Prepoznavanje glasa algoritmima za obradu signala

Stipinović, Karlo

SVEUČILIŠTE U SPLITU

POMORSKI FAKULTET

KARLO STIPINOVIĆ

PREPOZNAVANJE GLASA ALGORITMIMA ZA OBRADU SIGNALA

DIPLOMSKI RAD

SPLIT, 2019.

SVEUČILIŠTE U SPLITU

POMORSKI FAKULTET

STUDIJ: POMORSKE ELEKTROTEHNIČKE I INFORMATIČKE TEHNOLOGIJE

PREPOZNAVANJE GLASA ALGORITMIMA ZA OBRADU SIGNALA

DIPLOMSKI RAD

MENTOR:

Doc.dr.sc Joško Šoda

KOMENTOR: **STUDENT:**

Mag. ing. el. Ivan Pavić **Karlo Stipinović (MB: 0171262473)**

SPLIT, 2019.

	POMORSKI FAKULTET U SPLITU	Stranica: Šifra:	1/1 F05.1.-DZ
	DIPLOMSKI ZADATAK	Datum:	22.10.2013.

Split, 26. rujna 2019.

Zavod/studij: Pomorske elektrotehničke i informacijske tehnologije

Predmet: _Napredna poglavlja iz obrade i analize signala

D I P L O M S K I ZADATAK

Student/ca: _____KARLO STIPINOVIĆ

Matični broj: _____0171262473

Zavod/studij: ____Pomorske elektrotehničke i informacijske tehnologije

ZADATAK: PREPOZNAVANJE GLASA ALGORITMIMA ZA OBRADU SIGNALA

OPIS ZADATKA: KORISTEĆI NEURONSKU MREŽU I ALGORITME UČENJA PREDSTAVITI NAPREDNE ALGORITME ZA OBRADU SIGNALA. NA PRIMJERU RIJEČI: "YES", "NO" I "MARVIN" PREDLOŽITI POGODNI KLASIFIKACIJSKI ALGORITAM STROJNOG UČENJA. KORISTEĆI STATISTIČKU OBRADU USPOREDITI DOBIVENE REZULTATE.

CILJ: POTREBNO JE UZ UPORABU RAZVOJNOG ALATA PYTHON PREDLOŽITI FUNKCIONALNO RJEŠNJE ZA PREPOZNAVANJE GLASA, KORISTEĆI NEURONSKU MREŽU KOJA SE BAZIRA NA LSTM ALGORITMU UČENJA.

ZADATAK URUČEN STUDENTU/CI: _VELJAČA 2019._____

POTPIS STUDENTA/CE: _____

MENTOR: _doc. dr. sc. Joško Šoda_____

SAŽETAK

U radu se opisuje izrada programa za klasifikaciju izgovorenih riječi *„Marvin“*, *„Yes“* i *„No“* pomoću neuronske mreže sa dugoročnim pamćenjem (*engl. Long Short Term Memory - LSTM*) u programskom jeziku *Python*. Kao uvod u obradu i analizu audio signala opisuju se pojmovi uzorkovanja, Fourierovog reda i transformacije te diskretne Fourierove transformacije. Dobivanje korisnih informacija uz istovremeno smanjenje količine podataka ostvareno je pomoću sljedećih svojstava zvučnih signala: stopa prelaska nule, kepstralni koeficijenti po Mel-skali, spektralni centroid i spektralno opadanje. Za detaljni pregled preciznosti klasifikacijskog modela proučavalo se njegove funkcije greške i preciznosti, te klasifikacijsko izvješće i konfuzijsku matricu.

Ključne riječi: LSTM neuronska mreža, Python, Fourierova transformacija, svojstva, preciznost

ABSTRACT

This thesis describes development of a program for classification between spoken words *„Marvin“*, *„Yes“* and *„No“* using *Long Short Term Memory (LSTM)* neural network in Python programming language. As an introduction to the area of audio signal analysis and processing, description of following terms is provided: Fourier series, Fourier transform and discrete Fourier transform. Features that are used as a measure of acquiring useful information and data reduction of audio signal are the following ones: zero crossing rate, Mel-Frequency Cepstral Coefficients, spectral centroid and spectral rolloff. Precision and loss functions with classification report and confusion matrix are shown as a measure of detailed overview of classification model accuracy.

Key words: LSTM neural network, Python, Fourier transform, features, accuracy

SADRŽAJ

1. UVOD

Automatsko prepoznavanje i razumijevanje izgovorenog jezika je jedan od najznačajnijih koraka prema prirodnom sučelju čovjek-stroj. Istraživanja u ovom polju su zadnjih nekoliko desetljeća proizvela zapanjujuće rezultate, vodeći i do mnoštva novih uzbudljivih primjena u svakodnevnom životu i specijalnim područjima. Mnoge statističke metode su razvijene da bi omogućile strojevima prepoznavanje i razumijevanje govora direktno iz podataka. S time se pomoću brojnih istraživanja u polju procesuiranja govora omogućilo mnoštvo uspješnih primjena u telefonskim sustavima, sustavima televizijskog emitiranja, sustavima koji koriste osobna računala i mnogim drugima [11].

Govor je primarni i najuvjerljiviji način komunikacije među ljudima. Istraživanja u polju strojnog prepoznavanja govora su iz razloga automatizacije rada strojeva i tehnološke znatiželje za izgradnju strojeva koji oponašaju ljude, privukla veliku količinu interesa i entuzijazma. Iako je tehnologija daleko od mogućnosti uspješne međusobne komunikacije čovjeka i stroja poput komunikacije između ljudi, istraživanja tijekom posljednjih godina pokazuju značajan napredak [11].

Veliki interes i napredak u ovom polju tijekom zadnjih nekoliko desetljeća pokazuju primjeri istraživanja koja sežu od 1950-ih kada su se pokušavali izgraditi sustavi za prepoznavanje izoliranih izgovorenih znamenki brojeva za jednog govornika 1952., od strane tvrtke „Bell Laboratories“ [12], prepoznavanja slogova jednog govornika od strane trvtke „RCA Laboratories“ [13], te izgradnja sustava za prepoznavanje desetak samoglasnika od strane Massachusetts Institute of Technology (MIT), „Lincoln Laboratory-a“ [14]. Zatim su se 60-ih u Sovjetskoj uniji razvile implikacije za razvoj automatskog prepoznavanja govora u vidu dinamičkog programiranja za poravnavanje u vremenu između dva primjera, odnosno izgovora, da bi se dobio odgovarajući rezultat. Od tada je dinamičko programiranje postalo neophodna tehnika za pronalaženje podudarajućih uzoraka. U 1970-ima su s brojnim istraživanjima najviše doprinijele trvtke „International Business Machines Corporation“ (IBM) i „Bell Laboratories“, primjerice sa glasovnim aktiviranjem pisaće mašine, glasovnim biranjem, pokušajima automatizacije telefonskih poziva, dizajnom sustava prepoznavanja glasa neovisnom o govorniku itd. [11].

Brzi razvoj statističkih metoda 1980-ih, pogotovo razvoj metode Skrivenog Markovljevog Modela (HMM) je uvelike ubrzao napredak u području prepoznavanja govora [11]. Korištenje raznih statističkih metoda uz izdvajanje svojstava iz signala u frekvencijskom i vremenskom području je dominiralo u većini vrsta audio klasifikacija i prepoznavanja glasa ili govora, no zadnjih godina se pojavila još jedna jako značajna metoda, primjena strojnog učenja u vidu neuronskih mreža te višeslojne neuronske mreže zvane duboke neuronske mreže (DNN). Korištenje neuronskih mreža kao akustičkih modela za prepoznavanje govora baziranih na statističkim modelima, uglavnom na HMM-u, je uvedeno još prije više od 20 godina. Međutim, do prije nekoliko godina, neuronske mreže nisu bile standardni alat u sustavima automatskog prepoznavanja govora u stvarnom vremenu. Računalna ograničenja i količina dostupnih podataka za treniranje su uvelike ograničili mogući napredak. Tek odnedavno, povećanjem računalne moći i količine sakupljenih podataka, pristupi temeljeni na tzv. „dubokom učenju“, koje koristi višeslojne neuronske mreže, pokazali su poboljšane performanse u usporedbi sa konvencionalnim metodama strojnog učenja za primjene u raznim područjima. U području prepoznavanja govora, duboke neuronske mreže izgrađene sa sposobnostima pamćenja su dostigle rekordnu preciznost [15].

U poglavlju 2. O SIGNALIMA I AUDIO je definiran pojam signala te osnovne vrste i oblici signala sa ilustriranim primjerima. Također se pojašnjava pojam zvuka, te proces dobivanja audio signala u digitalnom obliku (uzorkovanje) i osnovni procesi obrade i analize signala poput Fourierovog reda i transformacije.

U poglavlju 3. IZDVAJANJE SVOJSTAVA I PREPOZNAVANJE GOVORA POMOĆU NEURONSKIH MREŽA se objašnjava proces izdvajanja svojstava iz izvornog audio signala kao ključan proces obrade i analize signala te mjere smanjenja količine podataka u polju prepoznavanja govora. Uz to se i objašnjava pojam povratnih neuronskih mreža, te se detaljno opisuje rad povratne neuronske mreže sa dugoročnom memorijom (*engl. Long Short Term Memory – LSTM*).

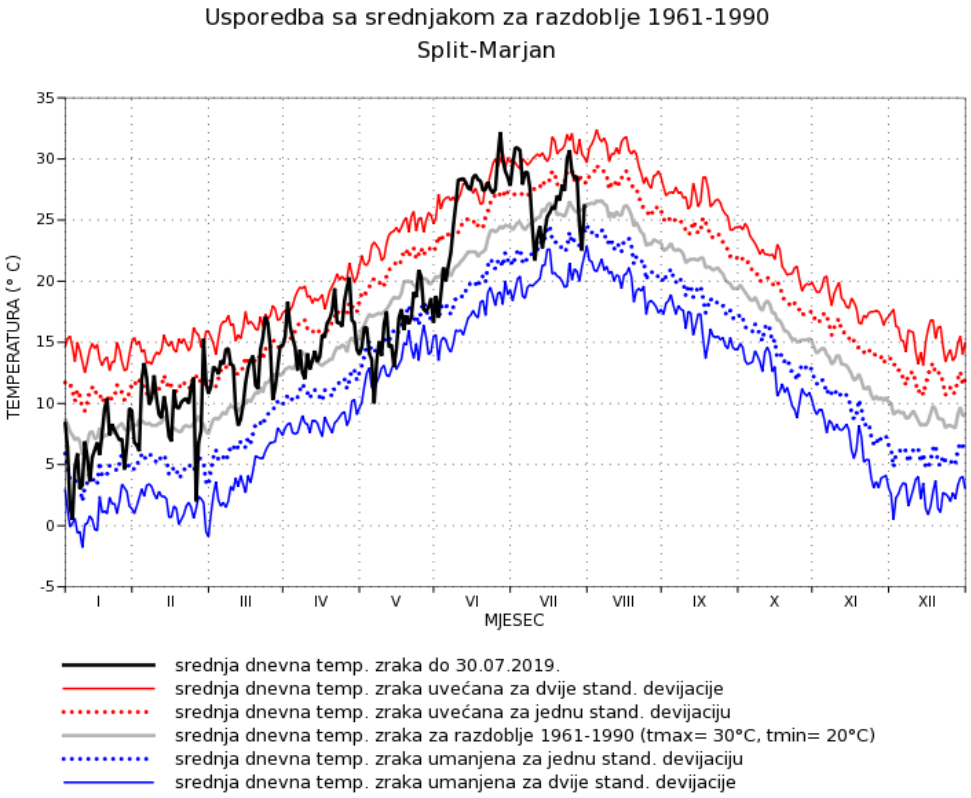
U poglavlju 4. PREPOZNAVANJE GOVORA U PROGRAMSKOM JEZIKU PYTHON detaljno je opisan proces i programski kod za kreiranje klasifikacijskog modela za razlikovanje između izgovorenih riječi „Marvin“, „Yes“ i „No“. U to spada i opis optimizacijskog algoritma te mjere vizualizacije i analize preciznosti za objektivno ocjenjivanje učinkovitosti modela i podataka.

2. O SIGNALIMA I AUDIO

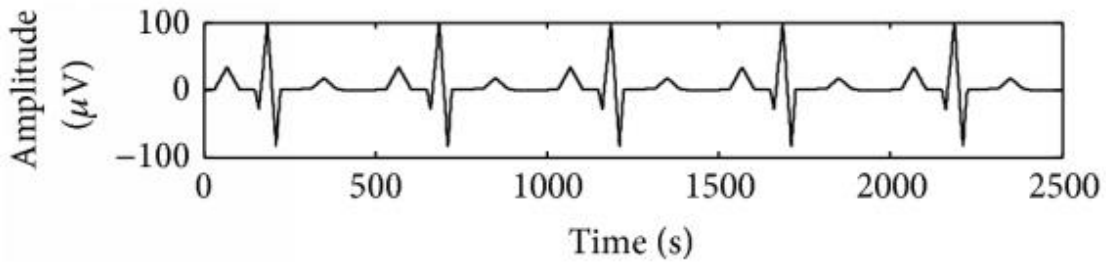
2.1. OPĆENITO

2.1.1. O signalima

Signal je generalno definiran kao skup znakova, simbola, fizičkih gestikulacija ili ostalih fizičkih veličina koje su podložne varijacijama u prostoru ili vremenu koje prenose informacije ili poruke. Signal može varirati vremenski i prostorno. Prisustvo signala je svakodnevno, pojavljivanjem u prirodnim sustavima i umjetnim, odnosno sustavima napravljenim od strane čovjeka. Prirodni signali su primjerice mjerenje temperature zraka tijekom određenog vremena, mjerenje otkucaja srca tijekom određenog vremena, mjerenje zvučnih vibracija itd[1]. Takvi se signali nazivaju analogni, i svrstavaju se u kontinuirane signale čija je glavna karakteristika kontinuirana promjena.[4]. Na slici 1 je prikazan kontinuirani signal odnosa između temperature i vremena u gradu Splitu, te na slici 2 kontinuirani signal otkucaja srca.

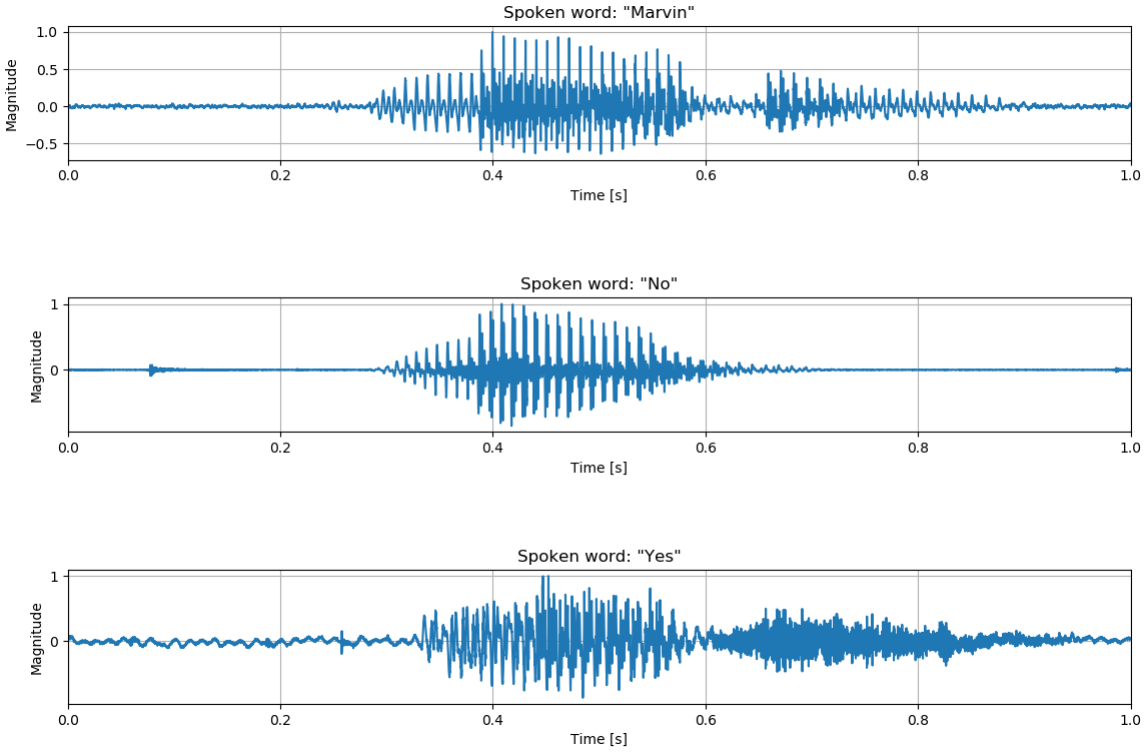


Slika 1. Godišnje temperature u Splitu [2]



Slika 2. Signal otkucaja srca [3]

Druga vrsta signala su digitalni signali, koji su konstruirani diskretizacijom valnih oblika fizikalne veličine, čime se signal prikazuje kao niz diskretnih vrijednosti. Ova vrsta signala je potrebna za prikaz signala i informacija, te manipuliranje i transformaciju istih na kompjuterima. Računala kontinuirane signale uzorkuju s određenim periodom uzorkovanja, kako bi što vjernije prikazali informaciju signala kao niz diskretnih vrijednosti. Na slici 3 prikazani su diskretizirani valni oblici zvučnog signala izgovorenih riječi „Marvin“, „Yes“ i „No“.



