA	Acc. ± (std.)
vggish	SVM	filtered_2s	0	0.24 ± (0.05)
vggish	SVM	filtered_2s	128	0.28 ± (0.04)
vggish	kNN	filtered_repeat_2s	128	0.28 ± (0.04)
vggish	SVM	filtered_2s	64	0.3 ± (0.05)
vggish	kNN	filtered_2s	128	0.3 ± (0.05)
vggish	kNN	filtered_2s	0	0.31 ± (0.05)
vggish	kNN	filtered_repeat_1s	0	0.34 ± (0.06)
vggish	kNN	filtered_repeat_1s	128	0.34 ± (0.07)
vggish	kNN	cutoff_2s	128	0.35 ± (0.05)
vggish	kNN	filtered_repeat_2s	0	0.35 ± (0.06)

Slika 4.4 – *Rezultati dobijeni VGGish mrežom*

Osim što je tačnost ubjedljivo najveća za *repeat_4s dataset*, uviđa se pravilnost da se bolji rezultati postižu za riječi koje se više puta ponavljaju, odnosno za podatke koji vremenski duže traju. Takođe može se uočiti da filtriranje podataka niskopropusnim filtrom nema pretjeran uticaj na tačnost podataka, te se u daljnim razmatranjima ono odbacuje jer je redundantno i može usporiti rad aplikacije. Najjasniji zaključak je po pitanju izbora ekstraktora obilježja, gdje se jasno vidi čak 30% bolji rezultat koji se dobija korištenjem *OpenL3* mreže u odnosu na *VGGish* mrežu, te se ona nameće kao logičan izbor ekstraktora. Kao klasifikatora se nameće SVM, jer prema slici 4.3 jasno možemo vidjeti da od najboljih deset vrijednosti, čak devet koriste SVM.

4.2. Test mreže

U pethodnom odjeljku, kao glavni pokazatelj performansi je bila tačnost dobijena kros-validacijom istreniranog modela nad svim podacima iz *dataset*-a. To je dobar okvirni pokazatelj performansi i na osnovu njega je odabran ekstraktor obilježja, klasifikator i način obrade riječi, ali se bez testiranja mreže nad testnim podacima ne smije uzimati kao apsolutni pokazatelj tačnosti. Razlog zbog kojeg je uvijek potrebno imati dio testnih podataka se može vidjeti na slici 4.5, gdje je 10% ukupnog *dataset*-a iskorišteno kao testni podaci, dok je ostatak iskorišen za treniranje a osim kros-validacione tačnosti prikazana je i evaluacija sa test podacima (podacima koji nisu korišteni za trening mreže).

Ekstraktor	Klasifikator	Kernel	Dataset	PCA	Validacioni rez:	Test rez:
openl3	SVM	poly	representable_03	128	0.86 ± (0.03)	0.09
openl3	SVM	linear	representable_03	128	0.89 ± (0.04)	0.17
openl3	SVM	rbf	representable_03	128	0.94 ± (0.05)	0.19
openl3	SVM	poly	representable_03	0	0.9 ± (0.03)	0.92
openl3	SVM	linear	representable_03	0	0.89 ± (0.03)	0.91
openl3	SVM	rbf	representable_03	0	0.88 ± (0.05)	0.94
vggish	SVM	poly	representable_03	128	0.47 ± (0.02)	0.48
vggish	SVM	linear	representable_03	128	0.61 ± (0.03)	0.6
vggish	SVM	rbf	representable_03	128	0.75 ± (0.04)	0.76
vggish	SVM	poly	representable_03	64	0.33 ± (0.06)	0.34
vggish	SVM	linear	representable_03	64	0.52 ± (0.03)	0.52
vggish	SVM	rbf	representable_03	64	0.64 ± (0.05)	0.68
vggish	SVM	poly	representable_03	0	0.75 ± (0.03)	0.75
vggish	SVM	linear	representable_03	0	0.64 ± (0.03)	0.64
vggish	SVM	rbf	representable_03	0	0.7 ± (0.05)	0.67

