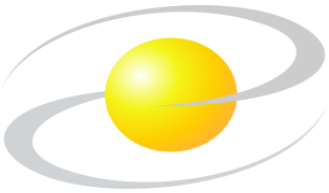


UNIVERZITET CRNE GORE

ELEKTROTEHNIČKI FAKULTET

Budimir Anđelić

RAZVOJ I IMPLEMENTACIJA SISTEMA ZA DETEKCIJU SJEČE ŠUMA ZASNOVANOG NA DUBOKOM UČENJU

MAGISTARSKI RAD

Podgorica, 2023. godine

UNIVERZITET CRNE GORE

ELEKTROTEHNIČKI FAKULTET

Budimir Anđelić

RAZVOJ I IMPLEMENTACIJA SISTEMA ZA DETEKCIJU SJEČE ŠUMA ZASNOVANOG NA DUBOKOM UČENJU

MAGISTARSKI RAD

Podgorica, 2023. godine

PODACI I INFORMACIJE O MAGISTRANDU

Ime i prezime: **Budimir Anđelić**

Datum i mjesto rođenja: 14.02.1998. Pljevlja, Crna Gora.

Prethodno završene studije:

Osnovne studije (Elektrotehnički fakultet Podgorica, Elektronika, Telekomunikacije i Računari, 180 ECTS kredita, 2019)

Specijalističke studije (Elektrotehnički fakultet Podgorica, Elektronika, 60 ECTS kredita, 2020)

INFORMACIJE O MAGISTARSKOM RADU

Elektrotehnički fakultet

Studijski program: Akademske magistarske studije, studijski program: Elektronika,

Telekomunikacije i Računari

Naslov rada: **Razvoj i implementacija sistema za detekciju sječe šuma zasnovanog na dubokom učenju**

UDK, OCJENA I ODBRANA MAGISTARSKOG RADA

Datum prijave magistarskog rada: 05.12.2022.

Datum sjednice Vijeća na kojoj je prihvaćena tema: 23.03.2023.

Komisija za ocjenu teme i podobnosti magistranda:

1. Prof. dr Božo Krstajić
2. Prof. dr Milutin Radonjić
3. Prof. dr Slobodan Đukanović

Mentor: Prof. dr Milutin Radonjić

Komisija za ocijenu rada:

1. Prof. dr Božo Krstajić
2. Prof. dr Milutin Radonjić
3. Prof. dr Slobodan Đukanović

Komisija za odbranu rada:

1. Prof. dr Božo Krstajić
2. Prof. dr Milutin Radonjić
3. Prof. dr Slobodan Đukanović

Lektor:

Datum odbrane: _____

Datum promocije: _____

Ime i prezime autora: Budimir Anđelić, Spec. Sci

ETIČKA IZJAVA

U skladu sa članom 22 Zakona o akademskom integritetu i članom 24 Pravila studiranja na postdiplomskim studijama, pod krivičnom i materijalnom odgovornošću, izjavljujem da je magistarski rad pod naslovom

"Razvoj i implementacija sistema za detekciju sječe šuma zasnovanog na dubokom učenju"

moje originalno djelo.

Podnosilac izjave,

Budimir Anđelić, Spec. Sci

U Podgorici, dana 10.04.2023. godine

PREDGOVOR

U ovom magistarskom radu je predočen potencijal razvoja i implementacije sistema za detekciju sječe šuma zasnovanog na dubokom učenju.

Moja iskrena zahvalnost mentoru, prof. dr Milutinu Radonjiću koji je prihvatio mentorstvo i dao korisna uputstva i savjete u toku studija. Neizmjerno cijenim slobodu koju sam imao i njegovo povjerenje u mene.

Zahvalnost dugujem i prof. dr Slobodanu Đukanoviću za izdvojeno vrijeme, komentare i sugestije.

Na kraju, posebnu zahvalnost dugujem ocu Mihailu, majci Marini i sestrama Miljani i Milenki na neizmjernoj podršci i ljubavi koju su mi pružali tokom cijelog trajanja studija. Ovaj magistarski rad posvećujem njima.

SAŽETAK

Bespravna sječa šuma predstavlja veliki ekološki problem koji narušava stabilnost šumskog ekosistema i pospješuje klimatske promjene, poplave, eroziju zemljišta, zakorovljavanje staništa, izumiranje životinjskih i biljnih vrsta. Ovaj rad predlaže metod za ublažavanje problema bespravne sječe šuma automatskim otkrivanjem zvuka sječe drveća. Tačnije, predložena je detekcija zvuka motorne testere koristeći duboko učenje.

Razmatrana su dva pristupa dubokog učenja, jedan zasnovan na potpuno povezanoj neuralnoj mreži i drugi zasnovan na konvolucionoj neuralnoj mreži (eng. *Convolutional Neural Network* - CNN). Demonstrirana je sposobnost neuralnih mreža da detektuju zvukove motorne testere, zvukove okoline i zvukove koji su slični motornoj testeri, preko njihovih vremenskih, frekvencijskih i vremensko-frekvencijskih reprezentacija.

U svrhu ovog istraživanja prikupljena su dva skupa podataka audio signala. Prvi skup podataka, preuzet sa YouTube-a, koristi se za trening i validaciju predloženih modela. Drugi skup podataka, koji je snimljen za potrebe ovog istraživanja, u realnom okruženju, koristi se za testiranje predloženih modela. Eksperimenti su pokazali da pristup zasnovan na CNN-u nadmašuje pristup zasnovan na potpuno povezanoj neuralnoj mreži, sa preciznošću zvučne klasifikacije od 94.96% na prvom skupu podataka i 88.87% na drugom skupu podataka.

Predloženi sistem za detekciju sječe šuma je implementiran na mikroprocesorskoj hardverskoj platformi opremljenoj sa mikrofonom i sa GPS/GSM/GPRS modulom. Ukoliko dođe do detekcije zvuka motorne testere, sistem obavještava korisnika o događaju tako što šalje informacije na *cloud.* Sistem se može u obliku senzorskih čvorova instalirati na stablima u šumi, dok bi se napajanje vršilo preko solarnih panela. Na ovaj način se može doprinijeti smanjenju bespravne sječe šuma te tako smanjiti dugoročne posljedice po okolinu, kvalitet života ljudi i ekonomiju.

Ključne riječi: sječa šuma, motorna testera, duboko učenje, audio karakteristike, Raspberry Pi

ABSTRACT

Illegal logging is a major environmental issue, which impedes the stability of forest ecosystem and supports climate change, flooding, soil erosion, weeding of habitats, extinction of animal and plant species. This paper proposes a method for mitigating the logging issue by automatically detecting the sound of logging activities. More specifically, it is proposed to detect the chainsaw sound using deep learning.

Two deep learning approaches were considered, one based on fully connected neural network and the other based on convolutional neural network (CNN). The ability of neural networks to detect chainsaw sounds, environmental sounds, and sounds similar to the chainsaw, through their time, frequency, and time-frequency representations is demonstrated.

For the purpose of this research, two datasets of audio signals were collected. First dataset, downloaded from YouTube, is used for training and validating the proposed models. Second dataset, which is recorded for the purpose of this research, in a real environment, is used for testing of the proposed models. The experiments have shown that the CNN-based approach outperforms the fully connected neural network-based one, with a sound classification accuracy of 94.96% on the first dataset and 88.87% on the second dataset.

The proposed logging detection system is implemented on a microprocessor's hardware platform equipped with a microphone and a GPS/GSM/GPRS module. If the sound of a chainsaw is detected, the system notifies the user on the event by sending information to the cloud. The system can be implemented in the form of sensor nodes on trees in the forest, while the power supply could be provided via solar panels. In this way, it is possible to contribute to the reduction of illegal logging and thus reduce the long-term consequences for the environment, people's quality of life and the economy.

Key words: logging, chainsaw, deep learning, audio features, Raspberry Pi

Sadržaj

1 UVOD

Šume predstavljaju jedan od najvažnijih prirodnih resursa, koji uključuje sve koristi koje proizilaze iz šumskog zemljišta [1]. Pored primarnog zadatka proizvodnje drveta, šume obezbjeđuju zaštitu od štetnih gasova, proizvodnju kiseonika, zaštitu zdravlja ljudi, stanište za životinje, zaštitu zemljišta, zaštitu od vjetra i poplava, turističke resurse [2]. Šume održavaju ekosistem tako što apsorbuju štetne gasove poput ugljen-dioksida, amonijaka i sumpor-dioksida za koje se vjeruje da su razlog klimatskih promjena. Osim apsorpcije štetnih gasova, biljke, žbunje i drveće oslobađaju kiseonik i tako djeluju kao „džinovska pluća“ koja prečišćavaju vazduh u atmosferi. Naselja u blizini šuma odlikuju se većim kvalitetom vazduha.

Jedna od najvažnijih funkcija za zdravlje ljudi jeste rekreacijska funkcija šuma. Rekreacijske aktivnosti uključuju šetnju, planinarenje, planinski biciklizam, skijanje, posmatranje ptica, sakupljanje pečuraka, itd. Osim toga, šume su bogate ljekovitim biljem, biljkama i drvećem. Ekstrakti, sjeme, lišće i kora ovih biljaka i drveća liječe razne bolesti i sastavni su dio farmaceutskih proizvoda.

Šume obezbjeđuju održivo okruženje za opstanak velikog broja životinja, poput ptica, sisara i insekata. Šumsko tlo je bogat medijum za mikroorganizme, koji su neophodni u procesu mineralizacije organskih ostataka, što obogaćuje zemljište hranljivim materijama. Šume su najveći prirodni prečišćivači voda. One utiču na postepeno slivanje vode, sprečavaju eroziju i akumuliraju poplavne vode. Drveće ima veoma jake korijene koji održavaju zemljište netaknutim u slučajevima erozije, a šumsko tlo apsorbuje hranljive materije i sediment. Kod degradiranih šuma sediment se uliva u potoke i zagađuje vodu. Osim korijenovog sistema važnu ulogu imaju suvo lišće i grančice, koji mogu da zadrže veliku količinu vode, i stabla koja pružaju zaštitu od olujnih vjetrova. Takođe, „šume oplemenjuju i oblikuju prostore i daju im pejzažne i vizuelne efekte i vrijednosti“ [3], koji privlače mnogobrojne turiste, što ima značajan ekonomski doprinos.

Iako opstanak čovječanstva u velikoj mjeri zavisi od zdravlja šuma, njihovo uništavanje se nastavlja iz dana u dan. Pored šumskih požara, ozbiljnu prijetnju šumskim ekosistemima predstavlja i bespravna sječa šuma, što narušava stabilnost ekosistema, sa posljedicama koje se ogledaju u procesima zakorovljavanja staništa, sušenju stabala, polomljenim stablima, eroziji zemljišta, klimatskim promjenama, nestanku mnogih biljnih i životinjskih vrsta, itd. Pored dugoročnih posljedica po životnu sredinu i kvalitet života ljudi, sječa šuma donosi i velike ekonomske gubitke.

Prema [3], šume i šumsko zemljište čine 69.4% teritorije Crne Gore, što je jedan od najvećih procenata u Evropi. Visoke šume čine 51%, dok su izdanačke šume zastupljene sa 48% od ukupne površine pod šumama. Izdanačke šume su manje kvalitetne šume, koje su nastale iz izdanaka ili žila na panjevima prethodno posječenih stabala [4]. Veliki procenat izdanačkih šuma u Crnoj Gori je značajan pokazatelj bespravne sječe šuma u prošlosti. Poređenja radi, površina izdanačkih šuma u skoro svim zapadnim zemljama kreće se najviše do 5% [3].