Slika 3. Zvučni signali izgovorenih riječi

Na slici 3 iako je prikazan diskretizirani spektar, izgleda kao kontinuirani, iz razloga što ima jako mali period uzorkovanja, odnosno veliku frekvenciju uzorkovanja od 16 kHz ili 16 000 uzoraka u sekundi.

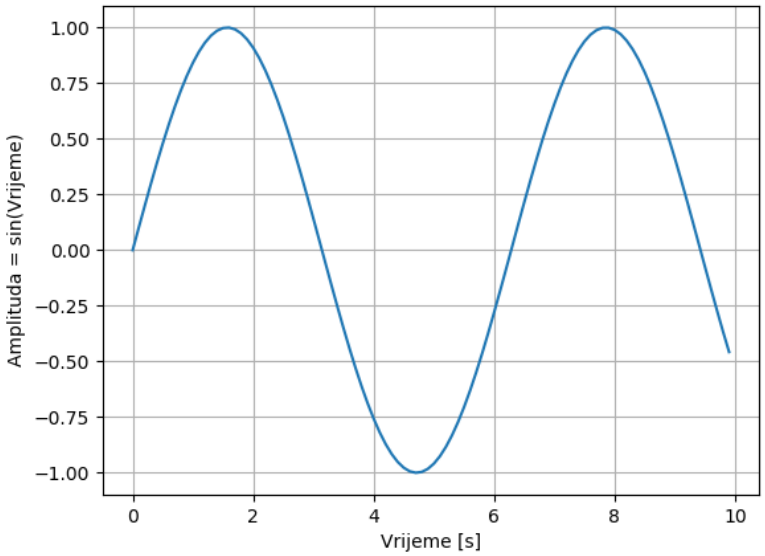
Kako bi se dobila korisna informacija iz signala, odnosno niza vrijednosti, signale se promatra u određenoj domeni. Najčešće je ta domena određeno vrijeme tijekom kojeg se bilježe podaci, odnosno stvara niz vrijednosti. Domena može biti i prostorna, koja je najčešće dvodimenzionalna ili trodimenzionalna u smislu da ima dvije ili tri neovisne varijable. U toj domeni se najčešće prikazuju slike, a u kombinaciji s vremenskom i video podaci. Još jedna domena, naročito potrebna u obradi zvučnih signala je frekvencijska domena, koja nastaje transformacijom signala u vremenskoj domeni.

2.1.2. Osnovne vrste i oblici signala

Postoje razne podjele signala, od kojih je jedna od važnijih periodičnost signala. Za signal *x(t)* se kaže da je periodičan, s periodom P, ako vrijedi jednadžba 1:

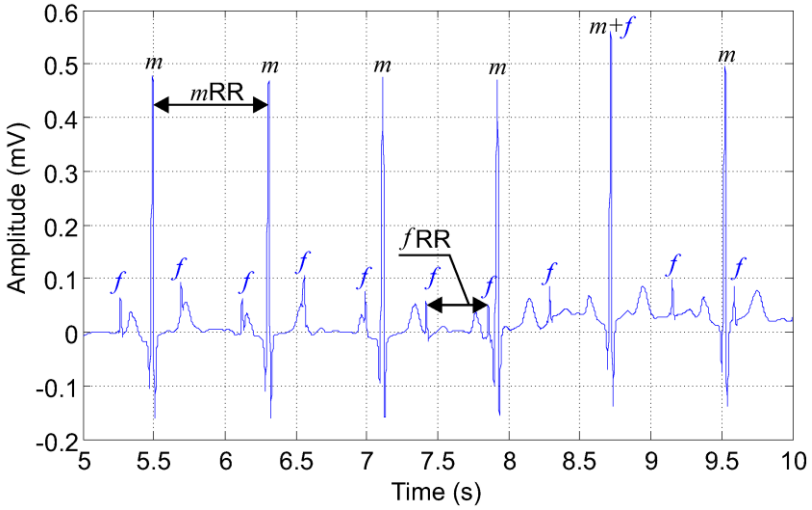
$$x(t) = x(t + P) \tag{1}$$

Period P se još naziva i osnovni period. Periodični signali sačinjavaju sumu periodičnih valnih oblika koji su harmonički povezani. Na slici 4 je prikazan primjer sinusoidalnog periodičkog signala.



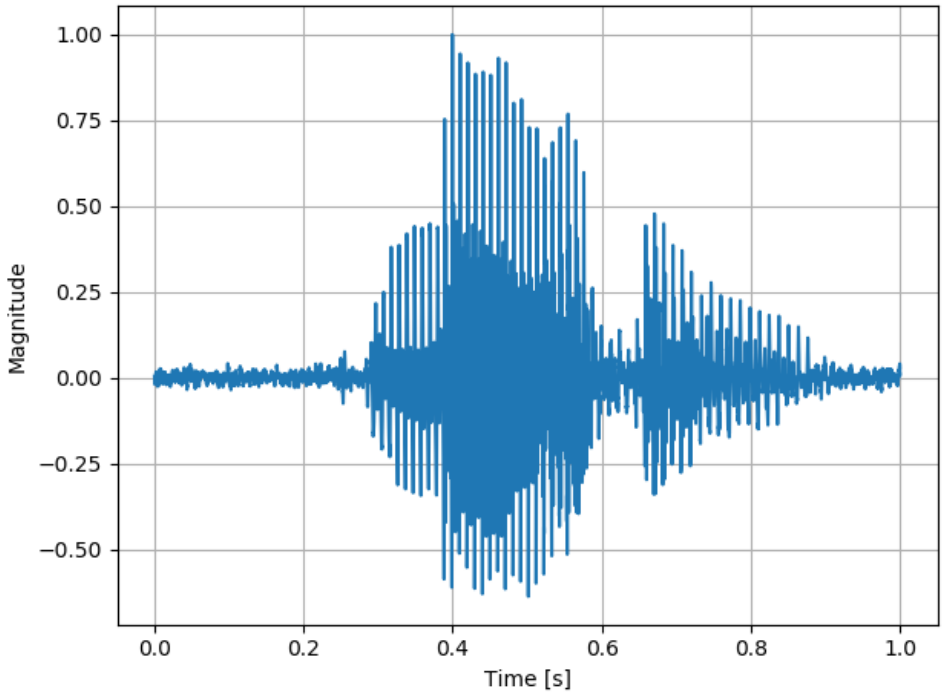
Slika 4. Sinusoidalni signal

Druga vrsta signala po periodičnosti su signali koji mogu biti sačinjeni kao suma periodičnih valnih oblika koji nisu harmonično povezani. Takvi se signali nazivaju kvazi-periodični. Svojstveni su po tome što vizualno prate određeni uzorak, no nemaju fiksni period u vremenu. Jedan primjer kvaziperiodičnih signala su otkucaji srca, prikazani na slici 5.



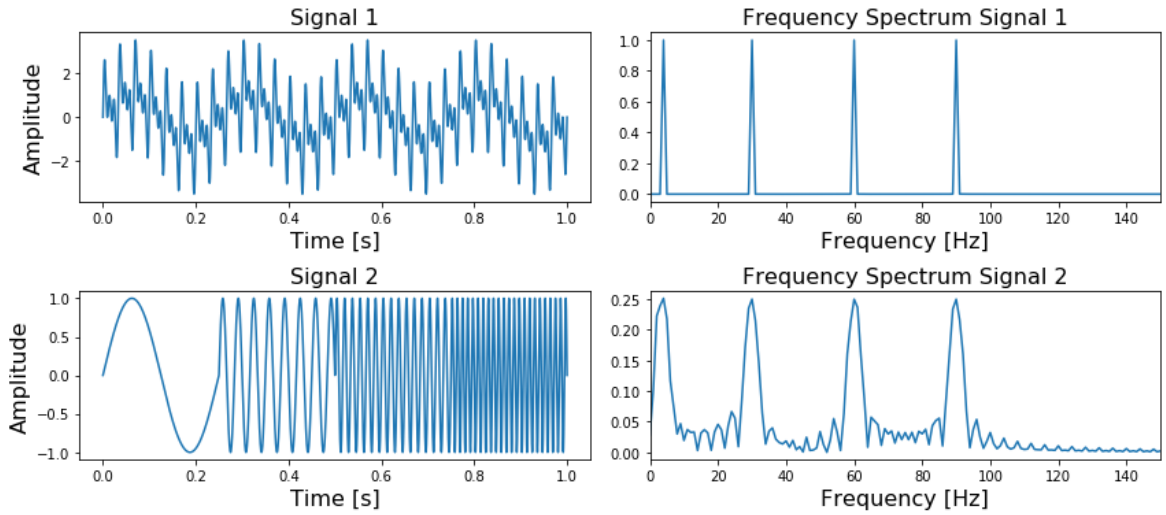
Slika 5. Abdominalni elektrokardiogram trudne žene (m) i fetusa (f) [20]

Većina signala zapravo nisu ni periodični ni kvazi-periodični nego aperiodični signali. Aperiodični signali se ne ponavljaju tijekom određenog vremenskog intervala i ne mogu se predstaviti matematičkom jednadžbom za razliku od periodičkih signala. Jedan od najučestalijih aperiodičkih signala je šum, u kojem slučaju ima prirodu slučajnog signala, a može biti također u obliku impulsa nejednolikog trajanja i ostalih signala koji nemaju točan period. Na slici 6 je prikazan primjer aperiodičkog signala govora.



Slika 6. Aperiodički signal govora

U slučaju šuma, odnosno slučajnog signala postoji također razlika kad se govori o promjeni njegovih karakteristika s vremenom. Ukoliko se primjerice izračunaju karakteristike poput prosjeka, minimuma i maksimuma tijekom kratkih isječaka signala u vremenu, i zaključi se da nema bitne promjene, signal se smatra stacionarnim. S druge strane ako se proračunom istih karakteristika opazi bitna promjena vrijednosti tijekom vremena, signal se smatra nestacionarnim[1]. Na slici 7 se može vidjeti razlika stacionarnog i nestacionarnog signala, gdje je „Signal 1“ stacionarni signal nastao zbrajanjem četiri sinusoidalna signala frekvencija 4, 30, 60 i 90 Hz-a, koje se frekvencije jasno mogu iščitati u frekvencijskom spektru istog signala. „Signal 2“ je signal dobiven pridruživanjem ista 4 sinusoidalna signala, na način da svaki od četiri signala čini po 250 milisekundi ukupnog trajanja signala, čime do očitovanja u frekvencijskom spektru dolaze dodatne frekvencijske komponente, tvoreći šum u frekvencijskom spektru [6].



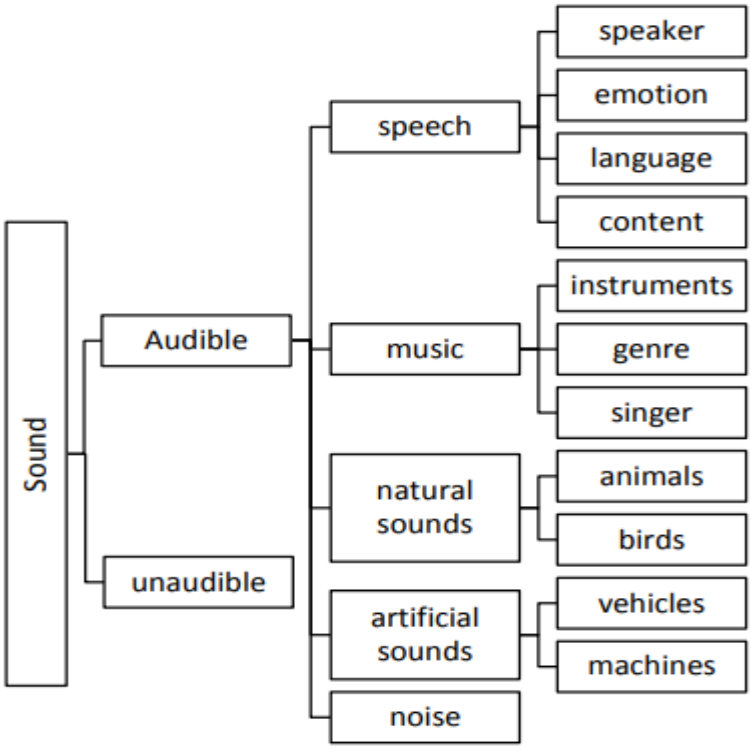
Slika 7. Stacionarni i nestacionarni signal u vremenskoj i frekvencijskoj domeni [5]

Kao dio generalne podjele signala prema strukturi valnog oblika, osim slučajnih postoje i determinirani signali kod kojih su vrijednosti signala predvidive i mogu se odrediti matematičkim izrazom. Primjeri takvih signala su najčešće signali definirani kroz matematičke funkcije.

2.2. AUDIO SIGNALI

2.2.1. Općenito

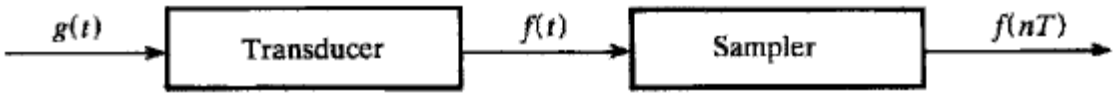
Audio signali ili zvuk za čovjeka nastaje od varijacija tlakova u zraku koji stvaraju vibracije na bubnjiću ljudskog uha, nakon kojeg se vibracije u unutarnjem uhu pretvaraju u živčane impulse koji se obrađuju u mozgu. Ljudski sustav sluha funkcionira u opsegu zvukova frekvencija od 20 Hz do 20 kHz. Raspon zvučnog intenziteta je otprilike od 0 do 120 dB (decibela) koji su zapravo raspon jačine zvuka od šuštanja lišća do zvuka uzlijetanja aviona. Ljudski slušni sustav je sposoban procesuirati kompleksne zvučne mješavine i time čovjeku omogućiti bogati niz informacija o okolišu s obzirom na lokaciju i karakteristike objekata koji proizvode zvuk. Zvučni signali se mogu podijeliti u kategorije zvukova prirode i okoliša, umjetnih zvukova, govora i glazbe. Govor se može promatrati kao niz fonema, a muzika kao razvojni uzorak nota, dok su prirodni ili okolišni zvukovi najčešće u obliku šuma [6]. Spomenute kategorije imaju svoje podkategorije prikazane na slici 8. Audio signali variraju s vremenom, te se mogu prikazati u vremenskoj ili frekvencijskoj domeni. Svi zvukovi su karakteristični po svojim obilježjima koji se mogu jasno očitovati promatranjem navedenih domena određenog audio signala.



Slika 8. Klasifikacija zvuka [9]

2.2.2. Uzorkovanje

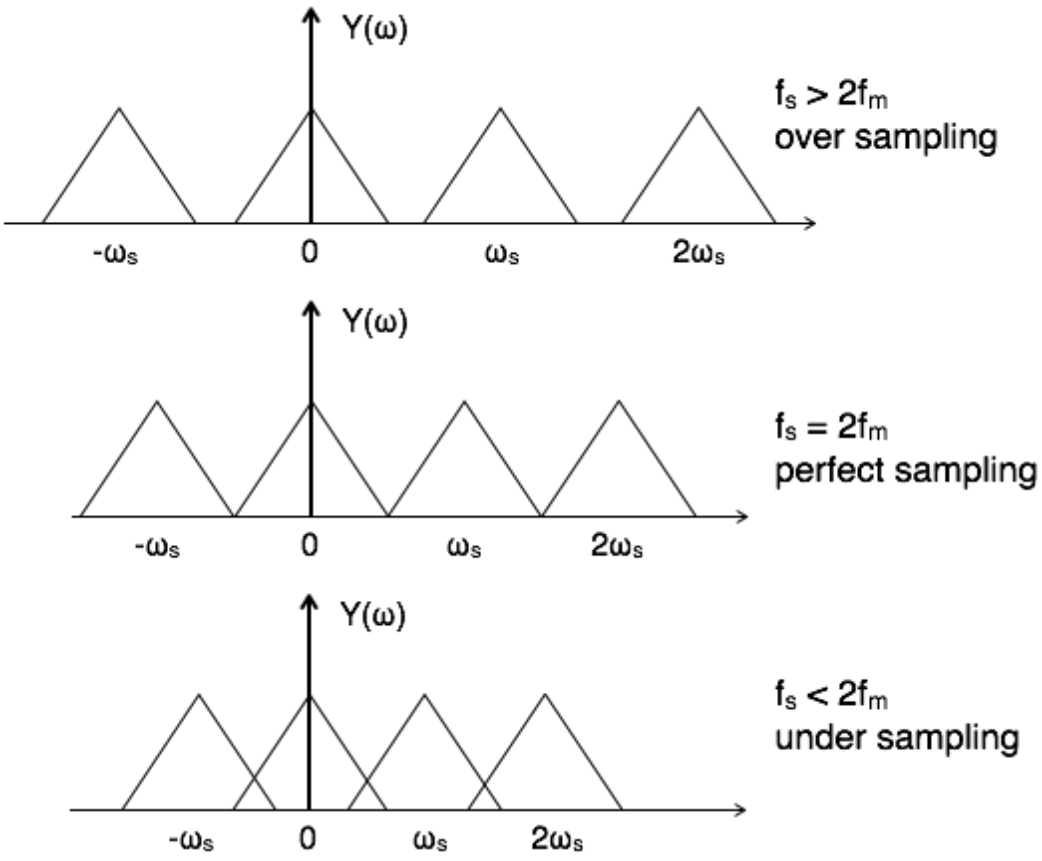
S obzirom da su većina audio signala kontinuirani u amplitudi i vremenu, a kao štoje već prije spomenuto, računala rade sa digitalnim podacima, da bi se obradio i analizirao signal na računalu, potrebno je pretvoriti signal iz analognog u digitalni. Na slici 9 je prikazan proces A/D pretvarača, koji pretvara analogni signal u digitalni. Analogni signal *g(t)* se mjeri senzorom, u ovome slučaju s mikrofonom, gdje varijacije u tlaku pomiču membranu koja siječe magnetsko polje te se kreira električni signal, dok se preko pretvornika pretvara u digitalni, točnije u sekvencu *f(nT)* s periodom uzorkivanja T.