Slika 4.5 – *Rezultati testiranja mreže*

Sa prethodne slike može se vidjeti da u slučaju *OpenL3* mreže dolazi do ogromnog pada u tačnosti kada je PCA uključen, dok u slučaju *VGGish* mreže tačnost počinje da opada tek sa PCA 64 veličinom. Osim neočekivanih rezultata po pitanju testiranja mreže, može se vidjeti i uticaj promjene kernela SVM-a na tačnost rezultata. Jasno se vidi da *poly* kernel daje najbolje rezultate kada se PCA ne koristi, dok je *rbf* kernel najbolji kada koristimo PCA. Takođe uočljiv je i porast u tačnosti kod *VGGish* mreže, uzrok toga je mala promjena u *dataset*-u, gdje se za razliku od *repeat_4s dataset*-a koristi drugačiji način razdvajanja riječi od tišine i šuma. Za generisanje novog *dataset*-a je iskrištena funkcija detaljnije opisana u poglavlju 5 koja koristi adaptivni prag za izdvajanje riječi iz zvučnog zapisa, a zatim je ponavlja u trajanju od četiri sekunde.

Dosadašnji rezultati su dobijeni klasifikacijom samo jednog klasifikatora, postavlja se pitanje da li će ovaj sistem bolje raditi ukoliko umjesto jednog klasifikatora koristimo dva ili tri. Na slici 4.6 su prikazani rezultati gdje je implementiran ovakav sistem odlučivanja i može se vidjeti napredak u slučaju korištenja dva klasifikatora, dok korištenje sva tri negativno utiče na postignutu tačnost.

Ekstraktor	Klasifikator	Kernel	Dataset	PCA	Test rez:
openl3	SVM	poly + rbf + linear	representable_03	0	0.9333
openl3	SVM	poly + rbf	representable_03	0	0.9524
openl3	SVM	rbf + linear	representable_03	0	0.9524
openl3	SVM	poly + linear	representable_03	0	0.9238

Slika 4.6 – *Tačnost predviđanja riječi korištenjem više klasifikatora*

5. Aplikacija za prepoznavanje govora

Kao što je u uvodnom dijelu rečeno, osnovni motiv realizacije ovog sistema jeste glasovno upravljanje pametnom kućom. Aplikacija je izrađena kao konzolna aplikacija koja ima tri opcije, glasovni unos komande nakon čega se u konzoli ispišu prepoznate riječi, glasovna reprodukcija prethodno izgovorene komande i izlazak iz aplikacije. Da bi se omogućila potpuna sinteza ovog sistema potrebno je implementirati tri cjeline:

1. Izdvajanje pojedinih riječi iz kontinualnog govora
2. Izdvajanje obilježja dobijenih riječi i njihova klasifikacija
3. Reprodukcija dekodovane komande

5.1. Izdvajanje riječi iz govora

Da bismo iskoristili sistem koji je detaljno objašnjen u prethodnim poglavljima, prvo je potrebno omogućiti snimanje govora i izdvajanje riječi iz snimljenog govora. Za snimanje govora iskorištena je biblioteka *sounddevice* i njena funkcija *record()*, ova funkcija omogućava da se uz pomoć mikrofona snimi audio zapis specifikovanog trajanja sa željenom frekvencijom odmjeravanja koji se kasnije može sačuvati kao *.wav* fajl ili direktno koristiti očitane vrijednosti kao niz. Preporuka je da se prvo sačuvaju vrijednosti, pa fajl ponovo učita, jer se tako osigurava predviđeno ponašanje (ovo utiče na performanse sistema, ali prema mjerenjima izvršenim na računaru vrijeme koje je potrebno da se fajl sačuva, a zatim ponovo učita u memoriju iznosi oko 1ms).

Nakon što se pravilno učita fajl, on se dijeli na prozore trajanja 60 ms, gdje se svaki odmjerak poredi sa vrijednosti praga odlučivanja. Prag odlučivanja u ovom slučaju izosi 5% apsolutne vrijednosti maksimalne amplitude učitanog fajla. Nakon što se sve vrijednosti uporede, ukoliko je više od 40% vrijednosti iznad praga odlučivanja ovaj prozor se kandiduje kao dio riječi. Ukoliko postoje više od 2 ovakva prozora, taj dio fajla predstavlja riječ i na njegu se dodaju još uzastopnih prozora koji ispunjavaju zadati uslov. Ovakvom realizacijom se omogućava prilično uspješno izdvajanje riječi iz govora, koje stvara ograničenja da ne postoje prevelike pauze tokom izgovora jedne riječi (duže od 120 ms) i da govornik naglasi svaku riječ sa pauzom od bar 120 ms.

Vrijeme koje je potrebno da se riječi izdvoje iz govora trajanja 4 s je između 10 ms i 20 ms na desktop računaru, oko 30 ms na *Jetson Nano Developer Kit*-u.