Na osnovu izvještaja o zdravstvenom stanju šuma, koji izrađuje Ministarstvo poljoprivrede i ruralnog razvoja, u Crnoj Gori je 2015. godine bespravno posječeno 5257 m^3 drvne mase [5]. Tokom 2018. godine, u Crnoj Gori je bespravno posječeno 5867.98 m^3 drvne mase, od čega je zaplijenjeno svega 777.60 m^3 [6]. Prema [7], u Crnoj Gori je 2019. godine, od ukupno 6473.58 m^3 bespravno posječene drvne mase, zaplijenjeno oko 447.63 m^3, što je nešto manje od 7%. Naredne, 2020. godine, u šumama Crne Gore je bespravno posječeno 4160.94 m^3 drvne mase, od čega je 7.7% zaplijenjeno [8]. U 2021. godini od ukupno bespravno posječenih 5977.65 m^3 drvne mase, zaplijenjeno je svega 4.68% [9]. Potrebno je uzeti u obzir da su količine bespravno posječene drvne mase u realnosti mnogo veće, o čemu svjedoče terenski obilasci tokom kojim su evidentirane čiste sječe [1], koje nijesu markirane kao nelegalne aktivnosti. Ovo predstavlja direktnu štetu nastalu bespravnom sječom šuma. Pored direktne postoji i indirektna šteta, a ona može biti i do deset puta veća od direktne [3]. Negativnom uticaju na stanje šuma doprinosi i činjenica da se vrši bespravna sječa samo najkvalitetnijih stabala.

U Upravi za šume, koja je nadležna za gazdovanje šumama u Crnoj Gori, ima 422 zaposlena, od čega je svega 70 inženjera, 171 čuvar šuma i 56 šumarskih tehničara [3]. Prema ovim podacima svaki šumarski inženjer pokriva oko 140 km^2, a svaki šumar oko 56 km^2. Osim nedostatka radne snage, evidentan je i nedostatak adekvatne opreme i motornih vozila, kako bi se kontrolisala bespravna sječa šuma [10].

Tema ovog rada je razvoj i implementacija sistema za detekciju sječe šuma uz pomoć metoda dubokog učenja. Efikasan pristup za sprečavanje sječe je detekcija zvuka motorne testere, jednog od najčešće korišćenih alata za sječu drveća. Predloženi pristup problemu detekcije zvuka motorne testere zasniva se na ekstrahovanju vremenskih, frekvencijskih i vremensko-frekvencijskih karakteristika zvuka i primjeni metoda dubokog učenja nad dobijenim karakteristikama. Kao modeli za detekciju, u radu se koriste višeslojna potpuno povezana neuralna mreža i konvoluciona neuralna mreža. Predloženi model detekcije će se implementirati na mikroprocesorskoj platformi opremljenoj sa mikrofonom i GPS/GSM/GPRS modulom. Ovaj sistem bi se instalirao u šumi i napajao preko solarnog panela. Kada dođe do detekcije zvuka motorne testere, sistem obavještava korisnika o događaju i informacije skladišti na *cloud* platformi.

Rad je organizovan na sljedeći način. Poglavlje 2 pruža pregled dosadašnjih istraživanja na temu detekcije bespravne sječe šuma. U poglavlju 3 se stavlja akcenat na obradu zvučnog signala, kao i na karakteristike koje se koriste za opisivanje zvuka. Razmatrane su vremenske, frekvencijske i vremensko-frekvencijske karakteristike. Poglavlje 4 se fokusira na metode mašinskog učenja, tačnije na neuralne mreže. Opisana su dva tipa neuralnih mreža i to potpuno povezana neuralna mreža i konvoluciona neuralna mreža. Poglavlje 5 obuhvata opis implementacije predloženog sistema. U njemu je predstavljena primijenjena metodologija, skupovi podataka, dobijeni rezultati i data je diskusija rezultata. Završno poglavlje predstavlja zaključke istraživanja i daje pregled budućih pravaca istraživanja.

[1] Čista sječa je zahvat kojim se uklanjaju sva stabla na površini većoj od 20 ari, na kojoj nije predviđena promjena namjene zemljišta.

2 PREGLED DOSADAŠNJIH ISTRAŽIVANJA

Pojam mašinskog učenja (eng. *Machine Learning* – ML), uveden 1950. godine od strane Alan Turing-a [11], odnosi se na sisteme koji su sposobni da uče na osnovu priloženih podataka [12]. Postoji veliki broj ML algoritama, kao što su: logistička regresija, linearna regresija, *random forest*, metod k-najbližih susjeda (eng. *k-Nearest Neighbor* – k-NN), *Support Vector Machines* – SVM, neuralne mreže, itd. ML se snažno kandiduje kao efikasna paradigma za sprečavanje bespravne sječe šuma pomoću detekcije zvuka motorne testere, najčešće korišćenog alata za sječu drveća.

U [13] je predložen sistem za detekciju zvuka motorne testere razvijen u obliku senzorskih čvorova koji su dio bežične akustične senzorske mreže. Svaki zvuk je predstavljen preko skupa MFCC (eng. *Mel Frequency Cepstral Coefficients*) koeficijenata, a zatim se primjenjuje PDF (eng. *Probability Density Function*). Korišćena je metoda unarne klasifikacije koja omogućava prepoznavanje samo zvukova motorne testere, ne razmatrajući ostale zvukove. Obrada podataka se radi na „ivici mreže“ (eng. *edge computing*), a prenosi se samo informacija da li je detektovan zvuk motorne testere ili nije (1 ili 0). Eksperimentalnim putem je utvrđen broj MFCC koeficijenata kako bi se postigla što veća tačnost, a najbolji rezultati su postignuti sa 14 MFCC koeficijenata. Postignuta tačnost klasifikacije je 98%. Skup podataka sadrži 10835 uzoraka koji se odnose na zvukove okoline, motora i ljudskog govora, i 560 uzoraka koji se odnose na zvukove motorne testere. Evidentno je korišćenje neadekvatnog skupa podataka – manje od 5% audio signala se odnosi na motornu testeru.

Kao rješenje problema bespravne sječe drveća, autori u [14] su predložili tehniku zasnovanu na ekstrakciji 2D Haar talasnih koeficijenata iz spektrograma zvuka. Praćena su dva različita pristupa. Prvi pristup primjenjuje dva praga odlučivanja za izdvojene karakteristike, dok drugi vrši binarnu klasifikaciju koristeći SVM algoritam. Postignuta tačnost dostiže 97%. Međutim, postoje specifični zvukovi koji imaju veliku grešku u klasifikaciji. Zvukovi koje proizvode životinje kao što su krave i jeleni, neki insekti i ljudski glas imaju spektar sa visokim stepenom sličnosti sa motornom testerom. Zvukovi motorne testere u navedenom istraživanju čine 35% skupa podataka, dok zvukovi slični motornoj testeri čine svega 3% skupa podataka.

Studija [15] se fokusirala na mogućnost detekcije bespravne sječe šuma na osnovu prepoznavanja zvuka motorne testere, kombinovanjem MFCC sa k-NN i SVM algoritmima. Skup podataka sadrži 3265 audio signala od 5 sekundi, sa manje od 10% signala koji se odnose na zvuk motorne testere. Tačnost od 95.63% je postignuta sa SVM, a 94.02% sa k-NN algoritmom. Iako su tačnosti približne, SVM algoritam se pokazao kao mnogo bolja opcija zbog kraćeg vremena obrade. Za obradu zvuka bilo je potrebno 30 puta manje vremena. Međutim, tačnost drastično opada (53.16% za SVM, 40.50% za k-NN) ako postoje zvukovi slični motornoj testeri.

U [16] je predložen još jedan pristup zasnovan na prepoznavanju zvuka u cilju detekcije sječe šume. Analizom spektralnih karakteristika zvučnog signala, računanja Q indeksa sličnosti i odnosa signal/šum u predloženom pristupu, moguće je detektovati zvuk motorne testere. Skup podataka se sastojao od 17 kategorija audio signala. Zvukovi testere su snimljeni, dok su ostali zvukovi preuzeti sa javnih platformi. Kao prag odlučivanja odabrana je vrijednost Q indeksa sličnosti od 70%. Za veliki odnos signal/šum je postignuta velika tačnost, a kad se taj odnos smanjuje postiže se manja tačnost.

U [17] je predstavljen algoritam za detekciju zvuka motorne testere na osnovu senzorskog čvora sa ograničenom energijom. Ciljano je na nisku kompleksnost i upotrebu u realnim uslovima. Algoritam koristi tri tehnike: adaptivni energetski prag, delta detekciju koraka i odnos energetskog opsega. Skup podataka se sastojao od 250 audio signala. Eksperimenti pokazuju da je ovaj algoritam u mogućnosti da postigne tačnost od 90.8%. Da bi se smanjila greška, sistem će obavijestiti korisnika samo kad se dogodi određen broj detekcija.

U poslednjih nekoliko godina, metode dubokog učenja postaju glavno sredstvo za detekciju zvuka. Duboko učenje je dio ML-a, gdje se naglasak stavlja na učenje uzastopnih slojeva sve smislenijih reprezentacija podataka.

U studiji [18] predloženo je inovativno rješenje zasnovano na akustičnom nadzoru i tehnologijama mašinskog učenja kako bi se očuvale šume. Iako su se CNN modeli pokazali kao jako efikasni kad su u pitanju javne baze podataka za istraživanje, autori tvrde da se ne mogu primijeniti direktno zbog velike razlike između javnih baza i ciljanih zvukova u šumama. Dva metoda koja su korišćena su VGGish (eng. *Visual Geometry Group*) model i FCN (eng. *Fully Convolutional Network*). Autori su postigli obećavajuće rezultate (90.1% tačnost) sa dvije baze podataka. Jedna je javna, a druga je snimljena u šumi u realnim uslovima. Skup podataka snimljen u šumi se sastojao od 22000 audio signala trajanja 1 sekundu.

Rad [19] predlaže alternativnu šemu za ekstrakciju karakteristika zvuka motorne testere baziranu na MFCC i sinusoidalnom lifteru. Iako je pravi izbor kada je u pitanju klasifikacija audio signala, MFCC ima lošiji učinak u prisustvu šuma. U [19], mel koeficijenti su propušteni kroz sinusoidalni lifter kako bi se riješio problem šuma. Eksperimenti pokazuju da lifter povećava kapacitet učenja neuralne mreže u poređenju sa MFCC bez liftera. Korišćen je skup podataka ESC-50 [20]. Sastoji se od 50 klasa, a svaka klasa sadrži 40 audio snimaka u trajanju od 5 sekundi. Postignuta preciznost korišćenjem MFCC bez liftera je 36.37%, a sa lifterom 56.67%.

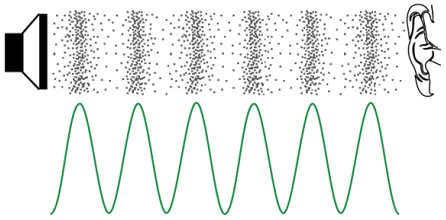
Mali broj radova na temu detekcije sječe šuma navodi hardversku implementaciju, pa se može zaključiti da se postignuti rezultati odnose na laboratorijske uslove. Takođe, u ovim radovima često nijesu korišćeni adekvatni skupovi podataka za treniranje i testiranje predloženih metoda. Klase nisu jednake veličine, pa metode prilikom treniranja smatraju da je jedna klasa bitnija od druge. Osim toga, autori često ne navode nad kojim podacima je vršeno testiranje, kao i da li su podaci za trening i testiranje ML algoritama iz različitih izvora. Samim tim, postignuta tačnost je upitna. Metode koje su do sad korišćene na ovu temu su uglavnom tradicionalni ML algoritmi. U ovom istraživanju, predlaže se upotreba dubokog učenja za detekciju sječe šuma, što je trenutno aktuelni pravac kad je u pitanju upotreba vještačke inteligencije. Pored rezultata u laboratorijskim uslovima, ovaj rad će predložiti hardversku implementaciju i predstaviti rezultate sistema u realnim uslovima.