Slika 9. Proces pretvorbe analognog signala u digitalni [1]

Sklop za uzorkovanje mjeri *f(t)* svaki period uzorkovanja *T* i pretvara ga u *diskretni vremenski niz, f(nT)*. Tipični A/D pretvarači imaju frekvenciju uzorkovanja od 0 do 100 000 uzoraka u sekundi. Telekomunikacijska tehnologija i radar koriste A/D pretvarače sa frekvencijama uzorkovanja do 100 MHz. Tom procesu je svojstvena i kvantizacija magnitude signala. Unutar hardvera A/D pretvarača, svaki uzorak magnitude analognog signala je pretvoren u jedinicu kompjuterske memorije koja varira od 4 do 32 bita. Za većinu primjena je dovoljna 12-bitna kvantizacija kako bi se dobila dovoljna preciznost za izbjegavanje kvantizacijske greške. Za primjene koje zahtijevaju ekstremnu preciznost ili analiziranje signala sa velikim opsegom jakosti, koriste se pretvarači sa većim memorijskim kapacitetom po jedinici memorije.

S obzirom da je uzorkovanje prijelaz kontinuiranog signala u diskretni, od velike važnosti je poznavati proces uzorkivanja kako nebi došlo do gubitka informacija. Odabir frekvencije uzorkivanja detaljno opisuje *Nyquist-Shannonov teorem uzorkovanja* koji kaže da se signal *x(t)* kontinuiran u vremenu može rekonstruirati točno sa uzorcima $x[n] = x(nT_s)$, ako se uzorci uzimaju po frekvenciji uzorkovanja $f_s = 1 / T_s$ koja je veća od dvostruke najveće frekvencije signala ($2f_{max}$) [8]. Uzorkovanje u frekvencijskom spektru je prikazano na slici 10, gdje je uočljivo da se neispunjavanjem uvjeta teorema uzorkovanja, za rezultat dobivaju nove frekvencijske komponente u uzorkovanom signalu, čime se značajno gube informacije od izvornog signala. Ta pojava se naziva preklapanje (eng. aliasing).



Slika 10. Uzorkovanje u frekvencijskom spektru [9]

2.2.3. Fourierov red i transformacija

Da bi se signal prikazao u frekvencijskom području, ključno je znanje Fourierovog reda i Fourierove transformacije. Francuski matematičar i fizičar po kojemu je ovaj način obrade dobio ime je Jean Baptiste Joseph Fourier, koji je u svrhu rješavanja diferencijalnih jednadžbi širenja topline uveo Fourierov red. Fourierov red je glavna ideja Fourierove analize, koja pokazuje da se svaka periodička funkcija može zapisati kao suma sinusa različitih amplituda, faza i frekvencija, te se prikazuje na sljedeći način:

$$x(t) = \frac{A_0}{2} + \sum_{n=1}^{\infty} (A_n \sin(n\omega_0 t) + B_n \cos(n\omega_0 t)) \tag{2}$$

gdje je:

x(t) – signal u vremenskoj domeni

$\frac{A_0}{2}$ – istosmjerna komponenta, služi za translaciju funkcije duž y-osi

A_n, B_n – Fourierovi koeficijenti

n – red harmonijske oscilacije

ω_0 – kutna frekvencija

Fourierovi koeficijenti u prethodnoj jednadžbi ovise o valnom obliku signala x(t), iz toga slijede izrazi:

$$A_0 = \frac{2}{T_0} \int_0^{T_0} x(t)dt \tag{3}$$

$$A_n = \frac{2}{T_0} \int_0^{T_0} x(t) \cdot \sin(n \cdot \omega_0 \cdot t)dt \tag{4}$$

$$B_n = \frac{2}{T_0} \int_0^{T_0} x(t) \cdot \cos(n \cdot \omega_0 \cdot t)\, dt \tag{5}$$

gdje je:

T_0 – osnovni period

Analiza aperiodičkih signala zahtijeva drukčiji pristup, stoga je razvijena Fourierova transformacija, gdje je korištena metoda tretiranja aperiodičkog signala kao periodičkog. Iz toga proizlazi da osnovni period T_0 aperiodičkog signala teži u beskonačnost. Aperiodički signali nisu sastavljeni od diskretnih spektralnih komponenti, već imaju kontinuirani frekvencijski spektar s frekvencijski ovisnom spektralnom gustoćom.

Spektar kontinuiranih aperiodičkih signala definira se slijedećim jednadžbama Fourierove transformacije:

Jednadžba analize:

$$X(f) = \int_{-\infty}^{+\infty} x(t)e^{-j2\pi ft}\,dt \tag{6}$$

Jednadžba sinteze:

$$x(t) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} X(f)e^{j2\pi ft}d\omega \tag{7}$$

Gdje je:

x(t) – signal u vremenskoj domeni

X(f) – signal u frekvencijskoj domeni

f– frekvencija

t – vrijeme

ω – kutna frekvencija

2.2.4. Diskretna Fourierova transformacija

Od velike važnosti u svim područjima obrade signala je upravo Diskretna Fourierova Transformacija (DFT). Koristi se za izračun diskretiziranog frekvencijskog spektra signala, te se zatim iz toga još i proračunava mnoštvo bitnih svojstava u obradi, analizi i klasifikaciji audio signala. Za dobivanje DFT koeficijenata koristi se tzv. Brza Fourierova Transformacija (FFT) kao algoritam na računalima i DSP (Digital Signal Processor) čipovima iz razloga što algoritam brzo konvergira.

Ako je zadan diskretni signal u vremenskom području, *x(n), n = 0,..., N-1*, broja uzoraka *N*, DFT se definira jed. 8:

$$X(k) = \sum_{n=0}^{N-1} x(n) exp\left(-j\frac{2\pi}{N}kn\right), \qquad k = 0, \dots, N-1, \tag{8}$$

Gdje je $j \equiv \sqrt{(-1)}$. Uočljivo je da je rezultat transformacije niz N koeficijenata od X(k), koji su generalno kompleksni brojevi. DFT koeficijenti sadrže u sebi frekvencijski spektar signala (magnituda) i fazni kut signala.

Inverzna DFT kao ulazne podatke uzima DFT koeficijente i kao rezultat vraća izvorni signal:

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k) exp\left(j\frac{2\pi}{N}kn\right), \qquad n = 0, \dots, N-1\,. \tag{9}$$

Kao rezultat toga, signal u vremenskoj domeni, x(n), n = 0,..., N-1 i kompleksni signal, X(k), k = 0,..., N-1 mogu se tretirati kao ekvivalentni prikazi signala. Ako jednadžbu IDFT napišemo u obliku kao što prikazuje jed. 10:

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k)\, y_k(n), \qquad n = 0, \dots, N-1\,, \tag{10}$$

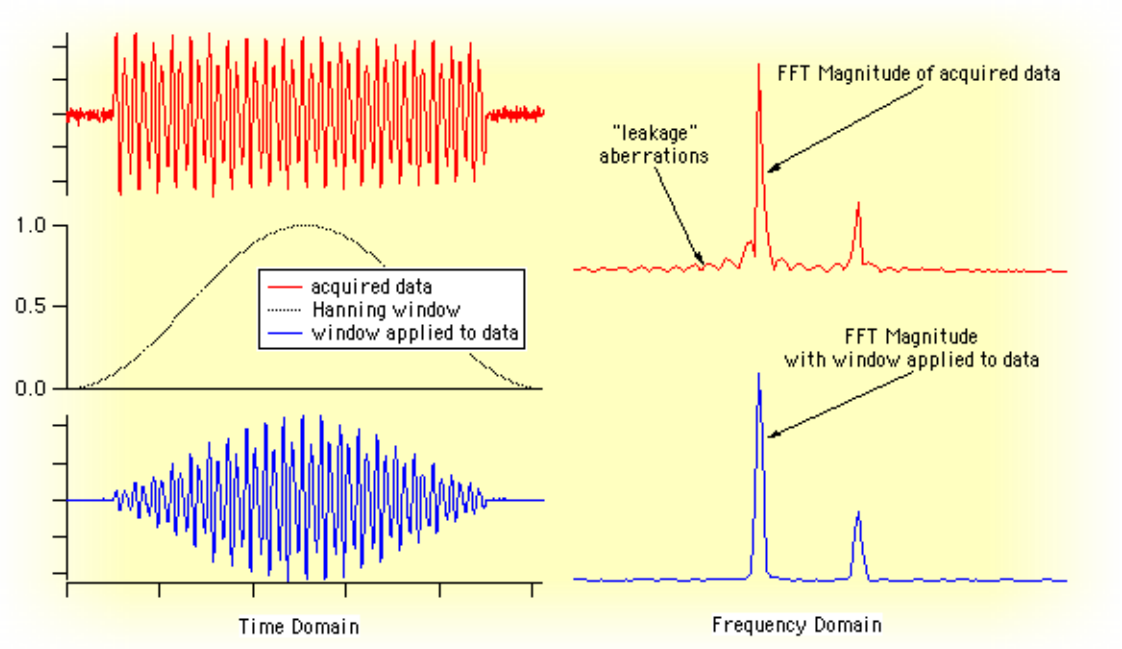
gdje je $y_k = exp\left(j\frac{2\pi}{N}kn\right), n = 0, \dots, N-1$, tada se može uočiti da se izvorni signal, x(n), može napisati kao težinski prosjek skupine osnovnih funkcija, gdje je svaki signal, $y_k(n)$, kompleksni eksponencijalni broj i njegova težina je jednaka k-tom DFT koeficijentu [10].

3. PREPOZNAVANJE GOVORA POMOĆU NEURONSKIH MREŽA

3.1. IZDVAJANJE SVOJSTAVA

Izdvajanje svojstava je bitan dio procesa analize signala, pogotovo u poljima prepoznavanja uzoraka i strojnog učenja. Cilj tog procesa je izvaditi željeni skup svojstava iz skupa podataka koji se promatraju. Željena svojstva se bira ovisno o vrsti informacije koju se želi dobiti iz podataka. Vađenje svojstava iz signala se također može smatrati kao način za smanjenje broja podataka, što je poželjno kod analitičkih algoritama gdje se cilja na što manji broj podataka koji sadrže što veću količinu korisnih informacija za izvršavanje željenog cilja. Kod analize audio signala to je najznačajniji proces s obzirom na veliku količinu podataka, koja značajno varira s obzirom na frekvenciju uzorkovanja signala. Da bi se taj proces uspješno ostvario, potrebno je imati kvalitetno znanje o području primjene, tako da se mogu primijeniti najbolja svojstva u tu svrhu [10].

Kod većine primjena, audio signal je analiziran u tzv. kratkotrajnim okvirima, gdje se signal podijeli u željeni broj okvira kratkog vremena trajanja, koji se po potrebi mogu i preklapati. Glavni razlog korištenja ove metode je što su audio signali nestacionarni po prirodi, što znači da se njihova svojstva bitno mijenjaju u vremenskom području, primjerice buka ili šum su signali drukčije jakosti od uobičajenog razgovora ljudi, a često se nalaze zajedno u istom signalu. Zbog navedenoga se promatrani signal dijeli u prozore gdje se smatra stacionarnim.



Slika 11. Prednost korištenja „Hanning window“ funkcije [18]

Za primjenu okvirne ili prozorske funkcije, neka je *x(n), n = 0,..., N – 1*, audio signal veličine *N* uzoraka. Tijekom kratkotrajnog procesuiranja, okvir originalnog signala željene veličine se pomnoži sa prozorskom funkcijom fiksne veličine *w(n)*, pomaknutom u vremenu za m_i. Rezultirajući signal, $x_i(n)$, na *i-tom* koraku procesuiranja je prikazan jednadžbom 11:

$$x_i(n) = x(n)w(n - m_i), \quad i = 0, \dots, K - 1, \tag{11}$$

gdje je *K* broj okvira i m_i pomak u vremenu, odnosno broj uzoraka za koji je prozorska funkcija pomaknuta da bi se dobio *i-ti* okvir. Jednadžba 11 implicira da je $x_i(n)$ jednak nuli u svim uzorcima osim u području uzoraka sa indeksima $m_i, \dots, m_i + W_L - 1$, gdje je W_L jednak dužini pomicajućeg prozora u broju uzoraka. Vrijednost od m_i ovisi o veličini prozora koji se pomiče, W_S. Primjerice, ako je prozor pomaknut za 10 ms na svakom koraku i frekvencija uzorkovanja, F_s je 16 kHz, tada je $m_i = i \cdot W_S \cdot F_S = i \cdot 0.01 \cdot 16000 = i \cdot 160$ uzoraka, $i = 0, \dots, K - 1$. Nadalje, ako je W_L = 300 uzoraka, onda peti okvir (i = 4) počinje na indeksu uzoraka $160 \cdot 4 = 640$ i završava na indeksu uzoraka $160 \cdot 4 + 300 - 1 = 939$. Tipična duljina prozora varira od 10 ms do 50 ms. Množenje s prozorskom funkcijom, osim što postavlja signal van okvira u nulu ima i funkciju smanjivanja pojavljivanja novih frekvencijskih komponenti na krajevima signala. U protivnom analiza i dijeljenje signala u vremenske okvire ne bi imalo smisla.

Ovaj pristup se koristi tijekom postupka vađenja svojstava iz signala, audio signal se podijeli na okvire, sa ili bez preklapanja okvira, te se za svaki okvir izracunavaju željena svojstva. To procesuiranje rezultira u generiranju niza svojstava, odnosno vektora svojstava od izvornog audio signala. Dimenzije vektora svojstava ovise o svojstvima koja se koriste te broju okvira u koje je podijeljen signal [10].

Svojstva koja se koriste u ovom radu, za kreiranje algoritma za raspoznavanje izgovorenih riječi su slijedeća:

- Zero Crossing Rate (ZCR)
- Mel-Frequency Cepstrum Coefficients (MFCCs)
- Spectral Centroid
- Spectral Rolloff

3.1.1. Stopa prelaska nule (ZCR)

Stopa prijelaza nule u signalu, od određenog okvira u audio signalu, je mjera promjene predznaka vrijednosti signala tokom trajanja tog okvira signala. Drugim riječima, to je broj prijelaza vrijednosti signala iz pozitivne u negativnu vrijednost ili negativne u pozitivnu, podijeljen sa duljinom okvira.