5.2. Izdvajanje obilježja riječi i klasifikacija

Ova druga tačka predstavlja okosnicu čitavog rada i detljano je objašnjena kroz prethodna dva poglavlja. Za ekstrakciju obilježja iskorištena je *OpenL3* pretrenirana mreža nad skupom podataka od 1050 riječi koje su prethodno izdvojene kao što je objašnjeno u tački 5.1 a zatim se izdvojeni sadržaj ponavlja onoliko puta koliko je potrebno da popuni 4 s trajanja (88200 odmjeraka pri frekvenciji odmjeravanja od 22050 Hz).

Nakon izdvajanja obilježja, vrši se predviđanje sa dva SVM klasifikatora, jedan koji koristi *rbf* kernel i drugi koji koristi *poly* kernel. Pošto je *OpenL3* mreža trenirana sa ulazima trajanja 1 s, imaćemo jedan set obilježja za svaku sekundu, odnosno predikciju za svaki sekund trajanja audio signala. Ovim je omogućeno bolje predviđanje riječi, jer se sada mogu spojiti dvije predikcije od dva različita klasifikatora (koje će u ovom slučaju imati ukupno 8 vrijednosti) i izabrati onu vrijednost koja se najviše ponavlja.

5.3. Reprodukcija dekodovane komande

Kao završni čin predstavlja dostavljanje rezultata korisniku, u ovom slučaju to je ispisivanje prepoznatih riječi na standardni izlaz. Osim tekstualnog izlaza u ovoj aplikaciji je implementirana i opcija reprodukcije izdvojenih riječi. Ovaj vid reprodukcije je naknadno dodan jer može korisniku da ukaže u čemu griješi tokom svog izgovora (da li je previše brzo izgovorio riječi pa ih aplikacija nije uspjela razdvojiti, da li je sporo izgovorio jednu riječ pa ju je aplikacija razdvojila na dva dijela itd.).

5.4. Portovanje aplikacije za Jetson Nano razvojno okruženje

Aplikacija kao i čitav sistem je razvijana na desktop vreziji *Ubuntu 18.04* operativnog sistema i prelazak na *Jetson Nano* razvojno okruženje je trebao biti veoma jednostavan jer je osnova njegovog *NVIDIA L4T* operativnog sistema takođe *Ubuntu 18.04* međutim taj prelaz nije bio ni najmanje jednostavan.

Jetson Nano koristi *JetPack SDK* [8] uz pomoć kojeg je omogućeno razvijanje *AI* aplikacija i podržano korištenje *Tensorflow* biblioteka, *CUDA* jezgara i ostalih prednosti koje donosi ovakav uređaj. Instalacija najnovijih biblioteka i paketa je veoma dobro dokumentovana i na *nVidia*-inoj stranici [9] posvećenoj za instaliranje *Tensoflow* biblioteke i u tom slučaju ne postoje nikakve teškoće i sam proces se završava kroz poziv par komandi. Problemi nastaju kada je potrebno instalirati starije verzije paketa i biblioteka jer određena verzija *JetPack SDK*-ja omogućava samo instalaciju „novijih“ paketa, te ne postoji kompatibilnost unazad, tako da za *Tensorflow 1.13.1* koji je potreban za rad *sklearn-audio-transfer* bibioteke, potreban je *JetPack v4.2* dok su na sajtu postavljene dvije najnovije verzije *JetPack v4.4* i *JetPack v4.3*. Nakon što se uspješno instalira kompatibilna *Tensorflow* biblioteka (tokom raznih pokušaja, aplikacija je pokrenuta i sa TF 1.15.2 verzijom) omogućeno je pokretanje aplikacije. Aplikacija neće raditi ukoliko se ne uključi GPU podrška (poseban dio koda u aplikaciji služi tome) i nakon toga se može koristiti. Njeno korištenje nije ni najmanje fluidno i za razliku od desktop verzije gdje od izgovaranja komande do njenog realizovanja prođe 10-15 sekundi u ovom slučaju je potrebno do 50 sekundi uz neprekidan prikaz upozorenja o korištenju zastarjele verzije *Tensorflow* biblioteke.

6. Zaključak

Biblioteka *sklearn-audio-transfer-learning* predstavlja jednostavno okruženje za izgradnju sistema za klasfikaciju audio fajlova, koje se može veoma lako i efikasno modifikovati da podržava klasifikaciju ne samo muzičkih nego i ostalih audio fajlova kao što je prikazano u izradi ovog rada.