3 OBRADA ZVUČNOG SIGNALA

Vještačka inteligencija (eng. *Artificial Inteligence* - AI) se može definisati kao sposobnost računara da obavlja zadatke koji se obično povezuju sa inteligentnim bićima, pri čemu se za inteligentna bića misli na one koji se mogu prilagoditi različitim situacijama [21]. Audio AI podrazumijeva primjenu AI tehnika i algoritama za tumačenje zvukova u okruženju. Može se podijeliti na dvije oblasti: priprema neobrađenih zvučnih signala i razvoj i treniranje modela. U ovom poglavlju akcenat će biti stavljen na obradu zvučnog signala, kao i na karakteristike koje se koriste za opisivanje zvuka. Cilj je uporediti vremenske, frekvencijske i vremensko-frekvencijske karakteristike zvuka. Pored zvukova motorne testere, neophodno je uzeti u obzir i zvukove uobičajene u šumskom okruženju (ambijentalne zvukove), kao i zvukove koji su veoma slični motornoj testeri, kako bi se izbjegli lažni alarmi.

3.1 Zvuk i talasni oblici

Zvuk je proizveden vibracijom objekta koja podstiče čestice medijuma da osciluju i sudaraju se [22]. Sudarajući se čestice mijenjaju pritisak medijuma, a devijacija pritiska se u vidu talasa prenosi kroz prostor. Samim tim, zvuk se može posmatrati kao talas koji prenosi energiju sa jednog mjesta na drugo. Kad se čestice medijuma sudaraju i kad su zbijene, to se naziva kompresijom, a kad su razdvojene onda je to razrijeđenost (slika 1).



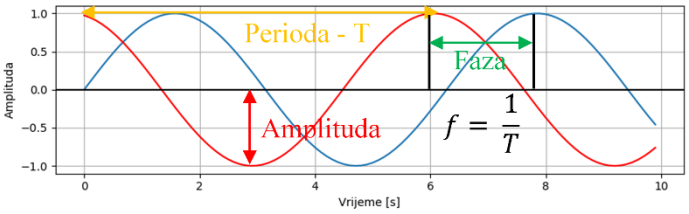
Slika 1. Kompresija i razrijeđenost čestica medijuma [23]

Talasni oblik je grafičko predstavljanje devijacije pritiska u odnosu na nulti nivo, u toku vremena. Pruža informacije o frekvenciji, intenzitetu, boji zvuka, itd. Talasni oblici se mogu podijeliti na periodične i aperiodične. Kod periodičnih talasnih oblika kompresije i razrijeđenosti se pojavljuju periodično, dok su kod aperiodičnih talasa izvorne vibracije nasumične ili haotične.

Posmatra se sinusoidalni talasni oblik zadat formulom (1), prikazan na slici 2.

$$y(t) = A \times \sin(2\pi f t + \varphi) \quad (1)$$

Perioda (T) je vrijeme za koje se izvrši jedna puna oscilacija. Frekvencija (*f*) je obrnuto proporcionalna periodi i predstavlja broj oscilacija u jedinici vremena. Izražava se u hercima (Hz). Spektralne komponente zvuka motorne testere nalaze se u frekvencijskom opsegu 500 - 8000Hz. Amplituda (A) je mjera devijacije pritiska. Faza (φ) omogućava vremensko pomjeranje signala u lijevu ili desnu stranu. Na osnovu faze, amplitude i frekvencije moguće je imati sve informacije o sinusoidi (slika 2).



Slika 2. Sinusoidalni talasni oblik

Snaga zvuka je stopa po kojoj se prenosi energija, tj. energija po jedinici vremena emitovana od izvora zvuka u svim pravcima [24]. Mjeri se u vatima $\left[W = \frac{J}{s}\right]$. Jačina ili intenzitet zvuka (I) je snaga zvuka po jedinici površine $\left[\frac{W}{m^2}\right]$. Čovjek ima sposobnost opažanja zvukova sa veoma malim intenzitetima i ta vrijednost se naziva prag čula:

$$TOH = 10^{-12}\frac{w}{m^2}, \tag{2}$$

dok je prag bola:

$$TOP = 10\frac{w}{m^2} \tag{3}$$

Intenzitet zvuka se obično predstavlja na logaritamskoj skali, jer je opseg između praga čula i praga bola veliki. Na taj način se kreira nova veličina koja se naziva *nivo intenziteta zvuka* i izražava se u decibelima [dB]. Decibel je odnos između dvije vrijednosti intenziteta. Zavisnost između intenziteta I koji se izražava u $\left[\frac{W}{m^2}\right]$ i nivoa intenziteta zvuka koji se izražava u dB data je formulom:

$$dB(I) = 10\log_{10}(\frac{I}{I_{TOH}}) \tag{4}$$

Ako je $I = I_{TOH}$, onda slijedi:

$$dB(I_{TOH}) = 10\log_{10}\left(\frac{I_{TOH}}{I_{TOH}}\right) = 0 \tag{5}$$

Nivo intenziteta zvuka od 0 dB znači da se radi o intenzitetu zvuka koji je jednak pragu čula TOH. Svaki put kad se nivo intenziteta zvuka poveća za 3dB, intenzitet zvuka se zapravo duplira.

Intenzitet, nivo intenziteta i snaga zvuka su objektivne mjere, dok je glasnost zvuka subjektivna mjera intenziteta, tj. koliko glasno čovjek doživljava zvuk. Glasnost zvuka zavisi od dužine trajanja, nivoa intenziteta i frekvencije zvuka. Zvuk koji je nivoa intenziteta 3dB i trajanja 100ms će se doživljavati kao manje glasan u odnosu na zvuk istog nivoa intenziteta ali trajanja 600ms. Glasnost zvuka se izražava u *Phones*-ima. Boja zvuka se može definisati kao razlika između dva zvuka sa istim intenzitetom, frekvencijom i trajanjem.

Kompleksni talasni oblici se mogu prikazati kao kombinacija sinusoida različitih amplituda, frekvencija i faza. Fundamentalna frekvencija je najniža frekvencija kompleksnog talasnog oblika [25], a harmonici se dobijaju množenjem fundamentalne frekvencije sa odgovarajućim brojem harmonika. Fundamentalna frekvencija se često naziva i prvi harmonik. Neharmoničnost je devijacija sinusoide u odnosu na najbliži idealan harmonik.

Audio signal je reprezentacija zvuka, koristeći promjenljiv nivo električnog napona za analogne signale ili niz binarnih brojeva za digitalne signale. Analogni signal u svakom vremenskom trenutku može imati bilo koju vrijednost, tj. može imati beskonačno mnogo vrijednosti. Kako računari razumiju samo dvije diskretne vrijednosti (0 i 1), potrebno je transformisati analogni u digitalni signal pomoću odabiranja i kvantizacije [25].

Odabiranje je proces uzimanja vrijednosti (odbirka) koju ima kontinualni signal u tačno definisanim vremenskim trenucima [26]. Odabrani signal je signal diskretan po vremenu, a odabiranje je diskretizacija po vremenu. Period odabiranja T_s je razmak između dva odbirka. Frekvencija odabiranja f_s je obrnuto proporcionalna periodu odabiranja i predstavlja broj odbiraka u jedinici vremena. Manja frekvencija odabiranja znači veći period odabiranja, a to dalje znači manja preciznost predstavljanja signala. Za audio signale obično se koriste frekvencije odabiranja od 22.05kHz i 44.1kHz. Da bi rekonstrukcija analognog signala na osnovu njegovih odbiraka bila moguća, potrebno je da frekvencija odabiranja f_s bude bar dva puta veća od maksimalne frekvencije u spektru analognog signala f_{max}. Ova teorema se naziva teorema o odabiranju i data je formulom:

$$fs \geq 2fmax, \qquad T \leq {}^{1}/_{2fmax} \tag{6}$$

Kvantizacija je dodjeljivanje vrijednosti odbircima iz konačnog skupa elemenata. Kvantizacija je diskretizacija po vrijednosti ili „zaokruživanje“. Nakon kvantizacije signal je diskretan i po vremenu i po vrijednosti. Broj kvantizacionih nivoa je $q = 2^n$, gdje je n broj bitova sa kojim se predstavlja pojedinačni odbirak ili bitska dubina. Što je broj kvantizacionih nivoa veći, to je bolja reprezentacija zvuka. Standardna bitska dubina n za audio CD (eng. *Compact Disk*) je 16 ili $q = 2^{16} = 65536$ kvantizationa nivoa.

Postupak snimanja zvuka je sljedeći: mehanički talas se prostire u pravcu mikrofona, oscilovanje membrane u mikrofonu stvara analogni električni signal, analogni signal putuje do zvučne kartice koja je zapravo ADC (eng. *Analog to Digital Convertor*), zvučna kartica vrši odabiranje, kvantizaciju i filtriranje. Koristi se niskopropusni filtar koji odbacuje sve frekvencije veće od Nikvistove frekvencije (polovina frekvencije odabiranja). Dobija se digitalni signal koji se može skladištiti na medijumu za prenos, računaru, itd. Postupak reprodukovanja zvuka je sljedeći: digitalni signal putuje do zvučne kartice koja ima ulogu DAC (eng. *Digital to Analog Convertor*), zatim analogni električni signal dolazi do zvučnika gdje izaziva oscilovanje membrane, što proizvodi mehanički talas ili zvuk.

3.2 Karakteristike zvuka

Karakteristike zvuka su svaki kvalitativno ili kvantitativno mjerljiv aspekt zvuka. Mogu se kategorizovati na osnovu nivoa apstrakcije (visok, srednji i nizak), vremenskog opsega (trenutne, segmentne i globalne) i domena signala (vremenski, frekvencijski i vremensko-frekvencijski) [27].

Karakteristike sa niskim nivoom apstrakcije imaju smisla za mašine, ali nemaju puno smisla za ljude. Tu se ubrajaju anvelopa, energija zvuka, sprektralni centroid, spektralni fluks, broj prolazaka kroz nulu, itd. Karakteristike sa srednjim nivoom apstrakcije počinju imati smisla za ljude. Tu spadaju visina tona, MFCC, itd. Karakteristike sa visokim nivoom apstrakcije imaju smisla za ljude i to su akordi, melodije, tempo, harmonija, ritam, žanr, itd.

Trenutne karakteristike su karakteristike koje daju trenutne informacije o zvuku (u vremenskom intervalu od 10ms). Segmentne karakteristike se računaju na osnovu segmenata zvuka (sekunde). Globalne karakteristike su jedan deskriptor za čitav audio signal.

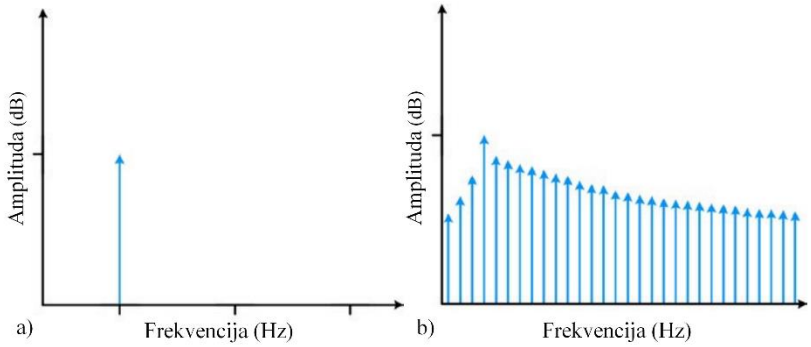
Karakteristike vremenskog domena su ekstrahovane iz talasnog oblika i to su anvelopa, srednja kvadratna energija, broj prolazaka kroz nulu, itd. Karakteristike frekvencijskog domena su karakteristike koje se odnose na frekvenciju. Za prelazak iz vremenskog u frekvencijski domen koristi se Furijeova transformacija (FT). Neke od karakteristika frekvencijskog domena su odnos energije, spektralni fluks i sprektralni centroid. Karakteristike vremensko-frekvencijskog domena kombinuju i vremenske i frekvencijske komponente zvuka. To su spektrogram, mel spektrogram, MFCC, konstantna Q transformacija, itd.