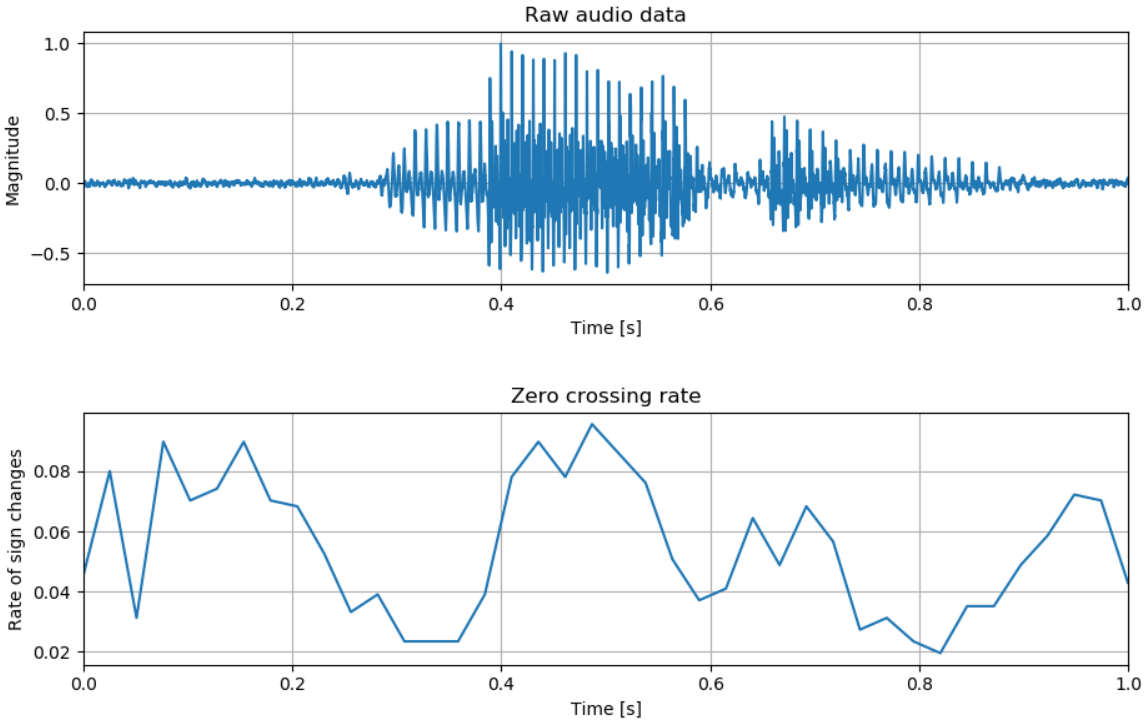
Neka je $x_i(n)$, $n = 1,\ldots, W_L$ niz audio uzoraka od *i-tog* okvira, gdje je W_L duljina okvira. ZCR se računa po jednadžbi 12:

$$Z(i) = \frac{1}{2W_L}\sum_{n=1}^{W_L} |sgn[x_i(n)] - sgn[x_i(n-1)]| \tag{12}$$

gdje je

$$sgn[x_i(n)] = \begin{cases} 1, & x_i(n) \geq 0 \\ -1, & x_i(n) < 0 \end{cases}$$

ZCR se može protumačiti kao mjera šuma u signalu. Npr. u slučaju povećanja šuma raste i vrijednost signala. Kod ovog svojstva je primjećeno da se kod binarne klasifikacije između govora i muzike korištenjem standardne devijacije dobiva veća preciznost [10].



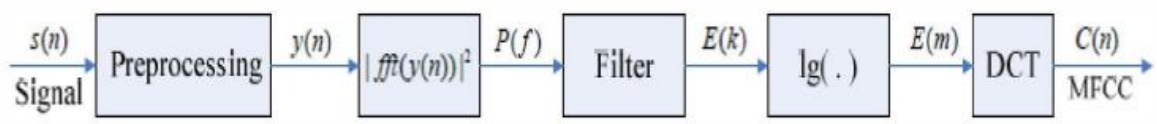
Slika 12. ZCR signal

3.1.2. Kepstralni koeficijenti po Mel-skali (MFCCs)

Opažanje zvuka u ljudskom uhu koristi nelinearno rješenje, zbog kojeg je za simuliranje ovog svojstva razvijena Mel-frekvencijska skala. Eksperimentalni rezultati su pokazali da slušni sustav čovjekovog uha ima visoku rezoluciju u niskim frekvencijama, te nisku rezoluciju u visokim frekvencijama. Formula Mel skale koja približno točno prikazuje taj odnos je u jed. 13:

$$Mel(f) = 2595 \log\left(1 + \frac{f}{700}\right) \tag{13}$$

gdje je *f* frekvencija. U praktičnoj primjeni, izračun MFCC-a (*engl. Mel-frequency Cepstral Coefficients*) se može podijeliti u nekoliko koraka, slika 15 prikazuje blok dijagram za MFCC algoritam.



Slika 15. Blok dijagram MFCC algoritma [23]

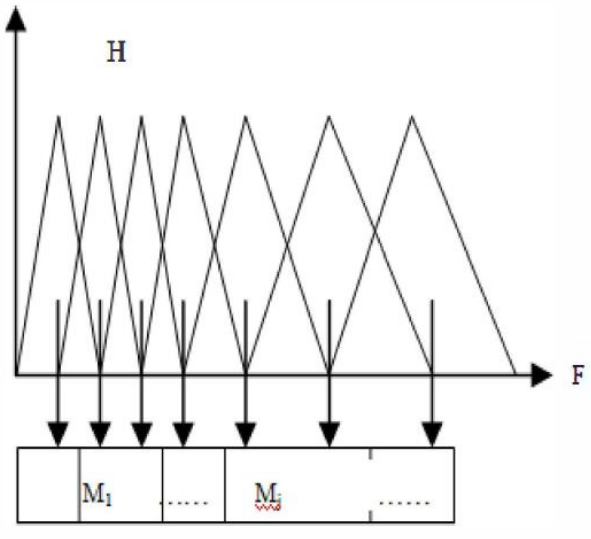
Koraci potrebni za izračun MFCC-a su sljedeći:

1. Predprocesuiranje signala, koje uključuje podjelu signala na okvire, i dodavanje prozorske funkciju okviru, nakon kojeg se dobiva signal okvira u vremenskoj domeni, *y(n)*.

2. Fourierova transformacija u kratkom vremenu, *engl. Short-Time Fourier Transform* (STFT), čime se signal u vremenskoj domeni $y(n)$, transformira u signal u frekvencijskoj domeni $Y(f)$. Zatim se se dobiva spektar energije signala $P(f)$, primjenjujući jednadžbu 14:

$$P(f) = |Y(f)|^2 \tag{14}$$

3. Filtriranje spektra energije signala Mel skalom, točnije sa *M* trokutastih filtera Mel skale, *m = 1, 2, ..., M*, čiji frekvencijski odziv izgleda kao na slici 16.



Slika 16. Mel filter [23]

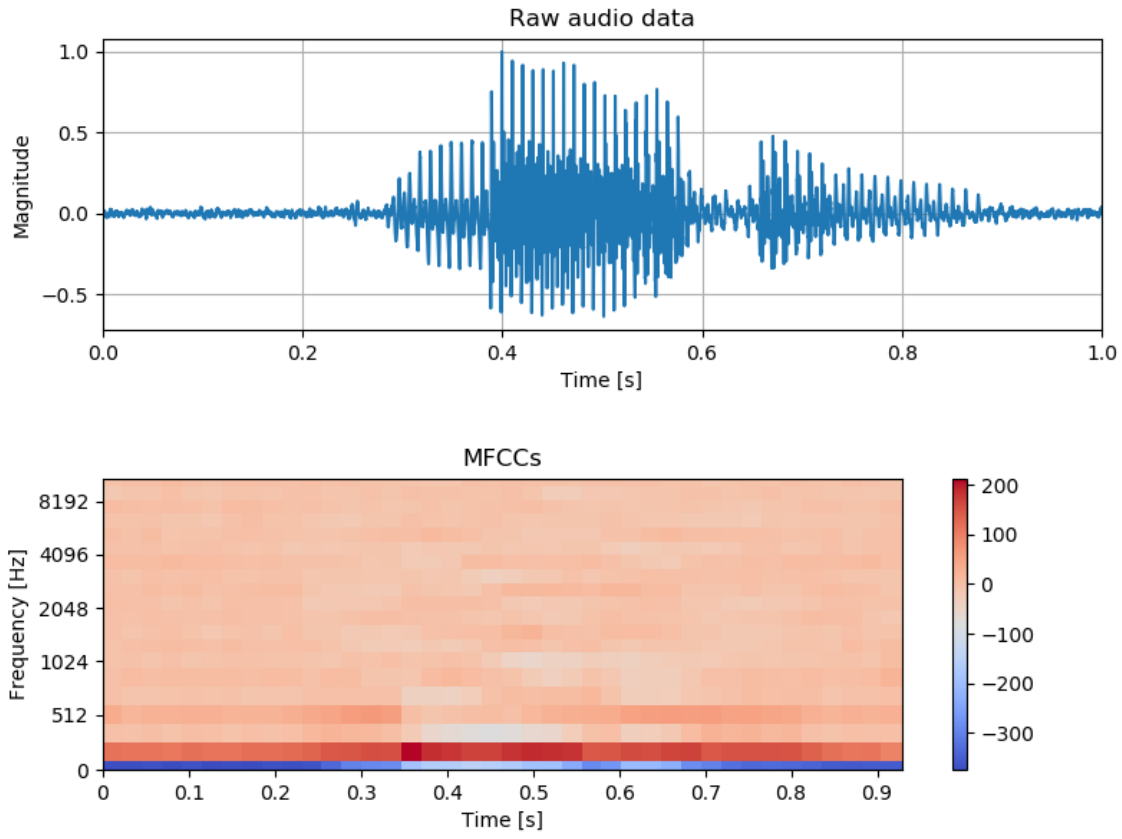
4. Logaritamski spektar energije, kojim se u skladu sa svojstvom slušnog sustava čovjeka, Mel – filtrirani spektar energije komprimira koristeći logaritamsku funkciju, prikazan jed. 15:

$$E(m) = \ln\left(\sum_{k=0}^{N-1} |Y(f)|^2 \, H_m(f)\right), m = 1, 2, \ldots M \tag{15}$$

5. MFCC se dobiva transformacijom logaritamskog spektra energije u vremensku domenu koristeći Diskretnu Kosinusnu Transformaciju, *engl. Discrete Cosine Transform (DCT)*:

$$C(n) = \sum_{m=1}^{M} E(m) \cos\left(\frac{\pi(m-0.5)n}{M}\right), n = 1, 2, \dots p \tag{16}$$

gdje je p potrebni broj MFCC koeficijenata. DCT eliminira relativnost u vektoru svojstava i minimizira kvadrat prosjeka greške, *engl. Mean Square Error (MSE)*. Kada je p velikog iznosa, MFCC teži u nulu, točnije, ako je p veći od 20, MFCC se može zanemariti u praksi [23]. MFCC ima široki spektar primjene u područjima prepoznavanja glasa, klasifikacije muzičkih žanrova, raspoznavanja govornika i ostalih primjena u audio analizi [10].



Slika 17. MFCC koeficijenti

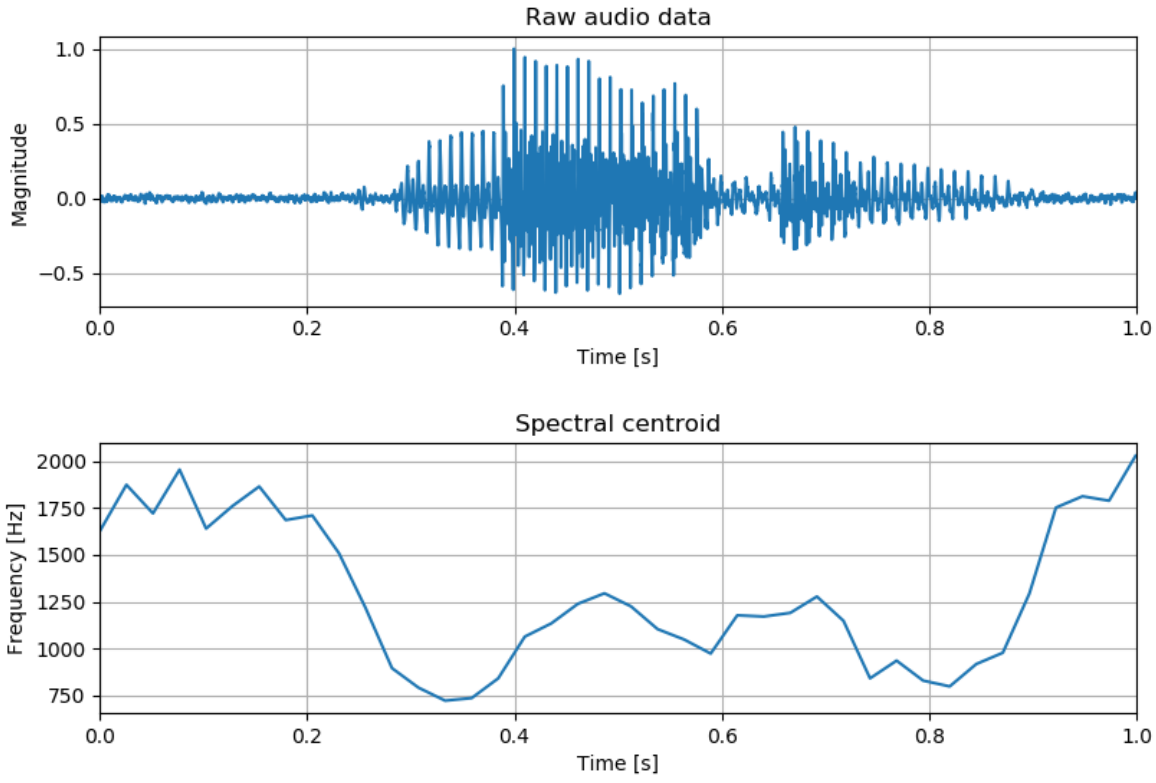
3.1.3. Spektralni centroid

Da bi se izračunala svojstva u frekvencijskoj domeni, prvo je potrebno dobiti vrijednost signala u frekvencijskoj domeni. Zatim se uzme da je $X_i(k)$, $k = 1,\ldots, W_{FL}$, magnituda DFT koeficijenata od i-tog audio okvira, gdje je W_{FL} broj koeficijenata u frekvencijskom spektru koji iznosi polovinu od W_L.

Spektralni centroid je „težište“ frekvencijskog spektra signala. Kod ovog svojstva je značajka da više vrijednosti odgovaraju višim frekvencijama zvuka [10].

Izraz za vrijednost spektralnog centroida danog okvira je:

$$C_i = \frac{\sum_{k=1}^{W_{fL}} kX_i(k)}{\sum_{k=1}^{W_{fL}} X_i(k)} \tag{17}$$



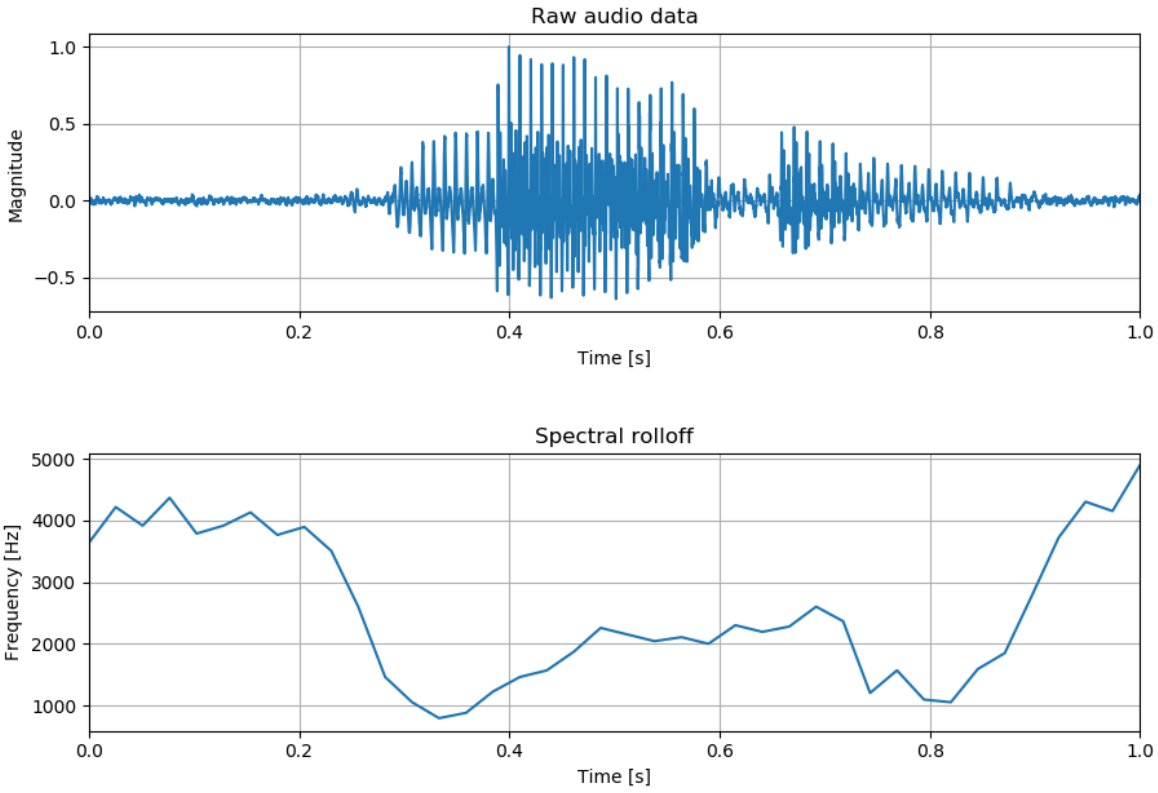
Slika 15. Spektralni centroid

3.1.4. Spektralno opadanje

Ovo svojstvo se definira kao frekvencija ispod koje je koncentriran određeni postotak (obično oko 90 %, u ovom radu je 85 %) distribucije magnitude spektra. DFT koeficijent *m* odgovara spektralnom opadanju okvira *i* [10], po izrazu jed. 18:

$$\sum_{k=1}^{m} X_i(k) = C \sum_{k=1}^{W_{fL}} X_i(k) \tag{18}$$

gdje je C parametar u iznosu postotka, koji bira korisnik. Frekvencija je uobičajeno normalizirana dijeljenjem sa duljinom prozora frekvencijskog spektra W_{fL}, tako da ima vrijednosti između 0 i 1. Ovaj tip normalizacije implicira da vrijednost iznosa 1 odgovara maksimalnoj frekvenciji signala, odnosno polovini frekvencije uzorkovanja. Ovo svojstvo opisuje spektralni oblik audio signala i može se zbog toga koristiti za razlikovanje signala govora od ostalih zvukova [10].