Tokom izrade ovog projekta, vremenski najzahtjevniji dio je bio izrada samog *dataset*-a, odnosno prikupljanje riječi, njihovo isjecanje i podjela na govornike i izgovorene riječi. Nakon toga proces razvoja aplikacije je bio postepen i može se podijeliti na tri glavne cjeline.

Prva cjelina obuhvata upoznavanje sa *sklearn-audio-transfer-learning* bibliotekom, njihovim oznakama i nazivima funkcija (koje nisu intuitivne), te ispravljanjem dijela koda jer je *OpenL3* biblioteka je u međuvremenu redizajnirana, tako da funkcije pozivane unutar glavne skripte *audio_transfer_learning.py* kao takve više ne postoje. Takođe jedna veoma bitna stavka kod *OpenL3* biblioteke je njen *memory leak* koji pri više manjih *batch*-eva za izvlačenje obilježja dovodi do prekida rada aplikacije usljed punjenja *swap* particije *Linux OS* i RAM-a.

U drugu cjelinu se može uvrstiti dio oko manipulacije *dataset*–om, odnosno transformacija riječi i odabir klasifikatora. Tokom ovog dijela, najveći napredak u tačnosti sistema (od nekih 30%) je dobijen zahvaljujući poznavanju obrade signala i programskog jezika *python* koji omogućuje veoma efikasnu manipulaciju raznim podacima.

Treća i posljednja cjelina je generisanje korisničke aplikacije. Ova aplikacija je zamišljena kao konzolna aplikacija prvenstveno za testiranje prepoznavanja govora sa mogućnošću unapređenja kroz naredne iteracije. Najveće ograničenje u ovom trenutku predstavlja kompatibilnost sa starijim bibliotekama gdje je najbolje rješenje pravljenje nove *enviroment* varijable u *python*-u i instaliranje svih potrebnih biblioteka ispočetka.

Tačnost prepoznavanja govora i rada aplikacije je zadovoljavajuća i kreće se 80-90% za govornike koji se nalaze i u *dataset*-u, dok je za govornike čiji glas nije ranije korišten iznosi oko 60-70%. Takođe, primjetna je zavisnost od mikrofona, gdje korištenjem boljeg *Blue Snowball* mikrofona se dobija tačnost blizu 90% u odnosu na ugrađeni mikrofon gdje je tačnost oko 80%. Najčešće pogrešne predikcije su između riječi „uključi“ i „isključi“ jer je razlika između ove dvije riječi samo u prvom slovu i ukoliko je ono izgovoreno nešto tiše, postoji mogućnost da ga aplikacija odsječe. Kao jedno od mogućih unapređenja ove aplikacije može se postaviti sistem odlučivanja predviđene riječi gdje se osim informacije kojih predikcija ima najviše (objašnjeno u poglavlju 5.2) se uvodi težinski faktor koji favorizuje određene konstrukcije (npr. komanda „uključi prozor“ ima manje smisla od komande „uključi svjetlo“).

7. Literatura

[1] J Pons, „sklearn-audio-transfer-learning“ *https://github.com/jordipons/sklearn-audio-transfer-learning*, posjećeno 4.7.2020.

[2] „VGGish model“ https://github.com/tensorflow/models/tree/master/research/audioset/vggish, posjećeno 4.7.2020.

[3] „OpenL3 documentation“ https://openl3.readthedocs.io/en/latest/, posjećeno 4.7.2020.

[4] J Pons, „musiccn github page “ *https://github.com/jordipons/musicnn*, posjećeno 4.7.2020.

[5] Milomir Babić, Prepoznavanje izolovanih riječi korištenjem sklearn-audio-transfer-learning, ETF Banja Luka, 2020.

[6] „AudioSet, sound vocabulary and dataset“ *https://research.google.com/audioset/* posjećeno 4.7.2020.

[7] R. Arandjelović and A. Zisserman, “Look, listen and learn.,” in *ICCV*, pp. 609–617, IEEE Computer Society, 2017.

[8] „nVidia JetPack SDK“ https://developer.nvidia.com/embedded/jetpack, posjećeno 5.7.2020.

[9] „TensorFlow on Jetson Platform“https://docs.nvidia.com/deeplearning/frameworks/install-tf-jetson-platform/index.html , posjećeno 5.7.2020.