Ekstrahovanje karakteristika iz vremenskog domena se vrši na sljedeći način: talasni oblik signala se prosleđuje do ADC-a gdje se obavljaju operacije odabiranja i kvantizacije. Slijedi podjela signala na frejmove, što je ekvivalentno primjeni pravougaone prozor funkcije na signal. Broj odbiraka u frejmu naziva se veličina frejma i obično je stepen broja 2. Tipične vrijednosti za veličinu frejma su od 256 do 8192. Trajanje jednog frejma (*df*) se može izračunati pomoću formule:

$$df = \frac{1}{fs} \times K, \tag{7}$$

gdje je *fs* frekvencija odabiranja, a K veličina frejma. Nakon podjele signala na frejmove slijedi računanje karakteristika i to za svaki frejm posebno, a potom agregacija kako bi se dobio vektor karakteristika.

Procedura ekstrahovanja karakteristika iz frekvencijskog domena je slična kao i za vremenski domen. Nakon podjele na frejmove slijedi FT. Prilikom primjene FT na frejm, pretpostavlja se da je proslijeđen cijeli broj perioda periodičnog signala [28]. Krajnje tačke proslijeđenog signala se tumače kao spojene. Ukoliko je ovaj uslov ispunjen, rezultat će biti idealna FT, što je prikazano i slikom 3 a). Međutim, u najvećem broju slučajeva FT se primjenjuje na frejmove koji ne sadrže cijeli broj perioda. Kako se krajnje tačke signala tumače kao spojene, to će doći do naglih promjena na krajevima signala, što se naziva diskontinuitet. Diskontinuitet će se na spektar signala odraziti kao prisustvo komponenata na frekvencijama kojih zapravo nema u originalnom signalu. Ovaj fenomen se naziva *curenje spektra* i prikazan je na slici 3 b).

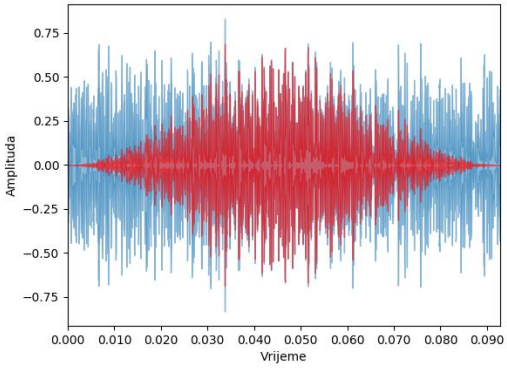


Slika 3. a) Idealna FT, b) Curenje spektra [28]

Dominantan pristup za ublažavanje fenomena curenja spektra je korišćenje prozor funkcije čija amplituda postepeno teži nuli na ivicama, i tako smanjuje diskontinuitete. Prozor funkcija koja je najčešće u upotrebi je Hannov prozor i on se definiše kao:

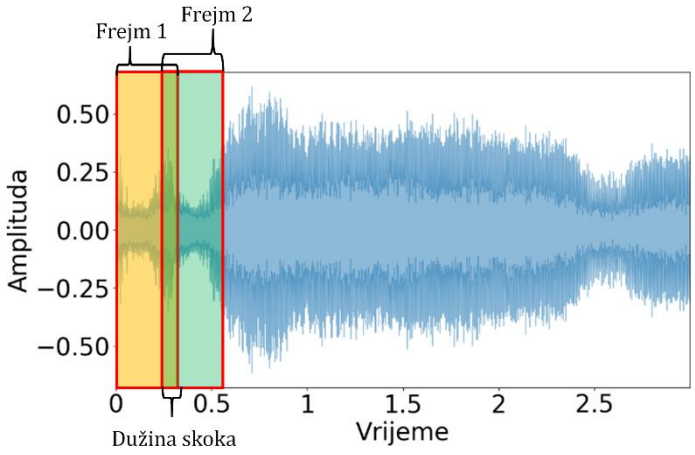
$$w(k) = 0.5 \times (1 - \cos\frac{2\pi k}{K-1}), k = 1 \dots K \tag{8}$$

Ovim pristupom je smanjena pojava curenja spektra, ali na račun gubitka informacija na krajevima frejma (slika 4). Plavom bojom je predstavljen frejm audio signala dobijen primjenom pravougaone prozor funkcije na signal, a crvenom bojom je predstavljen frejm audio signala dobijen primjenom Hannove prozor funkcije na signal.



Slika 4. Gubitak informacija na krajevima frejma

Da bi se izbjegao gubitak informacija u audio signalu vrši se preklapanje frejmova, pri čemu dužina skoka (eng. *hop length*) označava broj odbiraka koji se prozor pomjera udesno svaki put kad se uzima novi frejm (slika 5). Tipične vrijednosti za dužinu skoka su $^1/_4$ veličine frejma i $^1/_8$ veličine frejma.



Slika 5. Preklapanje frejmova

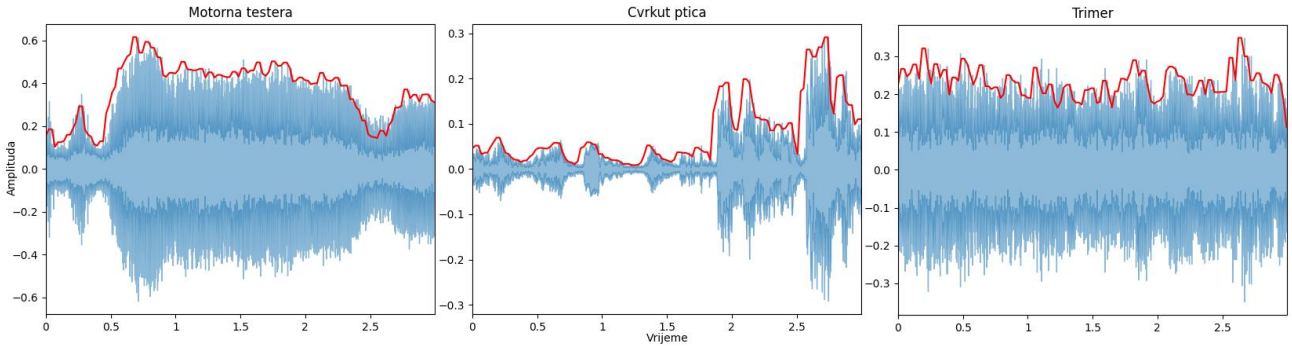
3.3 Karakteristike vremenskog domena

Karakteristike vremenskog domena audio signala korišćene u ovom radu su: anvelopa (eng. *amplitude envelope* - AE), srednja kvadratna energija (eng. *Root-Mean Square Energy* - RMSE) i broj prolazaka kroz nulu (eng. *Zero Crossing Rate* - ZCR). Sve ove karakteristike su trenutne, tj. daju trenutne informacije o zvuku.

AE opisuje kako se zvuk mijenja u toku vremena i predstavlja maksimalnu amplitudu svih odbiraka u frejmu [25]. Definiše se kao:

$$AE_t = \max_{t \cdot K \leq k \leq (t+1) \cdot K - 1} s(k), \tag{9}$$

gdje je *t* redni broj frejma, a *K* veličina frejma. AE daje grubu informaciju o jačini zvuka, ali je osjetljiva na nestandardne opservacije ili autlajere (eng. *outliers*). Uglavnom se koristi za detekciju početka muzičke note ili klasifikaciju muzičkog žanra. Na slici 6 prikazane su AE zvuka motorne testere, cvrkuta ptica i trimera (kosačice).



Slika 6. AE zvuka motorne testere, cvrkuta ptica i trimera

Za računanje i prikaz karakteristika audio signala u ovom istraživanju korišćen je programski jezik Python 3.7, a kao razvojno okruženje PyCharm Community Edition 2021.3.2. Besplatne verzije Python-a i PyCharm-a su dostupne na [29] i [30]. Za računanje AE korišćen je programski kod 1.

Programski kod 1.

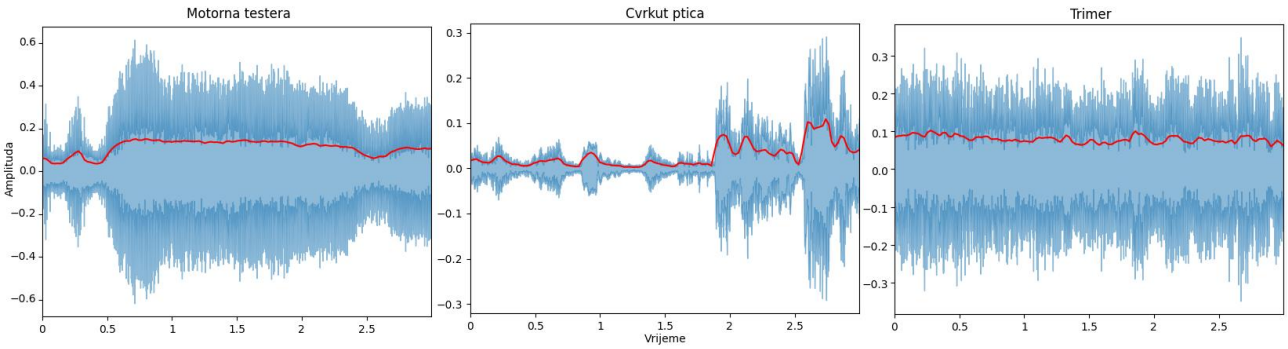
```
import librosa
import numpy as np

FRAME_SIZE = 2048
HOP_LENGTH = 512

file = "/putanja_do_fajla.wav"

def amplitude_envelope(signal, frame_size, hop_length):
    amplitude_envelope = []

    for i in range(0, len(signal), hop_length):
        current_frame_amplitude_envelope = max(signal[i:i + frame_size])
        amplitude_envelope.append(current_frame_amplitude_envelope)
    return np.array(amplitude_envelope)

if __name__ == "__main__":
    signal, sr = librosa.load(file, sr = 22050)
    ae_signal = amplitude_envelope(signal, FRAME_SIZE, HOP_LENGTH)
```

Linije koda koje započinju sa ključnom riječju *import* se odnose na module koje je potrebno učitati. *Librosa* je modul za analizu zvuka. *NumPy* je modul koji se koristi za rad sa nizovima, a posjeduje i funkcije za rad u domenu linearne algebre, FT i matrica.

RMSE se računa kao kvadratni korijen sume kvadrata energija svih odbiraka u frejmu podijeljene sa veličinom frejma:

$$RMS_t = \sqrt{\frac{1}{K}\sum_{k=t\cdot K}^{(t+1)\cdot K-1} s(k)^2} \tag{10}$$

Kao i AE, RMSE je takođe indikator jačine zvuka i koristi se za prepoznavanje žanra i audio klasifikaciju. Kako se RMSE računa na osnovu vrijednosti energija svih odbiraka u frejmu, manje je osjetljiva na autlajere. RMSE zvuka motorne testere, cvrkuta ptica i trimera prikazana je na slici 7.



Slika 7. RMSE zvuka motorne testere, cvrkuta ptica i trimera

Računanje RMSE-a u programskom jeziku Python se može izvršiti na dva načina. Prvi način podrazumijeva korišćenje modula *librosa* za audio analizu i ugrađene funkcije *rms*:

```
rms = librosa.feature.rms(signal, FRAME_SIZE, HOP_LENGTH)[0]
```

Drugi način za računanje RMSE-a je korišćenjem programskog koda 2, koji definiše funkciju *rms* na osnovu formule (10).