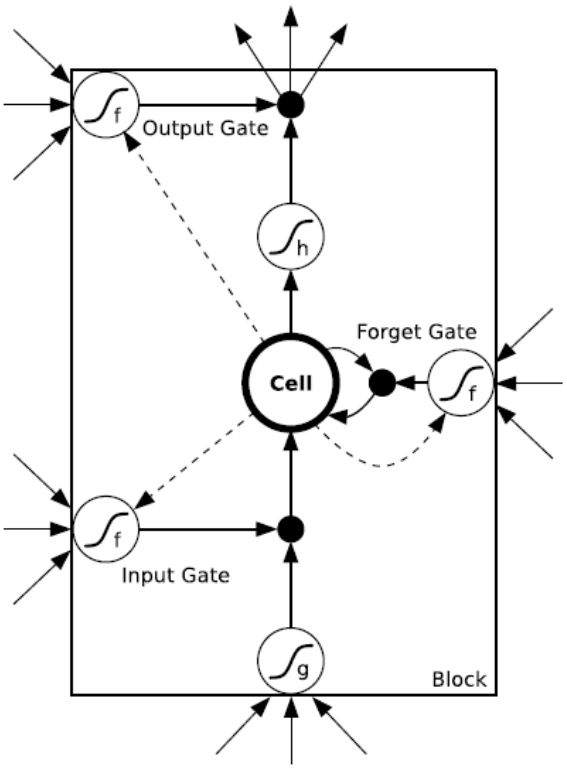


Slika 19. Spektralno opadanje

3.2. LSTM NEURONSKA MREŽA

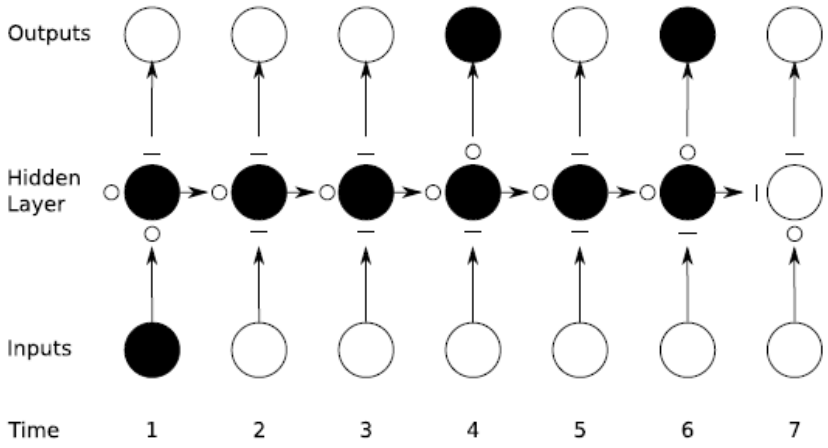
U prepoznavanju govora, odnosno obrađivanju niza podataka, kao koristan alat su se pokazale neuronske mreže sa povratnom vezom (*engl. Recurrent Neural Networks -RNN*). Takve neuronske mreže su pokazale značajan uspjeh u prepoznavanju i označavanju nizova podataka poput ručno pisanih podataka i u modeliranju riječi. Razlog uspjeha u područjima obrade niza podataka je što RNN za razliku od tradicionalne neuronske mreže ima sposobnost pamćenja prethodnih informacija, kako bi se učinkovitije prepoznale nove informacije. Međutim u kompleksnijim zadaćama, pokazalo se da obične RNN nisu sposobne povezivati informacije za efikasno prepoznavanje [19]. Naročito se to pokazalo u području akustike gdje su se kao najučinkovitije rješenje pokazale Duboke Neuronske Mreže (DNN). DNN s druge strane omogućuju uglavnom samo procesuiranje vremenskog prozora fiksne duljine kod akustičkih okvira.

Sposobnost pamćenja se time kod neuronskih mreža pokazala ključnom za obradu niza podataka, te je očigledno bilo potrebno uvesti neuronsku mrežu sa boljim sposobnostima pamćenja informacija na duže periode. Kao rješenje tog problema, 1997. je objavljen znanstveni rad o neuronskim mrežama sa dugoročnom memorijom (*engl. Long Short Term Memory – LSTM*) [20]. LSTM arhitektura se sastoji od skupa povratno spojenih pod-mreža, koji se nazivaju memorijski blokovi. Svaki blok se sastoji od jedne ili više memorijskih ćelija i tri multiplikacijske jedinice: ulazna i izlazna vrata, te vrata za zaboravljanje, koja omogućuju kontinuirane operacije analogne pisanju, čitanju i resetiranju za ćelije. LSTM mreža se od RNN mreže razlikuje isključivo u činjenici da sadrži memorijske blokove, dok je RNN mreža na istom mjestu u skrivenom sloju sadržavala jedinice za zbrajanje. Slika 11 prikazuje LSTM memorijski blok sa jednom ćelijom.



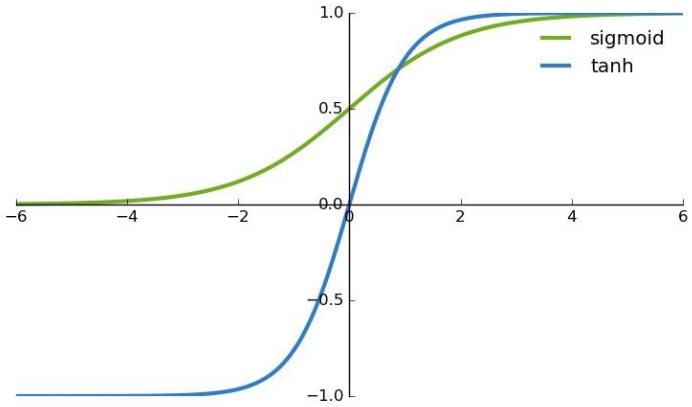
Slika 11. LSTM memorijski blok sa jednom ćelijom [21]

Tri navedena vrata su nelinearne jedinice za zbrajanje koje sakupljaju aktivacije iznutra i izvana bloka, te kontroliraju aktivacije ćelije preko umnožavanja. Ulazna i izlazna vrata množe izlaz i ulaz ćelije dok se vrata za zaboravljanje množe sa prethodnim stanjem ćelije. Unutar ćelije se ne primjenjuje aktivacijska funkcija. Navedena vrata omogućuju memorijskim ćelijama LSTM mreže pohranu i pristup informacijama tijekom dugih perioda vremena, čime se rješava problem RNN mreža. Primjerice, dok su ulazna vrata zatvorena (aktivacija im je približno nula), aktivacija ćelije neće biti prebrisana dolaskom novih ulaznih informacija, već time ostaje sačuvana do kasnijeg dijela niza, kada postaje dostupna otvaranjem izlaznih vrata [21]. Očuvanje informacije je prikazano na slici 12. Stanja ulaznih i izlaznih vrata, te vrata za zaboravljanje su prikazana ispod, lijevo i iznad skrivenog sloja. Stanje vrata je prikazano sa kružićem za otvoreno ('O'), te crtom za zatvoreno ('-'). Memorijska ćelija pohranjuje prvu ulaznu informaciju dok god su vrata za zaboravljanje otvorena i ulazna vrata zatvorena.



Slika 12. Očuvanje informacije u LSTM mreži [23]

Aktivacijska funkcija vrata 'f' je uobičajeno logistički sigmoid ($\sigma(x) = \frac{1}{1+e^{-x}}$), tako da su aktivacije vrata između 0 (zatvorena vrata) i 1 (otvorena vrata). Aktivacijske funkcije ulaza i izlaza iz bloka ('g' i 'h') su uobičajeno hiperbolička tangenta ($tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}$) ili logistički sigmoid (slika 12).



Slika 13. Logistički sigmoid i tanh funkcija [24]

Isprekidane linije prikazuju težinsku vezu od ćelije prema vratima, koja služi kako bi ćelija kontrolirala vrata, odnosno za učenje preciznijeg tajminga u radu LSTM bloka [24]. Bitno je i napomenuti da postoje razne varijacije LSTM mreže u odnosu na originalni rad, koji je sadržavao samo ulazna i izlazna vrata [20][21].

Za prikaz slijeda aktivacija i izračuna gradijenta LSTM bloka, koji služi za optimizaciju težinskih vrijednosti, neka x^t bude ulazni vektor za vrijeme *t*, neka je *N* broj LSTM blokova i *M* broj ulaza. Tada se dobivaju slijedeće težinske matrice za LSTM sloj:

- Ulazne težine: W_z, W_i, W_f, $W_o \in \mathbb{R}^{N \times M}$
- Povratne težine: R_z, R_i, R_f, $R_o \in \mathbb{R}^{N \times N}$
- Težine od ćelije prema vratima: p_i, p_f, $p_o \in \mathbb{R}^N$
- Težine pristranosti: b_z, b_i, b_f, $b_o \in \mathbb{R}^N$

Zatim su vektorske formule za LSTM sloj koji se širi unaprijed,odnosno prema izlazu, slijedeće za:

Ulaz bloka

$$\overline{z^t} = W_z x^t + R_z y^{t-1} + b_z \tag{19}$$

$$z^t = g(\overline{z^t}) \tag{20}$$

Ulazna vrata

$$\overline{i^t} = W_i x^t + R_i y^{t-1} + p_i \odot c^{t-1} + b_i \tag{21}$$

$$i^t = \sigma\left(\overline{i^t}\right) \tag{22}$$

Vrata za zaboravljanje

$$\overline{f^t} = W_f x^t + R_f y^{t-1} + p^f \odot c^{t-1} + b_f \tag{23}$$

$$f^t = \sigma\left(\overline{f^t}\right) \tag{24}$$

Ćelija

$$c^t = z^t \odot i^t + c^{t-1} \odot f^t \tag{25}$$

Izlazna vrata

$$\overline{o^t} = W_o x^t + R_o y^{t-1} + p_o \odot c^t + b_o \tag{26}$$

$$o^t = \sigma\left(\overline{o^t}\right) \tag{27}$$

Izlaz bloka

$$y^t = h(c^t) \odot o^t \tag{28}$$

gdje su σ, g i h nelinearne aktivacijske funkcije. Skalarno množenje vektora je označeno sa $\odot$. Formule za propagaciju ili širenje unazad kroz vrijeme t unutar LSTM bloka se računaju po slijedećim izrazima:

$$\delta y^t = \Delta^t + R_z^T \delta z^{t+1} + R_i^T \delta i^{t+1} + R_f^T \delta f^{t+1} + R_o^T \delta o^{t+1} \tag{29}$$

$$\delta\overline{o^t} = \delta y^t \odot h(c^t) \odot \sigma'(\overline{o^t}) \tag{30}$$

$$\delta c^t = \delta y^t \odot o^t \odot h'(c^t) + p_o \odot \delta\overline{o^t} + p_i \odot \delta\overline{i^{t+1}} + p_f \odot \delta\overline{f^{t+1}} + \delta\overline{c^{t+1}} \odot f^{t+1} \tag{31}$$

$$\delta\overline{f^t} = \delta c^t \odot c^{t-1} \odot \sigma'\left(\overline{f^t}\right) \tag{32}$$

$$\delta\overline{i^t} = \delta c^t \odot z^t \odot \sigma'\left(\overline{i^t}\right) \tag{33}$$

$$\delta\overline{z^t} = \delta c^t \odot i^t \odot g'\left(\overline{z^t}\right) \tag{34}$$

Gdje je Δ^t vektor delta vrijednosti koji se prenosi sa prijašnjeg sloja. Ako je E funkcija gubitka, taj vektor odgovara rezultatu od $\frac{\partial E}{\partial y^t}$. Delta vrijednosti za ulaze su jedino potrebne ako je ispod sloj koji zahtijeva treniranje, i može se izračunati na slijedeći način:

$$\delta x^t = W_z^T \delta\overline{z^t} + W_i^T \delta\overline{i^t} + W_f^T \delta\overline{f^t} + W_o^T \delta\overline{o^t} \tag{35}$$

Konačno, gradijenti za težine se računaju na slijedeći način, gdje ⋆ može biti bilo koji element skupa $\{\overline{z}, \overline{i}, \overline{f}, \overline{o}\}$, a $\langle \star_1, \star_2 \rangle$ označava produkt dvaju vektora:

$$\delta W_\star = \sum_{t=0}^{T} \langle \delta \star^t, x^t \rangle \tag{36}$$

$$\delta R_\star = \sum_{t=0}^{T-1} \langle \delta \star^{t+1}, y^t \rangle \tag{37}$$

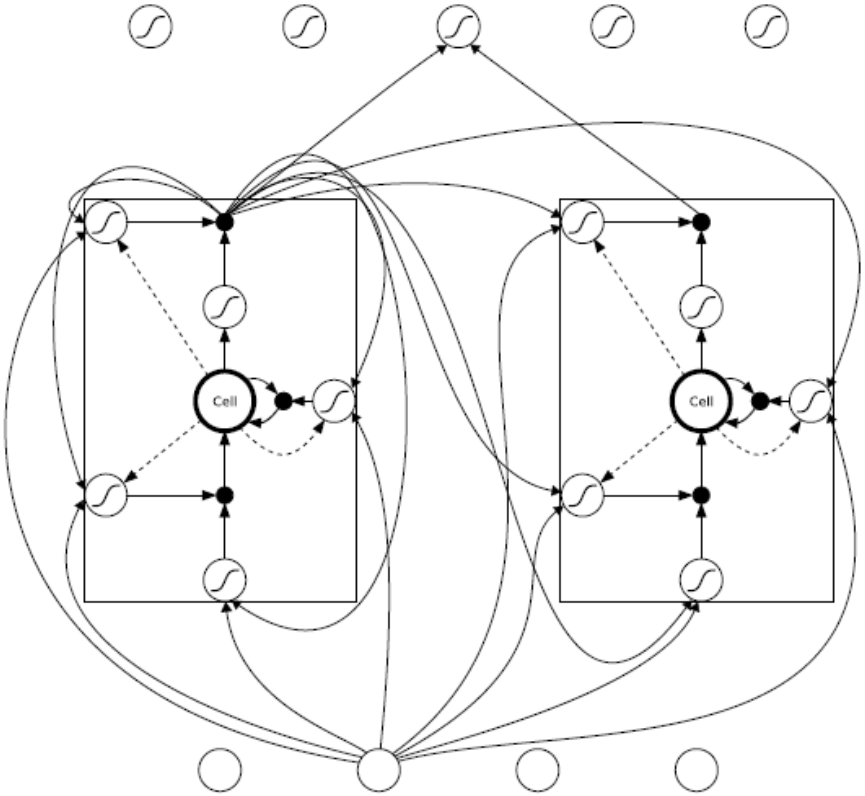
$$\delta b_\star = \sum_{t=0}^{T} \delta \star^t \tag{38}$$

$$\delta p_i = \sum_{t=0}^{T-1} c^t \odot \delta \overline{i^{t+1}} \tag{39}$$

$$\delta p_f = \sum_{t=0}^{T-1} c^t \odot \delta \overline{f^{t+1}} \tag{40}$$

$$\delta p_o = \sum_{t=0}^{T} c^t \odot \delta \overline{o^t} \tag{41}$$

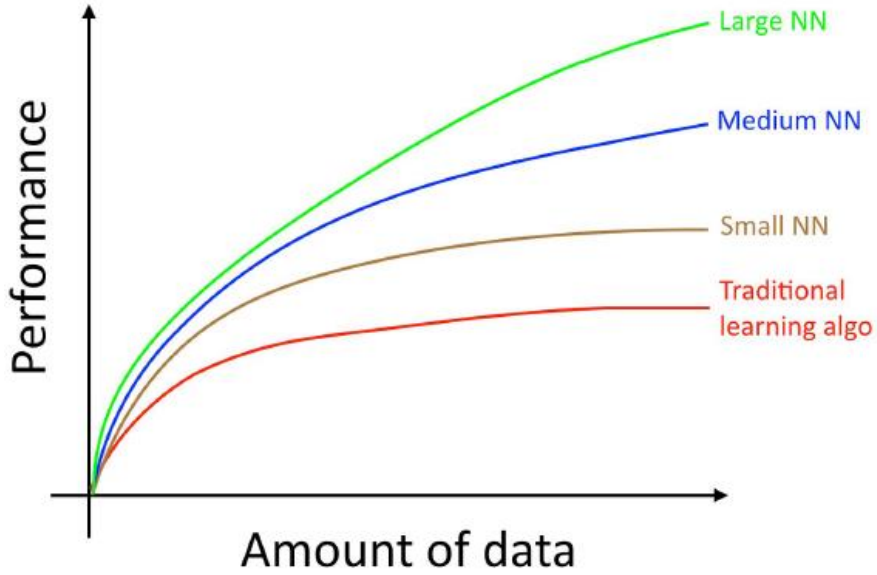
Na slici 14 je prikazan primjer LSTM mreže sa 2 memorijska bloka, od kojih je svaki sa jednom ćelijom, 4 ulazne jedinice i 5 izlaznih jedinica, neke od težinskih veza su izostavljene zbog jednostavnijeg prikaza.