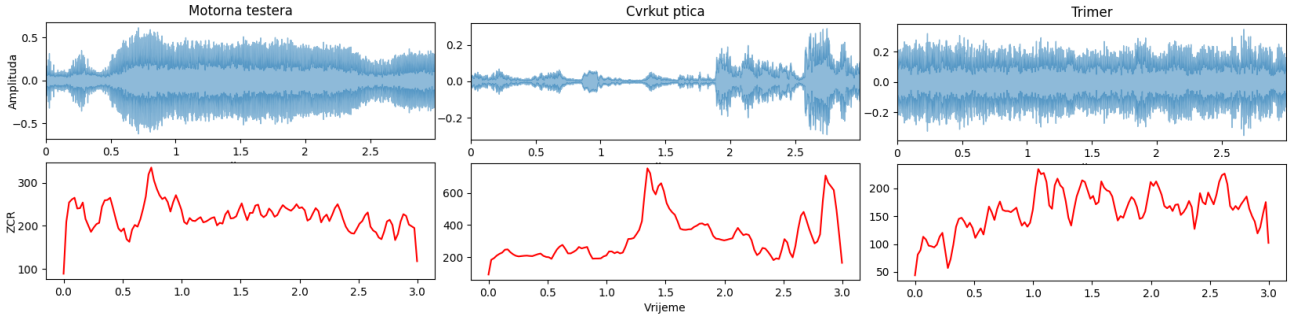
Programski kod 2.

```
def rms(signal, frame_size, hop_length):
    rms = []

    for i in range(0, len(signal), hop_length):
        rms_current_frame = np.sqrt(np.sum(signal[i:i + frame_size] ** 2)/
        frame_size)
        rms.append(rms_current_frame)

    return np.array(rms)
```

ZCR je broj prolazaka signala preko horizontalne ose. Računa se kao:

$$ZCR_t = \frac{1}{2}\sum_{k=t\cdot K}^{(t+1)\cdot K-1} \left|sgn\big(s(k)\big) - sgn\big(s(k+1)\big)\right|, \quad sgn(k) = \begin{cases} 1, & k > 0 \\ -1, & k < 0 \\ 0, & k = 0 \end{cases} \tag{11}$$

ZCR se koristi za prepoznavanje udaračkih instrumenata, procjenu visine tona, detekciju govora, itd. ZCR zvuka motorne testere, cvrkuta ptica i trimera prikazana je na slici 8.



Slika 8. ZCR zvuka motorne testere, cvrkuta ptica i trimera

Računanje ZCR-a u programskom jeziku Python je izvršeno pomoću modula *librosa* za audio analizu i ugrađene funkcije *zero_crossing_rate*:

```
zcr = librosa.feature.zero_crossing_rate(chainsaw, FRAME_SIZE, HOP_LENGTH)[0]
```

3.4 Furijeova transformacija

Furijeova transformacija (FT) predstavlja alat za konverziju signala iz vremenskog u frekvencijski domen, čiji je rezultat spektar signala. Na x osi spektra signala je prikazana frekvencija, a na y osi magnituda. Tamo gdje postoje skokovi na x osi, na tim frekvencijama su važne komponente originalnog signala. FT poredi originalni signal sa velikim brojem sinusoida različitih frekvencija [25]. Za svaku razmatranu frekvenciju rezultat poređenja su magnituda i faza. Magnituda govori o sličnosti originalnog signala i sinusoide, a faza koliko je sinusoidu potrebno pomjeriti lijevo ili desno za najveću sličnost. U slučaju male magnitude, zaključuje se da originalni signal ne sadrži sinusoidu sa razmatranom frekvencijom.

U literaturi, FT je često definisana pomoću sljedeće formule:

$$g\hat{}(f) = \int_{-\infty}^{\infty} g(t)e^{-i2\pi ft}dt, \tag{12}$$

gdje je *g(t)* analogni signal. Iz frekvencijskog domena u vremenski domen je moguće doći korišćenjem inverzne Furijeove transformacije (IFT):

$$g(t) = \int_{-\infty}^{\infty} g\hat{}(f)e^{i2\pi ft}df \tag{13}$$

Kad je u pitanju mikroprocesorska obrada zvuka, radi se sa digitalnim signalima, pa je potrebno definisati diskretnu Furijeovu transformaciju (DFT) [31]:

$$X(k) = \sum_{n=0}^{N-1} x(n) \cdot e^{\frac{-i2\pi kn}{N}}, k = 0,1,2,\dots,N-1, \tag{14}$$

gdje je $x(n)$ diskretni signal sa N odbiraka.

Inverzna diskretna Furijeva transformacija (IDFT) se definiše kao:

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(K)\cdot e^{\frac{i2\pi nk}{N}}, n = 0,1,2,\dots,N-1 \tag{15}$$

Algoritam računanja DFT sa što manjim brojem aritmetičkih operacija naziva se brza Furijeova transformacija (eng. *Fast Fourier Transformation* – FFT) [31]. Posmatra se diskretni signal $x(n)$ koji ima N odbiraka, pri čemu je N paran broj. Signal $x(n)$ se dijeli na dva signala i to: $x(n)$ za $0 \leq \mathrm{n} \leq \frac{N}{2}-1$ i $x(n)$ za $\frac{N}{2} \leq \mathrm{n} \leq N-1$. Prema formuli (14), DFT signala $x(n)$ se računa kao:

$$X(k) = \sum_{n=0}^{N-1} x(n)\cdot e^{\frac{-i2\pi kn}{N}} = \sum_{n=0}^{\frac{N}{2}-1} x(n)\cdot e^{\frac{-i2\pi kn}{N}} + \sum_{n=\frac{N}{2}}^{N-1} x(n)\cdot e^{\frac{-i2\pi kn}{N}} =$$

$$= \sum_{n=0}^{\frac{N}{2}-1}\left[x(n) + x\left(n+\frac{N}{2}\right)(-1)^k\right]\cdot e^{\frac{-i2\pi kn}{N}}, \quad k = 0,1,2,\dots,N-1 \tag{16}$$

Za parne brojeve $k = 2r$:

$$X(2r) = \sum_{n=0}^{\frac{N}{2}-1}\left[x(n) + x\left(n+\frac{N}{2}\right)\right]\cdot e^{\frac{-i2\pi rn}{\frac{N}{2}}} \tag{17}$$

Za neparne brojeve $k = 2r+1$:

$$X(2r+1) = e^{\frac{-i2\pi n}{N}}\sum_{n=0}^{\frac{N}{2}-1}\left[x - x\left(n+\frac{N}{2}\right)\right]\cdot e^{\frac{-i2\pi rn}{\frac{N}{2}}}, \tag{18}$$

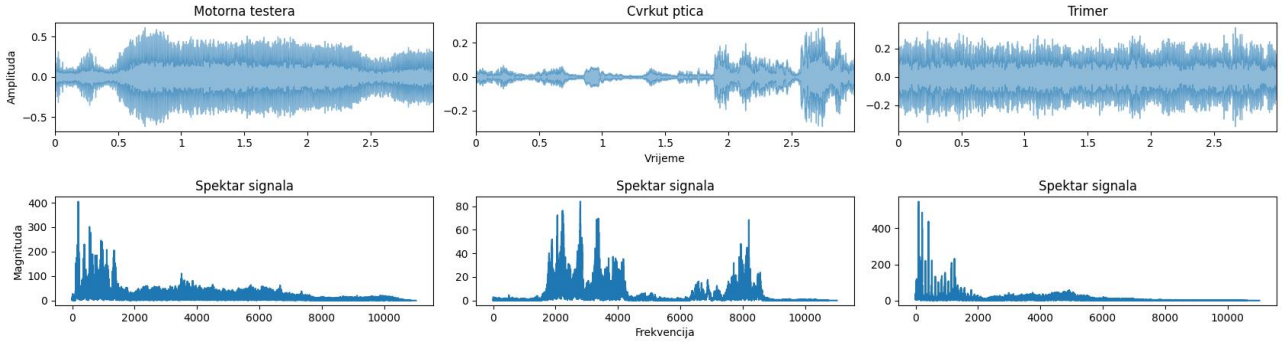
pri čemu je $r = 0,1,2,\dots,\frac{N}{2}$. Na ovaj način DFT od N elemenata podijeljena je na dvije DFT od $\frac{N}{2}$ elemenata, što je smanjilo broj aritmetičkih operacija sa N^2 na $2\cdot(\frac{N}{2})^2$. Takođe, DFT od $\frac{N}{2}$ elemenata je moguće podijeliti na dvije DFT od $\frac{N}{4}$ elemenata, itd.

U programskom jeziku Python, računanje DFT signala korišćenjem FFT algoritma se može izvršiti pomoću programskog koda 3.

Programski kod 3.

```
import librosa
import numpy as np

file = "/putanja_do_fajla.wav "
signal, sr = librosa.load(file, sr = 22050)

fft = np.fft.fft(signal)

magnitude = np.abs(fft)
frequency = np.linspace(0, sr, len(magnitude))
```

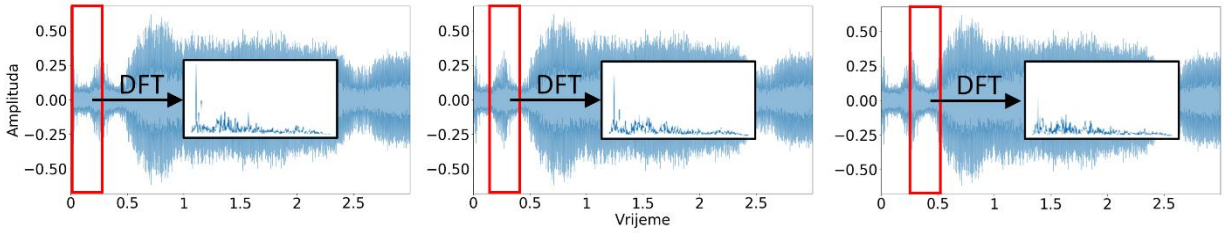
Spektar signala motorne testere, cvrkuta ptica i trimera, dobijen primjenom FFT na audio signale, prikazan je na slici 9.



Slika 9. Spektar signala motorne testere, cvrkuta ptica i trimera

3.5 Kratkotrajna Furijeova transformacija

DFT pruža informacije o frekvencijskim komponentama u čitavom signalu, ali ne i na kom vremenskom intervalu su prisutne frekvencijske komponente [25]. Te informacije se mogu dobiti primjenom kratkotrajne Furijeove transformacije (eng. *Short Time Fourier Transform* - STFT), koja razmatra manje djelove signala dobijene množenjem originalnog signala i prozor funkcije. Prozor funkcija je različita od nule samo na kratkom vremenskom intervalu. Kao prozor funkcija, obično se koristi Hannov prozor definisan formulom (8). Nakon primjene DFT na jedan frejm, prozor funkcija se pomjera udesno tako da se frejmovi preklapaju. Preklapanje obično iznosi 75% dužine prozora, tj. pomjeraj prozora je obično 25% dužine prozora. Postupak računanja STFT je prikazan na slici 10.



Slika 10. Postupak računanja STFT

STFT diskretnog signala $x(n)$ se može definisati kao:

$$S(m,k) = \sum_{n=0}^{N-1} x(n+mH) \cdot w(n) \cdot e^{\frac{-i2\pi kn}{N}}, \tag{19}$$

gdje je m redni broj frejma, H – dužina skoka, $w(n)$ – prozor funkcija, dok se N odnosi na broj odbiraka u frejmu.

Rezultat STFT je spektralna matrica, čije su dimenzije [*broj frekvencijskih binova, broj frejmova]*. Broj frekvencijskih binova predstavlja broj odbiraka u frekvencijskom domenu i računa se kao: *broj frekvencijskih binova = veličina frejma/2 + 1*. Broj frejmova se računa kao: *broj frejmova = (broj odbiraka – veličina frejma)/dužina skoka + 1*. Veličina frejma i dužina skoka su obično stepeni broja 2, pa je moguće koristiti FFT.

STFT omogućava računanje spektrograma koji je jedna od najvažnijih karakteristika kada je u pitanju obrada zvučnog signala. Prema [25], spektrogram predstavlja magnitudu STFT na kvadrat:

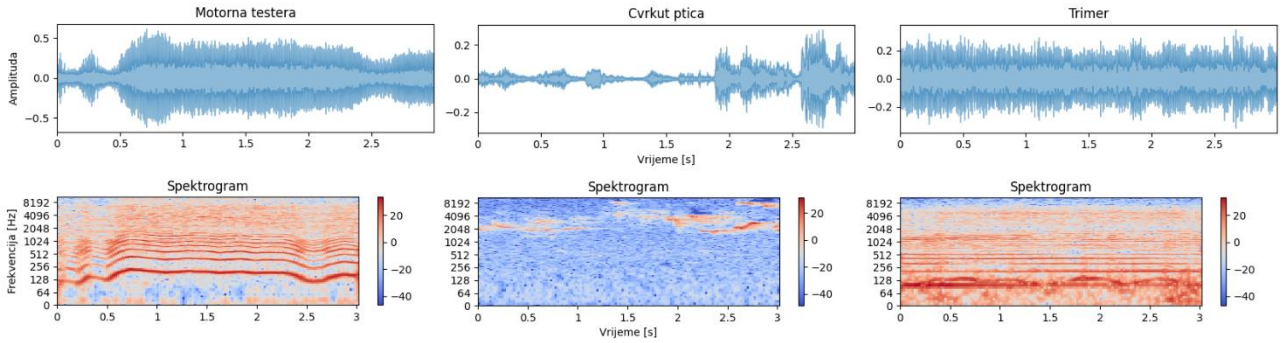
$$Y(m,k) = |S(m,k)|^2 \tag{20}$$

U grafičkoj reprezentaciji na horizontalnoj osi je prikazano vrijeme, na vertikalnoj frekvencija, a učešće frekvencijske komponente u originalnom signalu je predstavljeno bojom. Za računanje spektrograma u programskom jeziku Python, korišćen je programski kod 4, dat u nastavku:

Programski kod 4.

```
import librosa

FRAME_SIZE = 2048
HOP_LENGTH = 512

file = "/putanja_do_fajla.wav"

signal, sr = librosa.load(file)

stft = librosa.core.stft(signal, hop_length= HOP_LENGTH, n_fft = FRAME_SIZE)
spectrogram = np.abs(stft)
log_spectrogram = librosa.power_to_db(spectrogram)
```

Spektrogram zvuka motorne testere, cvrkuta ptica i trimera prikazan je na slici 11.



Slika 11. Spektrogram zvuka motorne testere, cvrkuta ptica i trimera

3.6 Mel spektrogram

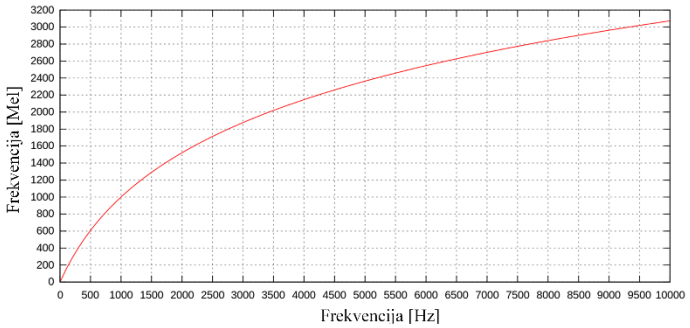
Naziv *mel* je nastao od engleske riječi *melody*. Mel skala je logaritamska skala, dobijena kao rezultat psihoakustičnog eksperimenta u kojem su slušaoci ocjenjivali udaljenost između zvukova na različitim frekvencijama [32]. Kao referentna tačka između linearne skale i mel skale uzeta je frekvencija od 1000Hz, što je jednako 1000Mel (slika 12).

Formula za prelazak iz linearne skale *f* u mel skalu *m* je:

$$m = 2595 log_{10}^{(1+\frac{f}{700})} = 1127 \ln\left(1 + \frac{f}{700}\right), \tag{21}$$

a inverzna formula za prelazak iz mel skale *m* u linearnu skalu *f* je:

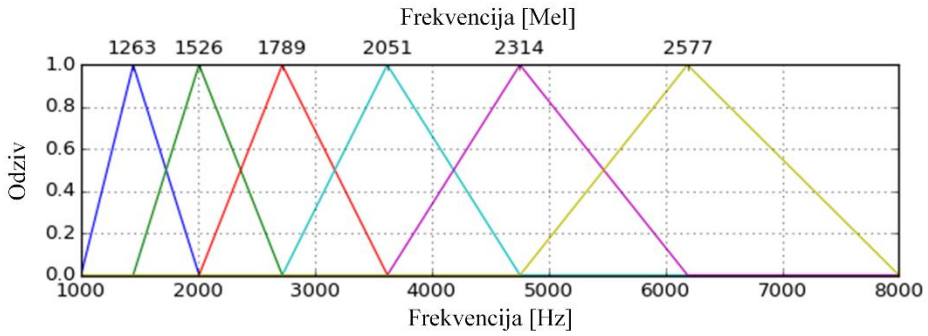
$$f = 700\left(10^{\frac{m}{2595}} - 1\right) = 700\left(e^{\frac{m}{1127}} - 1\right) \tag{22}$$



Slika 12. Mel skala [32]

Algoritam za dobijanje mel spektrograma se može definisati na sljedeći način:

1. primijeniti STFT na audio signal;
2. konvertovati amplitudu u dB pomoću formule (4);
3. konvertovati frekvenciju u mel skalu:
   3.1. izabrati broj *mel bands*; broj *mel bands* je parametar koji može da varira, zavisno od problema; obično se kreće između 40 i 130.
   3.2. konstruisati mel filtar banku:
      3.2.1. konvertovati najveću i najmanju frekvenciju signala iz Hz u Mel pomoću formule (21);
      3.2.2. na dobijenom mel opsegu postaviti onoliko jednako udaljenih tačaka koliko je izabrano *mel bands*; ove tačke se nazivaju centralne frekvencije;
      3.2.3. konvertovati tačke nazad u Hz pomoću formule (22);
      3.2.4. kreirati trouglasti filtar (slika 13);
   3.3. primijeniti mel filtar banku na spektrogram;



Slika 13. Mel filter banka za *mel bands* = 6 [33]

Svaki filtar u mel filtar banci je trouglast sa odzivom 1 na centralnoj frekvenciji. Odziv se linearno smanjuje prema 0 ka centralnim frekvencijama dva susjedna filtra, gdje je odziv 0 [33]. Oblik mel filtar banke je: *M = [broj mel bands, veličina frejma/2 + 1]*.

Posljednji korak je primjena mel filtar banke na spektrogram. Spektrogram je istih dimenzija kao i spektralna matrica: *Y = [veličina frejma/2 + 1, broj frejmova]*.

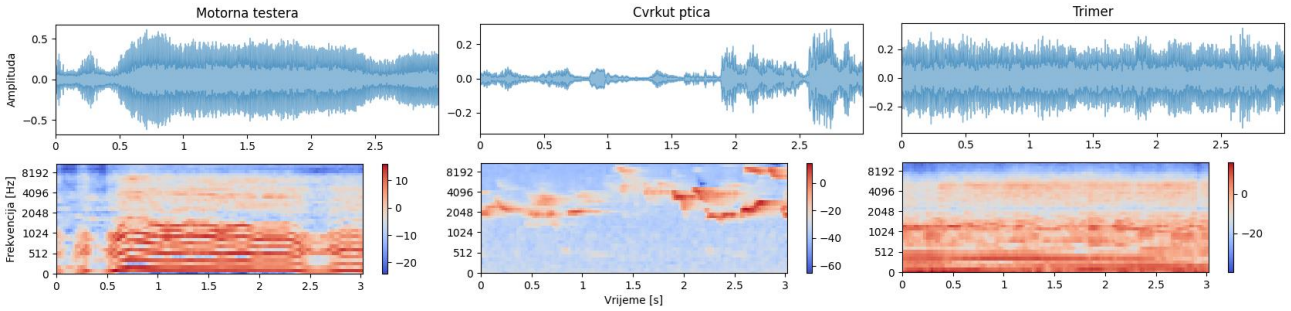
Kolone matrice *M* odgovaraju redovima matrice *Y*, pa se može izvršiti matrično množenje. Množenjem matrica *M* i *Y* dobija se mel spektrogram, koji će imati dimenzije: *MY = [broj mel bands, broj frejmova]*.

Programski kod 5 je korišćen za računanje mel spektrograma u programskom jeziku Python.

Programski kod 5.

```
import librosa

audio_file = "putanja_do_fajla.wav"
signal, sr = librosa.load(audio_file)

mel_spectrogram = librosa.feature.melspectrogram(signal, sr=sr, n_fft=2048,
                  hop_length = 512, n_mels = 40)
log_mel_spectrogram = librosa.power_to_db(mel_spectrogram)
```

Na slici 14 prikazan je mel spektrogram zvuka motorne testere, cvrkuta ptica i trimera, pri čemu je broj *mel bands* = 40.



Slika 14. Mel spektrogram zvuka motorne testere, cvrkuta ptica i trimera

3.7 Mel Frequency Cepstral Coefficients

Mel Frequency Cepstral Coefficients - MFCC je jedna od najčešće korišćenih karakteristika kad je u pitanju analiza audio signala, koja daje informacije o teksturi/boji zvuka. Na osnovu naziva se može zaključiti sljedeće:

- *Mel Frequency* - koristi se mel skala,
- *coefficients* - dobijaju se vrijednosti koje opisuju zvuk,
- *cepstral* – od riječi cepstrum.

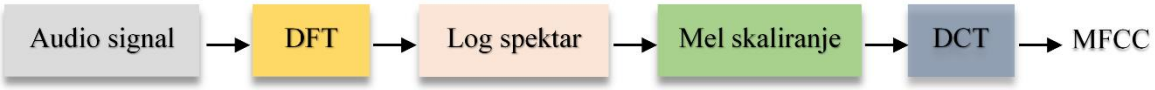
Cepstrum vodi porijeklo još iz 1960. godine kada se koristio za proučavanje ehoa u seizmičkim talasima, zatim 1970. godine za prepoznavanje govora i 2000. godine za obradu muzike [34]. Cepstrum je rezultat IDFT nad logaritmom spektra signala:

$$C(x(n)) = IDFT[\log(DFT[x(n)])] \tag{23}$$

Kako je navedeno u [35], umjesto IDFT se može koristiti DFT jer raspored frekvencija u spektru ostaje isti, samo se mijenja faktor skaliranja:

$$C(x(n)) = DFT[\log(DFT[x(n)])] \tag{24}$$

Uključivanjem informacija o ljudskoj percepciji u cilju poboljšanja cepstralne reprezentacije, primjenjuje se mel skaliranje na cepstrum i time se dobijaju MFCC. Razlika je u tome što se umjesto IDFT koristi diskretna kosinusna transformacija (DCT). Prednost koju DCT ima u odnosu na IDFT je u tome što su rezultujući koeficijenti DCT-a realne vrijednosti, što olakšava naknadnu obradu i skladištenje. Algoritam računanja MFCC-a je prikazan na slici 15.