Slika 14. LSTM mreža sastavljena od 4 ulazne jedinice i 2 LSTM memorijska bloka (bez prikazanih svih težinskih veza) [21]

4. PREPOZNAVANJE GOVORA U PROGRAMSKOM JEZIKU PYTHON

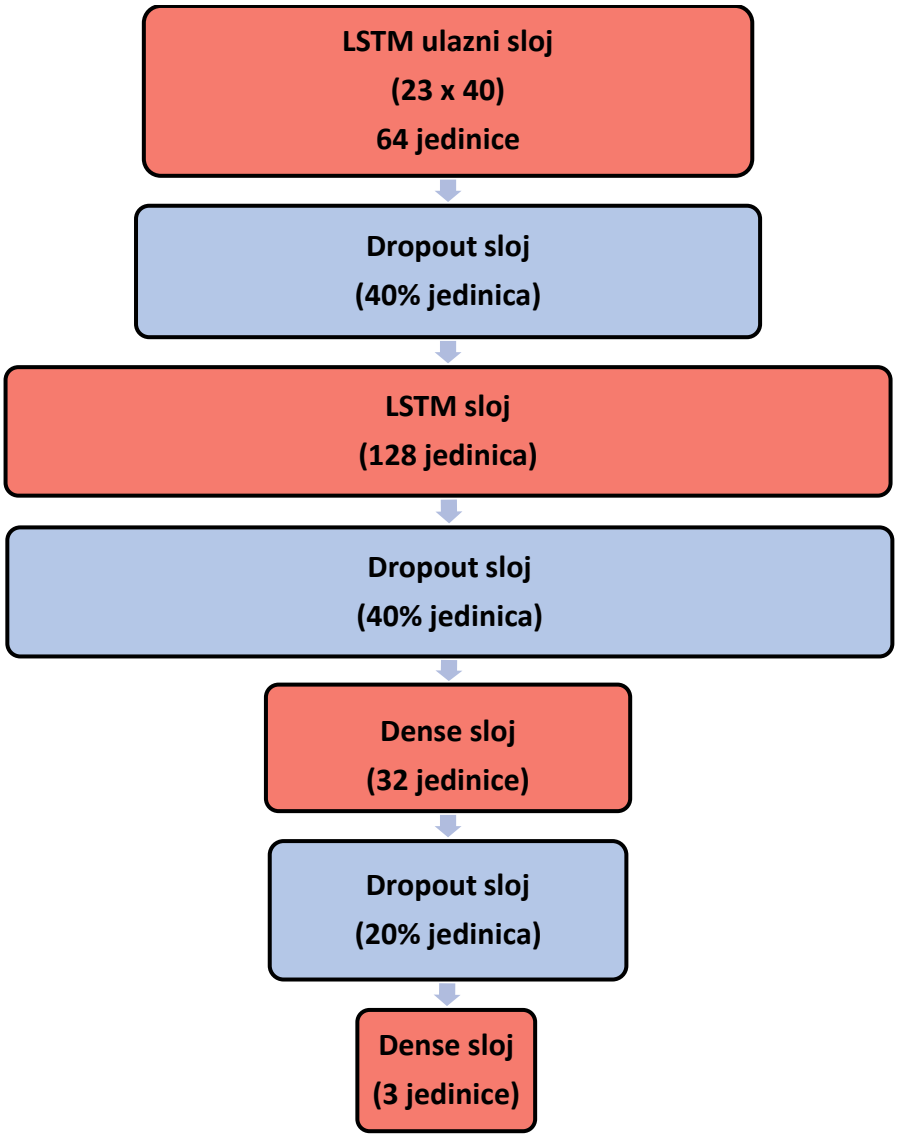
U ovom programu izvršavalo se prepoznavanje izgovorenih riječi, odnosno klasifikacija između riječi „Yes“, „No“ i „Marvin“. Podaci na kojima se trenirao algoritam za navedene riječi su preuzeti iz skupa podataka glasovnih naredbi koji su objavljeni od strane *Tensorflow-a,* besplatne platforme, otvorene javnosti za prakticiranje strojnog učenja, koju je razvio *Google Brain* tim [17]. Podaci sadrže izgovorene riječi od strane velikog broja govornika u različitim okolnostima u trajanju od jedne sekunde, uzorkovane frekvencijom uzorkovanja od 16 kHz. Različite okolnosti poput buke i raznih pozadinskih zvukova, doprinose težini klasifikacije. Zbog istog razloga tijekom treniranja algoritma dobiva se bolja generalizacija. Pojam generalizacije u ovom smislu se odnosi na kvalitetu učenja algoritma, odnosno koliko se dobro koncepti koje je algoritam naučio, primjenjuju na nove podatke, koji su još neviđeni od strane algoritma. Time se stvara bolji klasifikator za nove podatke, koji nije pristran prema podacima na kojima se već vršilo treniranje. Za doprinos preciznosti i generalizaciji u neuronskim mrežama, a naročito u slučaju dubokih neuronskih mreža kao što je i slučaj u ovom programu, korisno je imati veliku bazu podataka na kojoj se vrši učenje algoritma. Na slici 15 je prikazan odnos učinka i količine podataka neuronskih mreža i tradicionalnih algoritama strojnog učenja.



Slika 15. Učinak neuronskih mreža i metoda strojnog učenja [30]

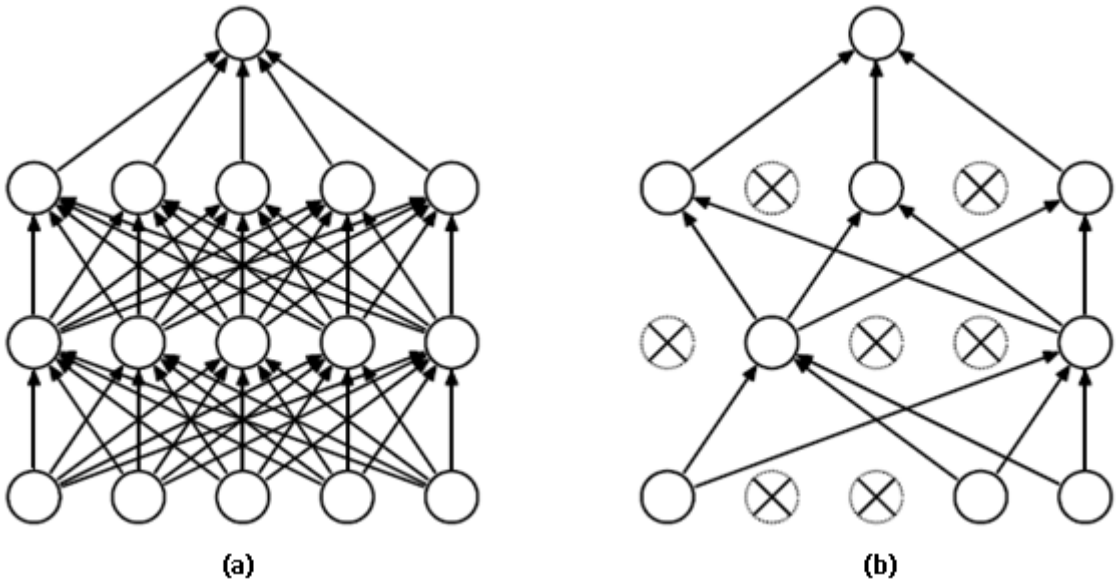
Navedena baza podataka koja je korištena u ovom modelu sadrži 6459 zvučnih zapisa. Količina podataka korištena za treniranje klasifikacijskog modela je 85 % ukupne količine podataka, dok su ostalih 15% podaci za testiranje, koji nisu dio baze podataka za treniranje modela kako bi model bio što manje pristran podacima na kojima se vrši treniranje.

Klasifikacijski model algoritma je napravljen pomoću aplikacijskog programskog sučelja *Keras,* pisanog u *Python-u,* sposobnog da radi preko *Tensorflow-a*, u ovom slučaju. To sučelje je razvijeno u svrhu brže eksperimentacija, odnosno implementacije raznih vrsta neuronskih mreža i ostalih algoritama strojnog učenja. Model počinje sa klasom *Sequential(),* unutar koje se dodaju slojevi, u ovom slučaju klasifikacijskog modela neuronske mreže. U prvi sloj se obavezno stavlja informacija o obliku ulaznih podataka, u ovom slučaju je to matrica svojstava i broja okvira za koje su svojstva izračunata. Na slici 16 je dan prikaz duboke neuronske mreže korištene u ovom radu.



Slika 16. Blok dijagram LSTM neuronske mreže od modela korištenog u ovom radu

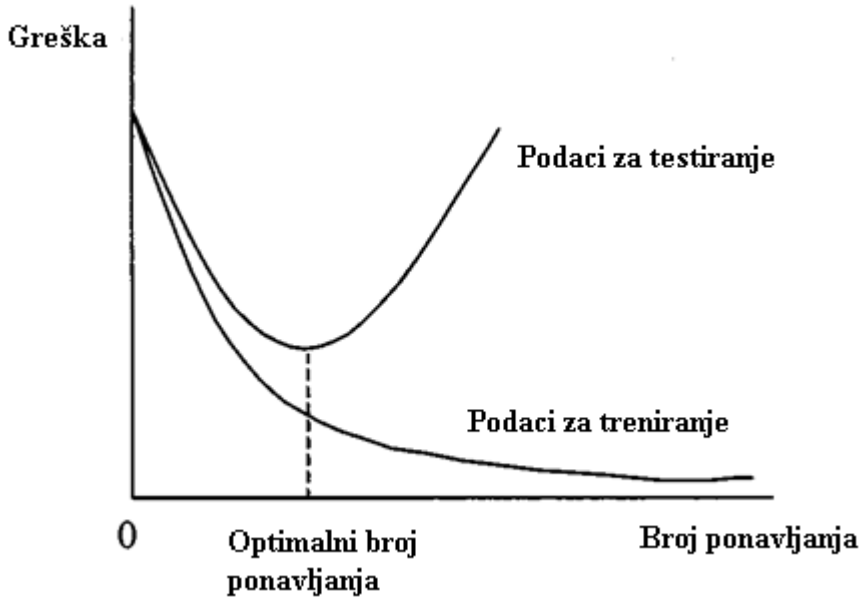
Na slici 16 su vidljivi slojevi: *LSTM*, *Dropout* i *Dense*. LSTM sloj je sloj LSTM jedinica, čiji je prvi argument dimenzija izlaznih podataka. Uz to je moguće promijeniti aktivacijske funkcije u LSTM blokovima po želji, uključiti vektor pristranosti i još razne druge postavke.*Dropout* sloj je jedna od tehnika rješavanja problema prenaučenosti ili tzv. *overfitting-a*, a koja se sastoji od pseudoslučajnog odbacivanja jedinica u neuronskoj mreži tijekom treniranja modela, kako bi se spriječilo neuronsku mrežu od prevelike prilagodbe podacima na kojima se vrši treniranje modela. Na slici 17 je prikazan izgled standardne neuronske mreže (a) i neuronske mreže nakon odbacivanja jedinica (b).



Slika 17. Standardna neuronska mreža (a) i neuronska mreža sa odbačenim jedinicama (b) [27]

Dense sloj je tipičan sloj standardne neuronske mreže koji izgleda kao jedan od dva skrivena sloja na slici 17.

Prenaučenost je problem koji se događa stvaranjem modela analize koji teži u približno savršeno odgovaranje određenom skupu podataka, koji je modelu zadan za treniranje, čime stvara pristranost prema tom skupu podataka i zbog toga ne uspijeva ostvariti dovoljno kvalitetna predviđanja za nove skupove podataka. Na slici 18 dan je prikaz tog problema, gdje je vidljivo da se s povećanjem broja ponavljanja treniranja modela, javlja velika razlika greške kod provjere modela na testnim ili validacijskim podacima u usporedbi sa podacima koji su se koristili za treniranje algoritma.



Slika 18. Utjecaj broja ponavljanja treniranja modela na preciznost modela

Zadnja komponenta modela je metoda modela *compile()* u kojoj se vrši konfiguracija modela za treniranje. U njoj se zadaje funkcija greške, optimizacijski algoritam te funkcija po kojoj će se mjeriti učinak modela.

Optimizacijski algoritam korišten u ovom modelu je tzv. *„Adam“*, koji spada u optimizacijske algoritme sa adaptirajućom stopom učenja. Ime *„Adam“* je skraćeno od izraza „adaptive moments“. Istraživanje neuronskih mreža je već odavno pokazalo da je stopa učenja jedan od najtežih parametara za postavljanje iz razloga što značajno utječe na učinak modela [28][29]. Ovaj algoritam računa adaptivne stope učenja za svaki parametar i radi po slijedećem postupku:

- Izračun eksponencijalnog težinskog prosjeka prošlih gradijenata (v_{dW}).

$$v_{dW} = \beta_1 v_{dW} + (1 - \beta_1)\frac{\partial J}{\partial W} \tag{42}$$

- Izračun eksponencijalnog težinskog prosjeka kvadrata prošlih gradijenata (s_{dW}).

$$s_{dW} = \beta_2 s_{dW} + (1 - \beta_2)\left(\frac{\partial J}{\partial W}\right)^2 \tag{43}$$

- Ti prosjeci imaju pristranost prema nuli i da bi se to izbjeglo, primjenjuje se korekcija pristranosti $\left(v_{dW}^{corrected}, s_{dW}^{corrected}\right)$.

$$v_{dW}^{corrected} = \frac{v_{dW}}{1 - \beta_1^t} \tag{44}$$

$$s_{dW}^{corrected} = \frac{v_{dW}}{1 - \beta_i^t} \tag{45}$$

- Parametri se zatim ažuriraju koristeći informaciju od izračunatih prosjeka

$$W = W - \alpha \frac{v_{dW}^{corrected}}{\sqrt{s_{dW}^{corrected} + \varepsilon}} \tag{46}$$

Gdje je:

- v_{dW} – eksponencijalni težinski prosjek prošlih gradijenata
- s_{dW} – eksponencijalni težinski prosjek kvadrata prošlih gradijenata
- β_1, β_2 – podesivi hiperparametri
- $\frac{\partial J}{\partial W}$ – gradijent trenutnog sloja
- W – matrica težina (parametar koji se ažurira)
- α – stopa učenja
- ε – vrlo mala vrijednost koja služi za izbjegavanje dijeljenja s nulom

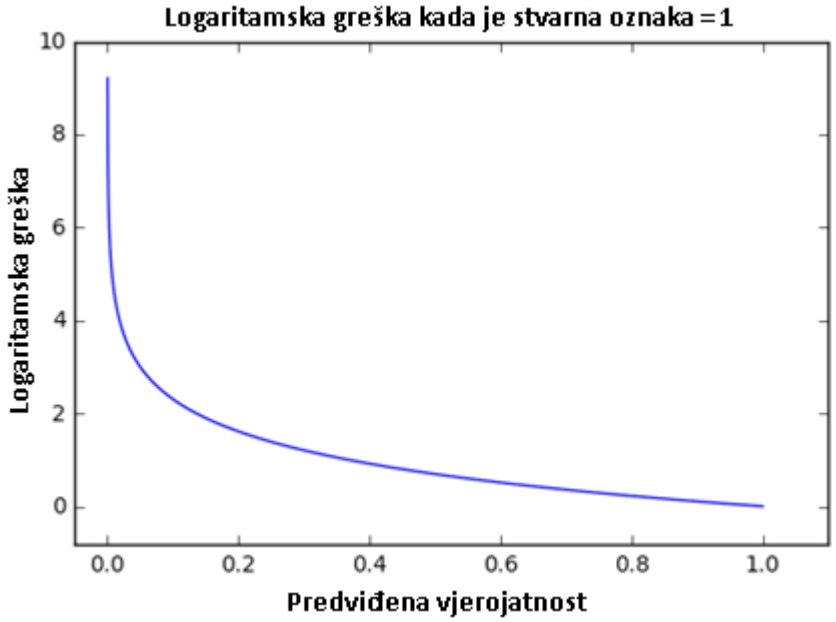
Funkcije greške se uglavnom dijele ovisno o vrsti problema, regresiji ili klasifikaciji, u regresiji se kontinuirano predviđaju vrijednosti, dok se u klasifikaciji pokušava napraviti predviđanje prijašnje kategoriziranih podataka. U ovom slučaju se rješavao problem klasifikacije, i iz tog razloga je odabrana funkcija entropijske greške, koja mjeri učinak klasifikacijskog modela gdje je izlazni rezultat vrijednost vjerojatnosti između 0 i 1.
U višeklasnoj klasifikaciji, za entropijsku grešku, uzima se suma logaritamskih greški za predviđanje svake klase u jednom promatranom primjeru, kao u jed. 47:

$$-\sum_{c=1}^{M} y_{o,c} \log(p_o, c) \tag{47}$$

gdje je:

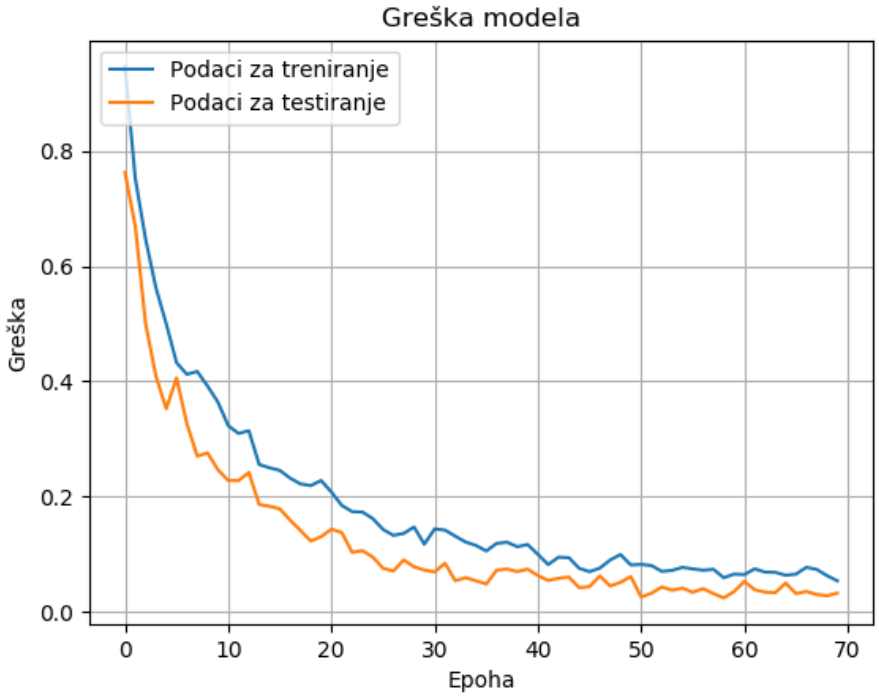
- M – broj klasa
- y – binarni indikator (0 ili 1) ako je oznaka klase c točna za promatrani primjer o
- p – predviđena vjerojatnost da je promatrani primjer o oznaka klase c

Funkcija entropijske greške se povećava sa povećavanjem razlike predviđene vrijednosti i stvarne vrijednosti. Primjerice, ako predviđena vjerojatnost iznosi 0.01, a stvarna oznaka je 1, rezultat bi bilo značajno povećanje entropijske greške. Na slici 19 je vidljiv grafički prikaz logaritamske greške, koja je u strojnom učenju istoznačnica entropijske greške. Savršeni model bi imao entropijsku grešku iznosa 0 [27].



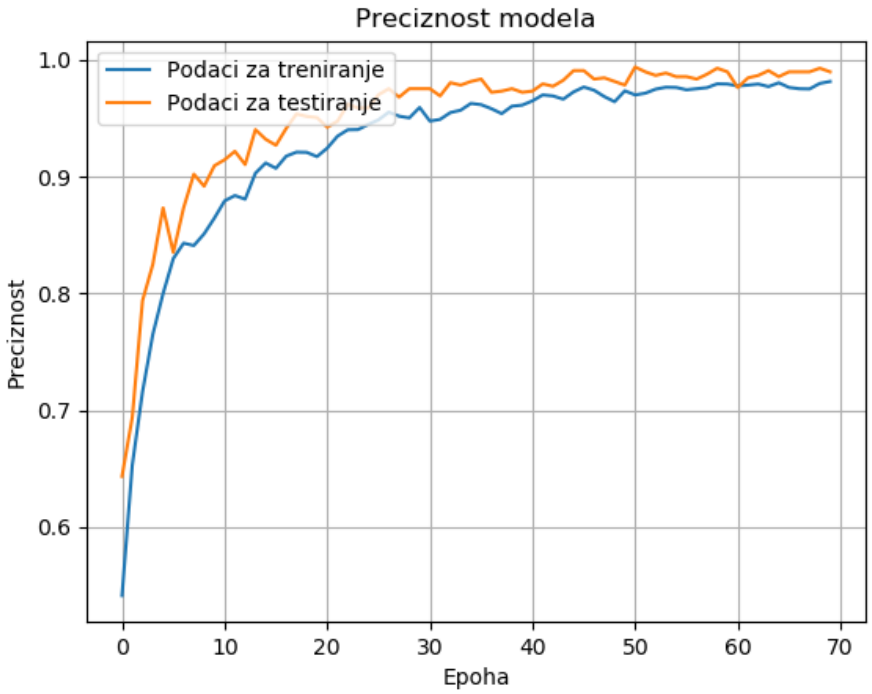
Slika 19. Logaritamska greška [29]

Na slici 20 je prikazan graf funkcije entropijske greške modela za prepoznavanje govora koji je korišten u ovom radu, u odnosu sa brojem epoha, koje označavaju broj puta primjene modela preko čitavog skupa podataka za treniranje modela kako bi se optimalnije podesila matrica težina.



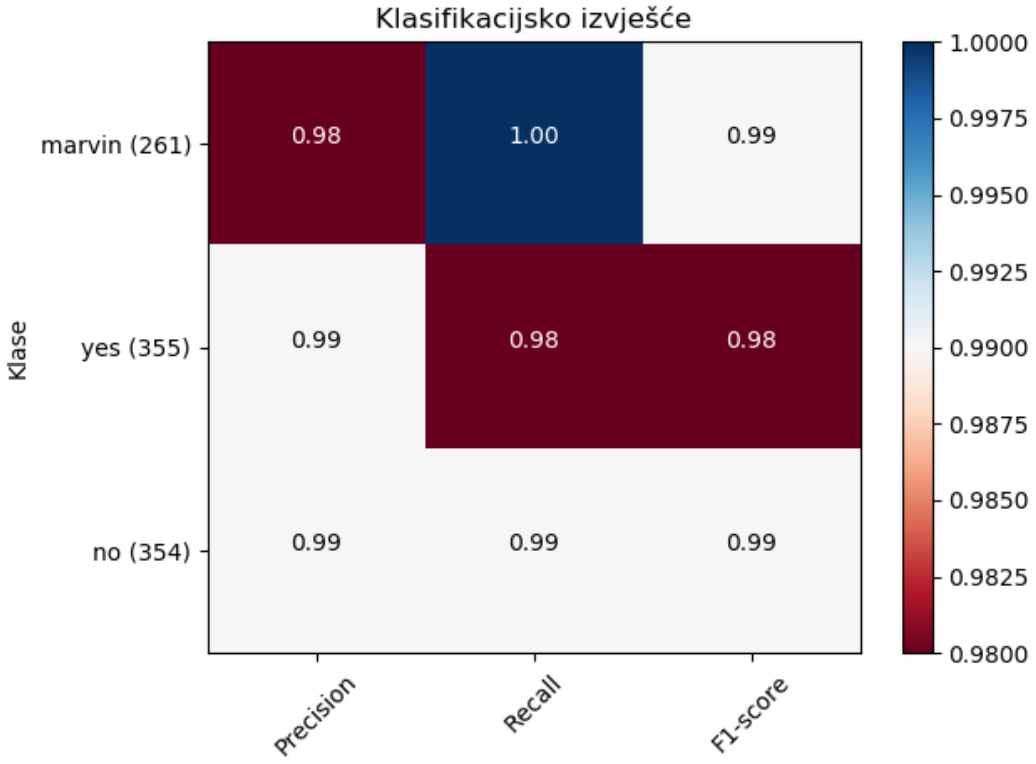
Slika 20. Graf funkcije greške modela

Na slici 21 je prikazan odnos preciznosti klasifikacije podataka na kojima se vršilo treniranje u usporedbi sa podacima na kojima se vršilo testiranje modela.



Slika 21. Graf preciznosti modela

Klasifikacijsko izvješće (engl. Classification report), na slici 22, prikazuje mjere preciznosti, sposobnosti pronalaženja pozitivnih primjera za određenu klasu (*engl. Recall*), F1 – rezultata i broja primjera za svaku klasu (*engl. Support*), primijenjeno na modelu korištenom u ovom radu. Ovo daje bolju intuiciju funkcioniranja klasifikatora u usporedbi sa samom preciznošću, koja može maskirati funkcionalne slabosti u određenoj klasi pri višeklasnoj klasifikaciji.



Slika 22. Klasifikacijsko izvješće modela

U klasifikacijskom izvješću mjere su definirane kao pravi i lažni pozitivni primjeri, te pravi i lažni negativni primjeri. Pozitivni i negativni primjeri su u ovom slučaju imena za točno i krivo klasificirane primjere za određenu klasu. Za jednostavno objašnjenje navedenih mjera, stvara se privremena binarna klasifikacija između jedne klase i ostalih klasa, odnosno za točno predviđanje za određenu klasu se koristi oznaka 1, a za krivo predviđanje za istu određenu klasu se koristi oznaka 0. Pravi pozitivni primjer je kada je za klasu za koju se vrši predviđanje, stvarna oznaka 1 i predviđena oznaka 1. Lažni pozitivni primjer je kada je stvarna oznaka 0, a predviđena oznaka 1. Lažni negativni primjer je kada je stvarna oznaka 1, a predviđena 0, te pravi negativni primjer kada je stvarna oznaka 0, i predviđena oznaka 0. Na slici 23 je jednostavan prikaz tih mjera.

		Actual	
		Positive	Negative
Predicted	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

Slika 23. Mjere za dobivanje preciznosti i pronalaženje pozitivnih primjera za određene klase [29]

Koristeći navedene pojmove, mjere klasifikacijskog izvješća se definiraju slijedećim jednadžbama:

- Preciznost:

$$precision = \frac{true\ positives}{true\ positives\ +\ false\ positives}$$

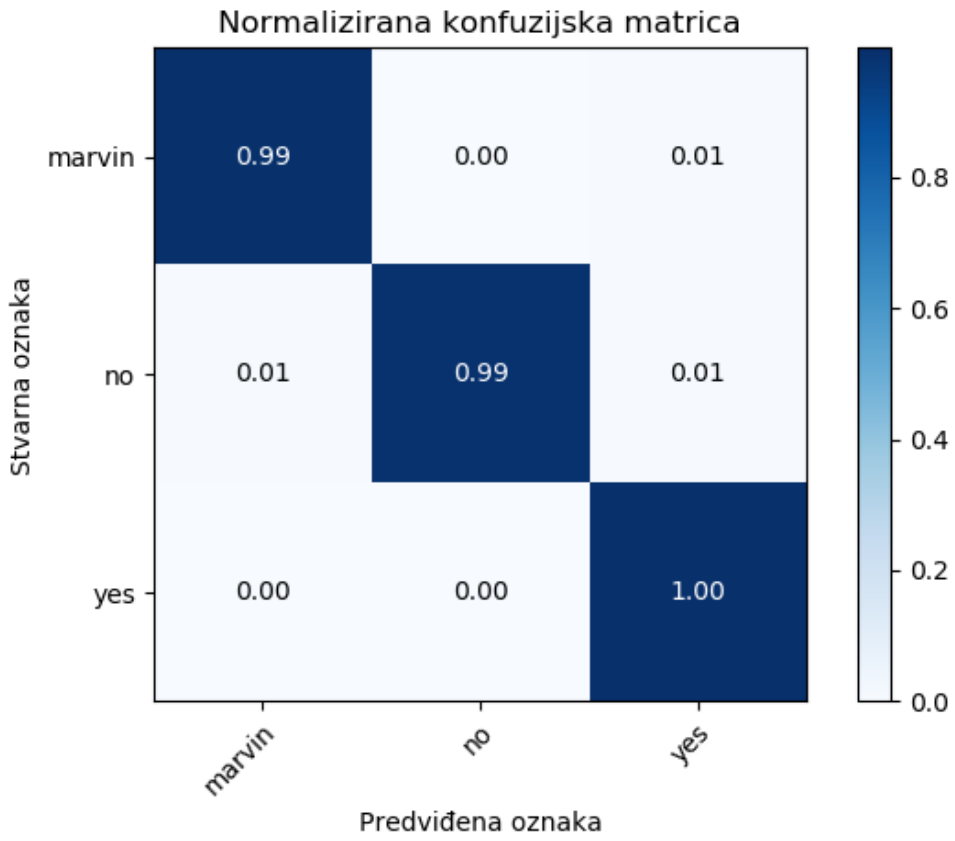
- Sposobnost pronalaženja pozitivnih primjera za određenu klasu (*engl. Recall*):

$$recall = \frac{true\ positives}{true\ positives + false\ negatives}$$

- F1 – rezultat:

$$F1 = 2 * \frac{precision * recall}{precision + recall}$$

Još jedan jednostavniji prikaz informacija o stvarnoj i predviđenoj klasifikaciji modela je konfuzijska matrica za model po kojem se vršila klasifikacija u ovom radu, prikazana na slici 24.



Slika 24. Konfuzijska matrica

5. ZAKLJUČAK

Prepoznavanje govora, točnije izgovorenih riječi, se u ovom radu vršilo pomoću modela neuronskih mreža sa dugoročnim pamćenjem (LSTM), koje su se pokazale kao jedan od najučinkovitijih alata za prepoznavanje sekvencijalnih podataka. Takva neuronska mreža spada u povratne neuronske mreže, koje za razliku od tradicionalnih neuronskih mreža imaju sposobnost pamćenja čime im je olakšana obrada i prepoznavanje novih informacija. Prednost LSTM mreže u odnosu na obične povratne neuronske mreže (RNN) je njihova kompleksna struktura koja im omogućuje dugoročno pamćenje korisnih informacija u usporedbi sa RNN mrežom.

Zvučni signali korišteni u radu su tisuće primjeraka izgovorenih riječi „Marvin“, „Yes“ i „No“. Izvorni podaci koji su preuzeti sa *Tensorflow-a*, sadržavali su 6459 podataka.

Proces treniranja neuronske mreže kao najvažniji korak sadrži izdvajanje svojstava iz izvornih podataka kao mjeru smanjenja količine podataka uz istovremeno održavanje ključnih informacija o izvornim podacima. U ovom radu su izvorni podaci zvučni signal, iz kojeg se izvlačilo svojstva stope prelaska nule, kepstralni koeficijenti po Mel-skali, spektralni centroid i spektralno opadanje.

Testirajući program na načine da se mijenja dizajn LSTM mreže, povećava i smanjuje broj svojstava i novih podataka, u ovom radu je najbolja preciznost dostignuta po postavkama programskog koda priloženog u dodatku pod brojem 1. Dodatna bitna činjenica koja je dovela do značajnog poboljšanja preciznosti je nasumičan raspored podataka. U poglavlju 4 su objašnjene mjere preciznosti po kojima se promatrao model, te na slici 25 je vidljiv iznos preciznosti sačuvanih težina za najmanji nađeni iznos greške.

```
Epoch 59/70
5408/5489 [============================>.] - ETA: 0s - loss: 0.0600 - accuracy: 0.9793
Epoch 00059: val_loss improved from 0.02588 to 0.02433, saving model to D:/Machine learning Python/Programi/SR/weights-best.hdf5
5489/5489 [==============================] - 4s 718us/sample - loss: 0.0595 - accuracy: 0.9794 - val_loss: 0.0243 - val_accuracy: 0.9928
```

Slika 25. Najbolje težine po mjerilu najmanje greške

LITERATURA

[1] Richard Shiavi, *Introduction to Applied Statistical Signal Analysis*, Third Edition Guide to Biomedical and Electrical Engineering Applications, 2007

[2] https://klima.hr/st_sred_30.png

[3] https://www.researchgate.net/figure/Artificial-sources-a-EEG-b-line-noise-c-eye-blink-artifact-and-d-heartbeat_fig2_264935611

[4] https://en.wikipedia.org/wiki/Signal

[5] http://ataspinar.com/2018/12/21/a-guide-for-using-the-wavelet-transform-in-machine-learning/

[6] Preeti Rao, *Audio Signal Processing*, Department of Electrical Engineering, Indian Institute of Technology Bombay, India, 2007.

[7] Mittal C. Darji, *Audio Signal Processing: A Review of Audio Signal Classification Features*, International Journal of Scientific Research in Computer Science, Engineering and Information Technology, 2017.

[8] Peter Cheung, *Lecture 13 - Sampling & Discrete signals*, *Signals & Linear Systems*, Department of Electrical & Electronic Engineering, Imperial College London, 2011.

[9] https://www.tutorialspoint.com/signals_and_systems/signals_sampling_theorem.htm

[10] Theodoros Giannakopoulos, Aggelos Pikrakis, *Introduction to Audio Analysis: A MATLAB Approach*, First edition 2014

[11] Biing-Hwang Juang, Sadaoki Furui, *"Automatic Recognition and Understanding of Spoken Language – A first Step Toward Natural Human – Machine Communication"*, Proceedings of the IEEE, Vol. 88, NO. 8, August 2000.