Slika 15. Algoritam računanja MFCC-a

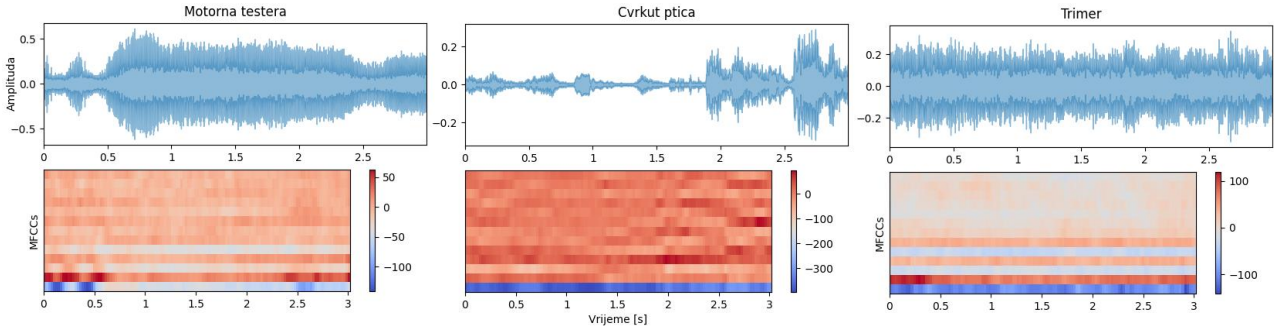
Algoritam za dobijanje MFCC-a se primjenjuje na svaki frejm. Tradicionalno se razmatra prvih 13 MFCC-eva od ukupno 39, jer oni zapravo sadrže najviše informacija o zvuku. Način na koji se mogu poboljšati performanse ML algoritma kada je u pitanju analiza MFCC-a je računanje prvog i drugog derivata MFCC-a, tj. ΔMFCC i ΔΔMFCC. Odnos MFCC-a trenutnog i prethodnog frejma je ΔMFCC. Na isti način, ΔΔMFCC je odnos ΔMFCC-a trenutnog i prethodnog frejma.

Za računanje MFCC-a u programskoj jeziku Python, korišćen je programski kod 6.

Programski kod 6.

```
import librosa

FRAME_SIZE = 2048
HOP_LENGTH = 512
NUM_MFCC = 13

audio_file = "putanja_do_fajla.wav"
signal, sr = librosa.load(audio_file, 22050)

MFCCs = librosa.feature.mfcc(signal, n_fft= FRAME_SIZE, hop_length=HOP_LENGTH,
                             n_mfcc = NUM_MFCC)
```

Na slici 16 prikazani su MFCC zvuka motorne testere, cvrkuta ptica i trimera, pri čemu je broj MFCC = 13.



Slika 16. MFCC zvuka motorne testere, cvrkuta ptica i trimera

3.8 Karakteristike frekvencijskog domena

Karakteristike frekvencijskog domena se ekstrahuju iz spektrograma audio signala. Karakteristike frekvencijskog domena korišćene u ovom radu su spektralni centroid (eng. *Spectral Centroid* - SC) i spektralni opseg (eng. *Spectral Bandwidth* - SB).

SC definisan je kao frekvencijski opseg gdje je skoncentrisana najveća količina energije. Računa se po formuli:

$$SC_t = \frac{\sum_{n=1}^{N} m_t(n) \cdot n}{\sum_{n=1}^{N} m_t(n)}, \tag{25}$$

gdje je n frekvencijski bin, t redni broj frejma, $m_t(n)$ magnituda signala na frekvencijskom binu n i frejmu t, a N ukupan broj frekvencijskih binova. Frekvencijski bin predstavlja interval između odbiraka u frekvencijskom domenu. SC se povezuje sa mjerom „svjetline“ zvuka (prisustvo visokih frekvencija) i obično se koristi u digitalnoj obradi audio signala za određivanje boje zvuka.

SB je izveden iz SC i označava spektralni opseg interesantnih dijelova u signalu. Matematički, to je srednja vrijednost udaljenosti frekvencijskih opsega od SC:

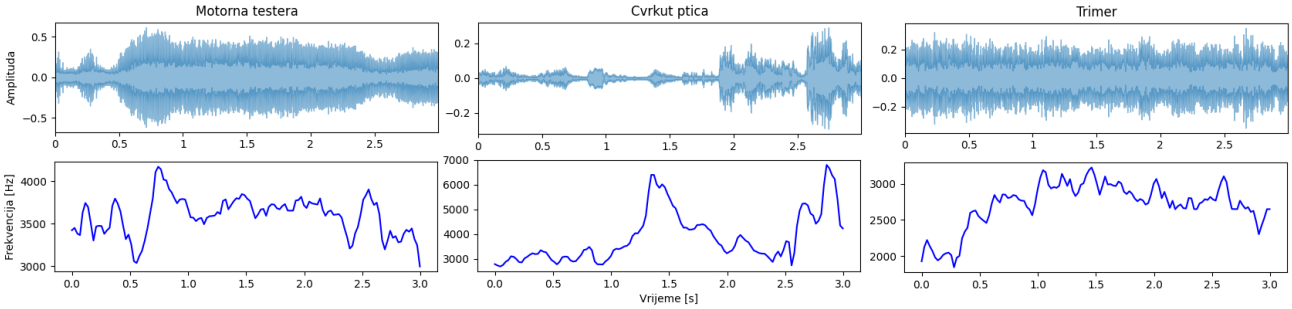
$$SB_t = \frac{\sum_{n=1}^{N} |n - SC_t| \cdot m_t(n)}{\sum_{n=1}^{N} m_t(n)} \tag{26}$$

Programski kod 7 je korišćen za računanje SC i SB u programskom jeziku Python.

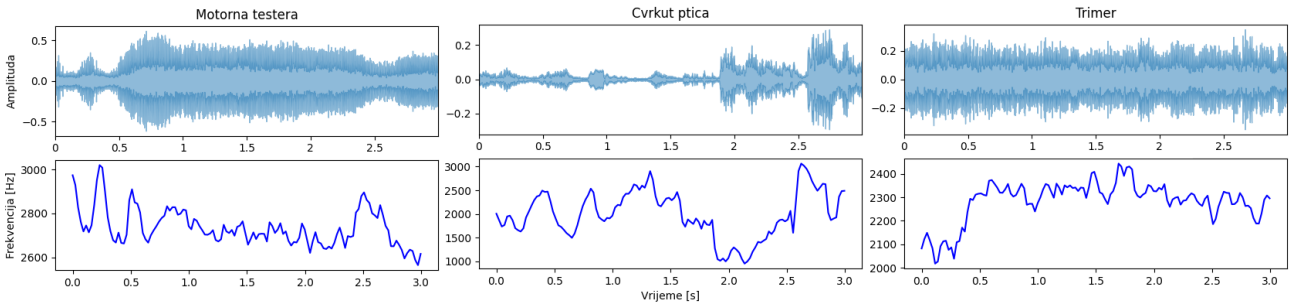
Programski kod 7.

```
import librosa

FRAME_SIZE = 2048
HOP_LENGTH = 512

audio_file = "putanja_do_fajla.wav"
signal, sr = librosa.load(audio_file)

sc = librosa.feature.spectral_centroid(y= signal, sr = sr, n_fft = FRAME_SIZE,
                                       hop_length= HOP_LENGTH)[0]

sb = librosa.feature.spectral_bandwidth(y= signal, sr = sr, n_fft =FRAME_SIZE,
                                       hop_length= HOP_LENGTH)[0]
```

Na slici 17 prikazan je SC, a na slici 18 SB zvuka motorne testere, cvrkuta ptica i trimera.



Slika 17. SC zvuka motorne testere, cvrkuta ptica i trimera



Slika 18. SB zvuka motorne testere, cvrkuta ptica i trimera

4 NEURALNE MREŽE KAO MODEL ZA DETEKCIJU ZVUKA

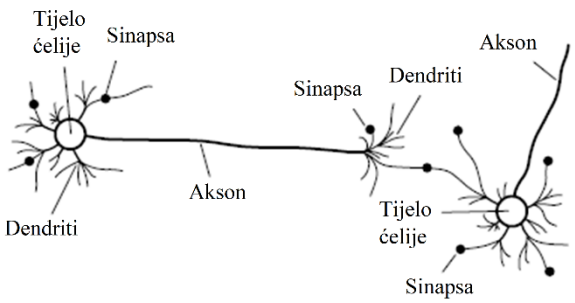
Neuralne mreže su jedan od algoritama ML, koji se odnosi na sisteme sposobne da uče na osnovu priloženih podataka. Takvim sistemima je potrebno obezbijediti veliki broj ulaznih podataka, na osnovu kojih se izvode pravila. Zatim se istim sistemima proslijedi novi podatak i oni će na osnovu pravila donijeti odluku. ML se može kategorizovati na:

- *učenje sa nadzorom* (eng. *supervised learning*) - sistemu se prosljeđuje skup označenih podataka; cilj je da sistem na osnovu ulazno-izlaznog mapiranja nauči pravila [36];
- *učenje bez nadzora* (eng. *unsupervised learning*) - sistemu se prosljeđuje skup neoznačenih podataka; cilj sistema je da pronađe smisao u podacima i da nauči da razvrstava podatke;
- *učenje sa podsticanjem* (eng. *reinforcement learning*) - ulazno-izlazno mapiranje se izvodi kroz kontinualnu interakciju sa okruženjem; sistem percipira okruženje i preduzima akcije; željene akcije će biti nagrađene, a nepoželjne kažnjene; vremenom, sistem nauči da izbjegava nepoželjne akcije.

Predmet istraživanja ovog rada je detekcija zvukova motorne testere, pomoću algoritama dubokog učenja. Duboko učenje se odnosi na neuralne mreže koje imaju više od jednog skrivenog sloja. U poređenju sa ostalim ML algoritmima koji su pogodni za male količine podataka, duboko učenje je efikasno samo sa velikim količinama podataka. Međutim, trening sa velikim količinama podataka i složenim modelima traje dugo i ima visoke zahtjeve vezano za hardversku platformu. Zbog toga se za jednostavne probleme obično primjenjuju tradicionalni algoritmi mašinskog učenja, dok je duboko učenje idealno za kompleksne probleme. Ovaj rad je tip učenja sa nadzorom.

4.1 Potpuno povezane neuralne mreže

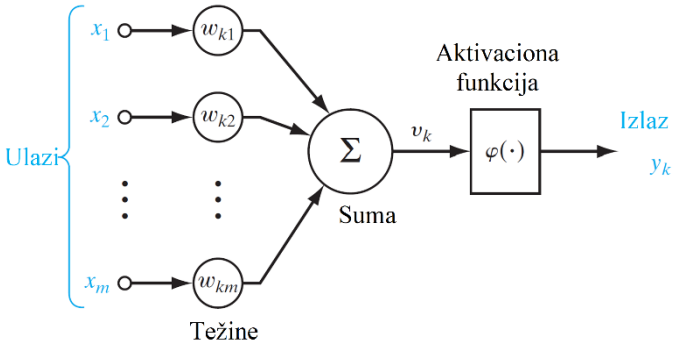
Neuralna mreža se u [37] definiše kao model odlučivanja zasnovan na čovjekovom mozgu. Čovjekov mozak se sastoji od velikog broja međusobno povezanih nervnih ćelija ili neurona. Neuron je osnovna jedinica za obradu informacija koja je fundamentalna za rad neuralnih mreža. Biološki neuron se sastoji od tijela ćelije, brojnih vlakana koja se nazivaju dendriti i jednog dugog vlakna koje se zove akson. Akson se na krajevima grana i spaja sa dendritima drugih neurona, kreirajući veze koje se nazivaju sinapse. Šematski prikaz biološkog neurona dat je slikom 19.