[12] K. H. Davis, R. Biddulph, and S. Balashek, *"Automatic recognition of spoken digits," J. Acoust. Soc. Amer.*, vol. 24, no. 6, pp. 637–642, 1952.

[13] H. F. Olson and H. Belar, *"Phonetic typewriter," J. Acoust. Soc. Amer.*, vol. 28, no. 6, pp. 1072–1081, 1956.

[14] J.W. Forgie and C. D. Forgie, *"Results obtained from a vowel recognition computer program," J. Acoust. Soc. Amer.*, vol. 31, no. 11, pp. 1480–1489, 1959.

[15] Roman A. Solovyev, Maxim Vakhrushev, Alexander Radionov, Vladimir Aliev, Alexey A. Shvets, *Deep Learning Approaches for Understanding Simple Speech Commands*, arXiv:1810.02364v1 [cs.SD] 4 Oct 2018

[16] https://www.wavemetrics.com/products/igorpro/dataanalysis/signalprocessing/spectralwindowing

[17] http://download.tensorflow.org/data/speech_commands_v0.01.tar.gz

[18] Radek Martinek, Radana Kahankova, Homer Nazeran, Jaromir Konecny, Janusz Jezewski, Petr Janku, Petr Bilik, Jan Zidek, Jan Nedoma and Marcel Fajkus, *„Non-Invasive Fetal Monitoring: A Maternal Surface ECG Electrode Placement-Based Novel Approach for Optimization of Adaptive Filter Control Parameters Using the LMS and RLS Algorithms“,* May 2017, https://www.mdpi.com/sensors/sensors-17-01154/article_deploy/html/images/sensors-17-01154-g007.png

[19] http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[20] Sepp Hochreiter, Jurgen Schmidhuber, *„Long Short-Term Memory“*, Neural Computation 9(8): 1735-1780, 1997

[21] Alex Graves, *„Supervised Sequence Labelling with Recurrent Neural Networks“*, February 2012

[22] http://ronny.rest/blog/post_2017_08_16_tanh/

[23] Wang Wenbo Li Sichun Yang Jianshe, Zhou Weicun, Liu Zhao, *„Feature Extraction of Underwater Target in Auditory Sensation Area Based on MFCC“*, China Ocean Acoustics Symposium, Harbin, China, IEEE/ OES 2016

[24] Klaus Greff, Rupesh K. Srivastava, Jan Koutnik, Bas R. Steunebrink, Jurgen Schmidhuber, *„LSTM: A Search Space Odyssey“*, Transactions on neural networks and learning systems, arXiv:1503.04069v2 [cs.NE] 4 Oct 2017

[25] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov, *„Dropout: A Simple Way to Prevent Neural Networks from Overfitting“*, Journal of Machine Learning Research 15 (2014), Department of Computer Science, University of Toronto, Canada

[26] http://wiki.fast.ai/index.php/Log_Loss

[27] https://ml-cheatsheet.readthedocs.io/en/latest/optimizers.html#adam

[28] Ian Goodfellow, Yoshua Bengio, Aaron Courville, *„Deep Learning“*, MIT Press Book, 2016. - http://www.deeplearningbook.org/

[29] https://towardsdatascience.com/beyond-accuracy-precision-and-recall-3da06bea9f6c (13. rujna 2019.)

[30] Andrew Ng, *„Machine learning yearning“*, Draft Version, 2018

DODATAK

1. SR-LSTM.py - Program za klasifikaciju riječi

```
import os
import json
import numpy as np
import random
import librosa
import pandas as pd
from scipy.io import wavfile
import scipy.signal
import matplotlib.pyplot as plt
from scipy import signal
from random import shuffle
from math import floor
from tensorflow.keras import layers
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.models import Sequential
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.utils.multiclass import unique_labels
from tqdm import tqdm
import itertools

path = 'D:/Machine learning Python/Programi/SpeechCommands/'

dirs = os.listdir(path)
train_df = pd.read_csv('D:/Machine learning Python/Programi/SR/train_labels.csv')
val_df = pd.read_csv('D:/Machine learning Python/Programi/SR/val_labels.csv')
train_df = train_df[:5489]
val_df = val_df[:970]

train_df = train_df.sample(frac=1)
val_df = val_df.sample(frac=1)

train_list = train_df['Filename'].tolist()
validation_list = val_df['Filename'].tolist()

train_labels = train_df['Label'].tolist()
validation_labels = val_df['Label'].tolist()


sample_rate = 16000

def load_and_process(audio_path):
```

```
        sr, audio_data = wavfile.read(audio_path)
        audio_signal = signal.resample(audio_data, sample_rate)
        audio_signal = audio_signal/np.amax(audio_signal)
        #print(len(audio_signal))
        zcr_signal = librosa.feature.zero_crossing_rate(audio_signal, frame_length=512,
hop_length=410) #(1,32)
        mfcc_signal = librosa.feature.mfcc(audio_signal,sample_rate, n_fft=1024,
hop_length=410) # (20,32)
        centr_signal = librosa.feature.spectral_centroid(audio_signal, sample_rate,
n_fft=1024, hop_length=410)# (1,32)
        rolloff_signal = librosa.feature.spectral_rolloff(audio_signal,n_fft=1024,
hop_length=410) # (1,32)

        features_mat = np.concatenate((zcr_signal, mfcc_signal, centr_signal,
rolloff_signal), axis=0)
        return features_mat


train_data = np.zeros((len(train_list), 23, 40))
validation_data = np.zeros((len(validation_list), 23, 40))

for i in tqdm(range(0,len(train_list))):
        train_data[i,:,:] = load_and_process(path + train_list[i])
        #print(str(i) + ' out of ' + str(len(train_list)) + ' completed.')

for i in tqdm(range(0,len(validation_list))):
        validation_data[i,:,:] = load_and_process(path + validation_list[i])
        #print(str(i) + ' out of ' + str(len(validation_list)) + ' completed.')


def build_model():
        model = Sequential()
        model.add(layers.LSTM(64,input_shape=(train_data.shape[1:]),
return_sequences=True))
        model.add(layers.Dropout(0.4))
        model.add(layers.LSTM(128))
        model.add(layers.Dropout(0.4))
        model.add(layers.Dense(32, activation='relu'))
        model.add(layers.Dropout(0.2))
        model.add(layers.Dense(3, activation='softmax'))
        #model.add(layers.Dropout(0.2))

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

```
    )

        return model

model = build_model()
model.summary()

checkpoint_path = "D:/Machine learning Python/Programi/SR/weights-best.hdf5"
cp_callback = ModelCheckpoint(checkpoint_path,
                              monitor ='val_loss',
                              verbose = 1,
                        save_best_only=True,
                        #error
                        mode = 'min')

callbacks_list = [cp_callback]
history = model.fit(train_data, train_labels,
                                          validation_data=(validation_data, validation_labels),
                                          epochs=70,
                                          callbacks = callbacks_list,
                                          )

model.load_weights('D:/Machine learning Python/Programi/SR/weights-best.hdf5')

print(train_data.shape)
print(validation_data.shape)

history_dict = history.history
print(history_dict.keys())
plt.figure(1)
plt.plot(history_dict['accuracy'])
plt.plot(history_dict['val_accuracy'])
plt.title('Preciznost modela')
plt.ylabel('Preciznost')
plt.xlabel('Epoha')
plt.legend(['Podaci za treniranje', 'Podaci za testiranje'], loc='upper left')
plt.grid(True)
#plt.show()

# Plot training & validation loss values
plt.figure(2)
plt.plot(history_dict['loss'])
plt.plot(history_dict['val_loss'])
plt.title('Gre¨ka modela')
plt.ylabel('Gre¨ka')
plt.xlabel('Epoha')
plt.legend(['Podaci za treniranje', 'Podaci za testiranje'], loc='upper left')
```

```
plt.grid(True)
plt.show()

val_predict = model.predict(validation_data)

print(type(val_predict))
print(type(validation_labels))
validation_labels = np.asarray(validation_labels)
val_predict = val_predict.argmax(axis=1)
print(val_predict)
print(confusion_matrix(validation_labels,val_predict, labels = [0,1,2]))
target_names = ['marvin','yes','no']
print(classification_report(validation_labels,val_predict, target_names=target_names))



# from sklearn.metrics import classification_report
# print(classification_report(val_predict, validation_labels))
def plot_confusion_matrix(y_true, y_pred,
                          normalize=False,
                          title=None,
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    if not title:
        if normalize:
            title = 'Normalizirana konfuzijska matrica'
        else:
            title = 'Confusion matrix, without normalization'

    # Compute confusion matrix
    cm = confusion_matrix(y_true, y_pred)
    # Only use the labels that appear in the data
    classes = ['marvin','no','yes']
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalizirana konfuzijska matrica")
    else:
        print('Confusion matrix, without normalization')

    print(cm)

    fig, ax = plt.subplots()
    im = ax.imshow(cm, interpolation='nearest', cmap=cmap)
    ax.figure.colorbar(im, ax=ax)
```

```
    # We want to show all ticks...
    ax.set(xticks=np.arange(cm.shape[1]),
           yticks=np.arange(cm.shape[0]),
           # ... and label them with the respective list entries
           xticklabels=classes, yticklabels=classes,
           title=title,
           ylabel='Stvarna oznaka',
           xlabel='Predviđena oznaka')

    # Rotate the tick labels and set their alignment.
    plt.setp(ax.get_xticklabels(), rotation=45, ha="right",
             rotation_mode="anchor")

    # Loop over data dimensions and create text annotations.
    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i in range(cm.shape[0]):
        for j in range(cm.shape[1]):
            ax.text(j, i, format(cm[i, j], fmt),
                    ha="center", va="center",
                    color="white" if cm[i, j] > thresh else "black")
    fig.tight_layout()
    return ax

np.set_printoptions(precision=2)

# Plot non-normalized confusion matrix
plot_confusion_matrix(validation_labels, val_predict, normalize=True,
                      title='Normalizirana konfuzijska matrica')
#classes=np.asarray([0,1,2])
plt.show()



target_names = ['marvin','yes','no']
classif_report = classification_report(validation_labels,val_predict,
target_names=target_names)
print(classif_report)
```

2. new_data2.py - stvaranje novih podataka sa dodanom bukom

```
import os
import numpy as np
import random
from numpy import asarray
import pandas as pd
import scipy
```

```
from scipy.io import wavfile
from scipy import signal
from tqdm import tqdm
from math import floor

path_marv = 'D:/Machine learning Python/Programi/SpeechCommands/marvin/'

path_no = 'D:/Machine learning Python/Programi/SpeechCommands/no/'

path_yes = 'D:/Machine learning Python/Programi/SpeechCommands/yes/'

marv_files = os.listdir(path_marv)
marv_list = [path_marv + x for x in marv_files if not x.startswith('audio')]
no_files = os.listdir(path_no)
no_list = [path_no + x for x in no_files if not x.startswith('audio')]
yes_files = os.listdir(path_yes)
yes_list = [path_yes + x for x in yes_files if not x.startswith('audio')]

train_list = sum([marv_list[0:floor(0.85*len(marv_list))],
        no_list[0:floor(0.85*len(no_list))],
        yes_list[0:floor(0.85*len(yes_list))]],[])

val_list = sum([marv_list[floor(0.85*len(marv_list)):],
        no_list[floor(0.85*len(no_list)):],
        yes_list[floor(0.85*len(yes_list)):]],[])


# Učitavanje buke
path_street_noise = 'D:/Machine learning Python/Programi/SR/street_noise.wav'
sr_s,street_noise = wavfile.read(path_street_noise)
m = np.amax(street_noise)

# Funkcija za dodavanje buke
def new_audio_data(path, sample_rate):
        """ Function for adding noise of 1 sec to audio file of 1 sec"""
        sr, audio_signal = wavfile.read(path)
        audio_signal = signal.resample(audio_signal, sample_rate)

        r=random.randint(2,3)
        k = (np.amax(audio_signal)/m)/r
        new_street_noise = k * street_noise

        new_audio_signal = audio_signal + new_street_noise
        audio_new = np.asarray(new_audio_signal, dtype=np.int16)
        return audio_new
```

```
# Petlje za kreiranje novih podataka za treniranje i testiranje sa dodanom bukom
new_train_wav=[]
for i in tqdm(range(0,len(train_list))):
        audio_new = new_audio_data(train_list[i],16000)
        file_name = "D:/Machine learning
Python/Programi/SpeechCommands/zaudio_train/audio_train_street{}.wav".format(i)
        scipy.io.wavfile.write(file_name,16000, audio_new)
        new_train_wav.append('audio_train_street{}.wav'.format(i))


new_val_wav=[]
for i in tqdm(range(0,len(val_list))):
        audio_new = new_audio_data(val_list[i],16000)
        file_name = "D:/Machine learning
Python/Programi/SpeechCommands/zaudio_test/audio_val_street{}.wav".format(i)
        scipy.io.wavfile.write(file_name,16000, audio_new)
        new_val_wav.append('audio_val_street{}.wav'.format(i))

dirs = os.listdir('D:/Machine learning Python/Programi/SpeechCommands/')

train_df = pd.read_csv('D:/Machine learning Python/Programi/SR/train_labels.csv')
val_df = pd.read_csv('D:/Machine learning Python/Programi/SR/val_labels.csv')

train_labels = train_df["Label"].tolist()
train_labels = train_labels[0:5489]

val_labels = val_df["Label"].tolist()
val_labels = val_labels[0:970]

new_train_list = [s.replace('D:/Machine learning Python/Programi/SpeechCommands/','')
for s in train_list]
new_val_list = [s.replace('D:/Machine learning Python/Programi/SpeechCommands/','')
for s in val_list]


new_train_dirs = [d + '/' for train in new_train_list for d in dirs if train.startswith(d)]
print(len(new_train_dirs))
new_train_wavs = [x + y for x,y in zip(new_train_dirs,new_train_wav)]
print(len(new_train_wavs))

new_val_dirs = [d + '/' for val in new_val_list for d in dirs if val.startswith(d)]
print(len(new_val_dirs))
new_val_wavs = [x + y for x,y in zip(new_val_dirs,new_val_wav)]
print(len(new_val_wavs))

# Dataframe creation and addition
```

```
new_train_df = pd.DataFrame({"Filename":new_train_wavs,"Label": train_labels})
new_val_df = pd.DataFrame({"Filename":new_val_wavs,"Label": val_labels})

updated_train_df = train_df.append(new_train_df, ignore_index=True)
updated_val_df = val_df.append(new_val_df,ignore_index=True)

updated_train_df.to_csv("D:/Machine learning Python/Programi/SR/train_labels.csv",
sep=',',index=False)
updated_val_df.to_csv("D:/Machine learning Python/Programi/SR/val_labels.csv",
sep=',',index=False)
# import sounddevice as sd
# scipy.io.wavfile.write('D:/Machine learning
Python/Programi/SR/new_marv.wav',16000,new_marv)
# myrec=sd.rec(int(1*16000),samplerate=16000, channels=1)
```

3. CR_plot.py - klasifikacijsko izvješće

```
import matplotlib.pyplot as plt
import numpy as np
import itertools


def plot_classification_report(classificationReport,
                               title='Klasifikacijsko izvješće',
                               cmap='RdBu'):

    classificationReport = classificationReport.replace('\n\n', '\n')
    classificationReport = classificationReport.replace(' / ', '/')
    lines = classificationReport.split('\n')

    classes, plotMat, support, class_names = [], [], [], []
    for line in lines[1:]:  # if you don't want avg/total result, then change [1:] into [1:-1]
        t = line.strip().split()
        if len(t) < 2:
            continue
        classes.append(t[0])
        v = [float(x) for x in t[1: len(t) - 1]]
        support.append(int(t[-1]))
        class_names.append(t[0])
        plotMat.append(v)

    plotMat = np.array(plotMat)
    xticklabels = ['Precision', 'Recall', 'F1-score']
    yticklabels = ['{0} ({1})'.format(class_names[idx], sup)
                   for idx, sup in enumerate(support)]
```

```
    plt.imshow(plotMat, interpolation='nearest', cmap=cmap, aspect='auto')
    plt.title(title)
    plt.colorbar()
    plt.xticks(np.arange(3), xticklabels, rotation=45)
    plt.yticks(np.arange(len(classes)), yticklabels)

    upper_thresh = plotMat.min() + (plotMat.max() - plotMat.min()) / 10 * 8
    lower_thresh = plotMat.min() + (plotMat.max() - plotMat.min()) / 10 * 2
    for i, j in itertools.product(range(plotMat.shape[0]), range(plotMat.shape[1])):
        plt.text(j, i, format(plotMat[i, j], '.2f'),
                 horizontalalignment="center",
                      color="white" if (plotMat[i, j] > upper_thresh or plotMat[i, j] <
             lower_thresh) else "black")

    plt.ylabel('Klase')
    #plt.xlabel('Metrics')
    plt.tight_layout()


def main():

    sampleClassificationReport = """precision    recall  f1-score   support

       marvin      0.97     0.96     0.96     1827
          yes      0.92     0.94     0.93     1775
           no      0.96     0.95     0.95     1770
    """

    plot_classification_report(sampleClassificationReport)
    plt.show()
    plt.close()

if __name__ == '__main__':
    main()
```