Slika 19. Šematski prikaz neurona [37]

Učenje je osnovna i suštinska karakteristika biološke neuralne mreže. Lakoća i prirodnost sa kojom biološka neuralna mreža uči, doveli su do pokušaja da se simulira na računaru. Upravo tako nastaju vještačke neuralne mreže (eng. *Artificial Neural Network* – ANN).

Vještački ekvivalent biološkog neurona prikazan je na slici 20. Tri osnovna dijela vještačkog neurona su skup sinapsi, suma i aktivaciona funkcija [36]. Svaku sinapsu karakteriše sinaptička težina, koja je osnovno sredstvo za dugoročnu memoriju ANN-a. Težine određuju snagu, ili drugim riječima važnost, svakog neuronskog ulaza. Neuralna mreža uči kroz ponovljena prilagođavanja težina. Težina vještačkog neurona može da leži u opsegu koji uključuje i pozitivne i negativne vrijednosti. Suma vrši sabiranje ulaznih signala pomnoženih odgovarajućim sinaptičkim težinama. Aktivaciona funkcija ograničava opseg amplitude izlaznog signala na neku konačnu vrijednost i unosi nelinearnost u ukupnu funkciju prenosa neuralne mreže. Normalizovani opseg amplitude izlaznog signala obično pripada intervalu [0,1] ili [-1,1].



Slika 20. Model vještačkog neurona [36]

Neuron *k*, prikazan na slici 20, se može matematički opisati sljedećim formulama:

$$v_k = \sum_{j=1}^{m} w_{kj} \cdot x_j \tag{27}$$

$$y_k = \varphi(v_k), \tag{28}$$

gdje su x_1, x_2, ..., x_m ulazni signali, w_{k1}, w_{k2}, ..., w_{km} sinaptičke težine neurona *k*, v_k je *net input* ili suma proizvoda ulaznih signala i odgovarajućih sinaptičkih težina, $\varphi(v)$ je aktivaciona funkcija, a y_k je izlazni signal neurona. Ulazni signali neurona mogu biti ulazni podaci ili izlazi drugih neurona. Izlazni signal može biti konačno rješenje ili ulaz u druge neurone.

Aktivaciona funkcija $\varphi(v)$ definiše izlaz neurona. Dva osnovna tipa aktivacione funkcije su:

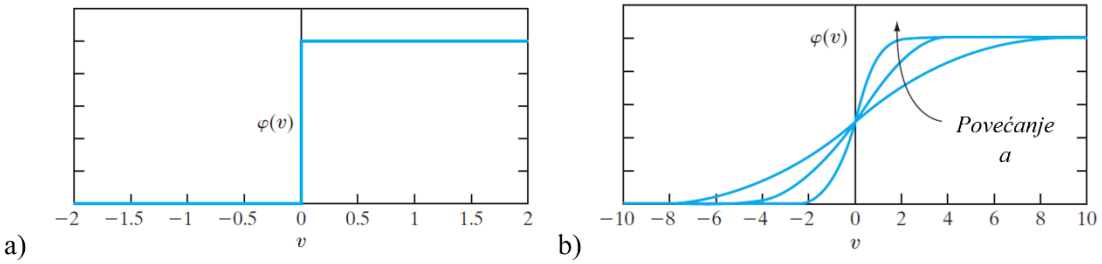
- funkcija praga prikazana na slici 21 a), koja se opisuje formulom:

$$\varphi(v) = \begin{cases} 1, v \geq 0 \\ 0, v < 0 \end{cases} \tag{29}$$

- sigmoid funkcija prikazana na slici 21 b). Jedna je od najčešće korišćenih aktivacionih funkcija, koja se opisuje formulom:

$$\varphi(v) = \frac{1}{1 + e^{-av}}, \tag{30}$$

gde je *a* parametar nagiba sigmoidne funkcije. Variranjem parametra *a*, mogu se dobiti sigmoid funkcije različitih nagiba. Kad *a* teži beskonačnosti, sigmoid funkcija postaje funkcija praga.



Slika 21. a) Funkcija praga, b) Sigmoid funkcija [36]

Funkcija praga kao izlaz daje vrijednost 0 ili 1, dok sigmoid funkcija kao izlaz daje opseg vrijednosti između 0 i 1. Također, sigmoidna funkcija je diferencijabilna, dok funkcija praga nije. Diferencijabilnost je važna karakteristika neuralnih mreža što će biti objašnjeno u poglavlju 4.2.

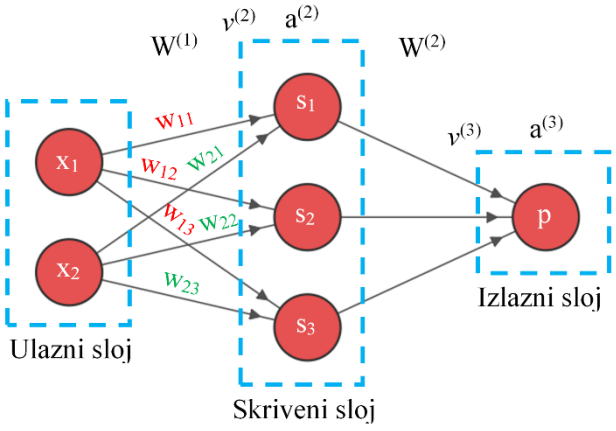
U nekim slučajevima, potrebno je da izlaz aktivacione funkcije bude u opsegu između -1 i 1. Tad se funkcija praga definisana formulom (29) zapisuje kao:

$$\varphi(v) = \begin{cases} 1, v > 0 \\ 0, v = 0 \\ -1, v < 0 \end{cases} \tag{31}$$

Ovako definisana funkcija praga naziva se *sign* funkcija. Za odgovarajući oblik sigmoid funkcije u opsegu između -1 i 1, koristi se hiperbolička tangens funkcija definisana formulom:

$$\varphi(v) = \tanh(v) = \frac{e^{v} - e^{-v}}{e^{v} + e^{-v}} \tag{32}$$

Neuroni u ANN-u su raspoređeni u slojeve i to u tri tipa slojeva: ulazni, skriveni i izlazni sloj. Ukoliko se ANN sastoji samo od ulaznog i izlaznog sloja onda se radi o *single layer perceptron* (SLP) neuralnoj mreži. Druga klasa neuralnih mreža se razlikuje od SLP-a po prisustvu jednog ili više skrivenih slojeva i naziva se *multi layer perceptron* (MLP) neuralna mreža. Dodavanjem skrivenih slojeva, mreža je sposobna da od ulaza izvuče karakteristike višeg reda, koje nemaju smisla za čovjeka. Na slici 22 prikazana je neuralna mreža koja pripada klasi MLP, sa jednim skrivenim slojem.



Slika 22. MLP neuralna mreža sa jednim skrivenim slojem

Za neuralnu mrežu prikazanu na slici 22 se kaže da je potpuno povezana, jer je svaki neuron jednog sloja povezan sa svakim neuronom susjednih slojeva. Ako neke od veza nedostaju, onda se radi o djelimično povezanoj mreži.

Veze između ulaznog i skrivenog sloja su okarakterisane sinaptičkim težinama koje se mogu predstaviti u obliku težinske matrice $W^{(1)}$:

$$W^{(1)} = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{bmatrix} \tag{33}$$

Prema formuli (27) ulaz u skriveni sloj ili *net input* predstavljen vektorom $v^{(2)}$ se računa kao proizvod ulaznog vektora x i težinske matrice $W^{(1)}$. Važi da je:

$$v^{(2)} = x \cdot W^{(1)} = [x_1 \quad x_2] \cdot \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{bmatrix} = [v_1 \quad v_2 \quad v_3] \tag{34}$$

Aktivaciona funkcija φ se primjenjuje nad vektorom $v^{(2)}$:

$$a^{(2)} = \varphi(v^{(2)}) \tag{35}$$

Net input $v^{(3)}$ se računa kao proizvod vektora $a^{(2)}$ i težinske matrice $W^{(2)}$:

$$v^{(3)} = a^{(2)} W^{(2)} \tag{36}$$

Izlaz iz neuralne mreže p se računa primjenom aktivacione funkcije φ nad *net inputom* $v^{(3)}$:

$$p = a^{(3)} = \varphi(v^{(3)}) \tag{37}$$

4.2 Treniranje potpuno povezanih neuralnih mreža

Za treniranje potpuno povezanih neuralnih mreža se koristi algoritam propagacije unazad, grafički predstavljen slikom 23.

Slika 23. Algoritam propagacije unazad

Neka je data potpuno povezana neuralna mreža prikazana na slici 22, koja se sastoji od ulaznog, skrivenog i izlaznog sloja. Elementi težinskih matrica $W^{(1)}$ i $W^{(2)}$ su proizvoljne vrijednosti između 0 i 1. Na ulaz neuralne mreže je doveden signal, predstavljen u obliku vektora x. Pomoću formula (34) - (37), vrši se računanje predikcije neuralne mreže, označene slovom p.

Sljedeći korak algoritma propagacije unazad je računanje greške E, koja je funkcija od predikcije p i željenog izlaza y: $E = E(p, y)$. Neka je u ovom slučaju funkcija greške srednja kvadratna greška (eng. *Mean Squared Error* – MSE) definisana formulom:

$$E = E(p, y) = \frac{1}{2}(p - y)^2 \tag{38}$$

Zatim je potrebno izračunati gradijent funkcije greške po težinskoj matrici, tj. kako se mijenja funkcija greške u zavisnost od promjene težinske matrice. Predikcija neuralne mreže *p* zavisi od ulaza *x* i od težinske matrice W: $p = f(x, W) = a^{(3)}$.

Funkcija greške *E* se može zapisati:

$$E = E(p, y) = E(f(x, W), y) \tag{39}$$

Sada je moguće pronaći gradijent funkcije greške *E* po težinskoj matrici $W^{(2)}$ koristeći teoremu o izvodu složene funkcije [38]:

$$\frac{\partial E}{\partial W^{(2)}} = \frac{\partial E}{\partial a^{(3)}} \frac{\partial a^{(3)}}{\partial v^{(3)}} \frac{\partial v^{(3)}}{\partial W^{(2)}} \tag{40}$$

Kako je funkcija greške $E = \frac{1}{2}(a^{(3)} - y)^2$, to slijedi:

$$\frac{\partial E}{\partial a^{(3)}} = 2 \cdot \frac{1}{2}(a^{(3)} - y) \cdot 1 = a^{(3)} - y \tag{41}$$

Neka je aktivaciona funkcija sigmoid funkcija. Važi da je $a^{(3)} = \varphi\left(v^{(3)}\right) = \frac{1}{1+e^{-v^{(3)}}}$. Slijedi:

$$\frac{\partial a^{(3)}}{\partial v^{(3)}} = \varphi'\left(v^{(3)}\right) = \varphi(v^{(3)}) \cdot (1 - \varphi\left(v^{(3)}\right)) \tag{42}$$

Net input $v^{(3)}$ je definisan formulom (36) pa slijedi:

$$\frac{\partial v^{(3)}}{\partial W^{(2)}} = a^{(2)} \tag{43}$$

Sada je gradijent funkcije greške *E* po težinskoj matrici $W^{(2)}$:

$$\frac{\partial E}{\partial W^{(2)}} = (a^{(3)} - y) \cdot \varphi'\left(v^{(3)}\right) \cdot a^{(2)} \tag{44}$$

Zatim je potrebno pronaći gradijent funkcije greške *E* po težinskoj matrici $W^{(1)}$:

$$\frac{\partial E}{\partial W^{(1)}} = \frac{\partial E}{\partial a^{(2)}} \frac{\partial a^{(2)}}{\partial v^{(2)}} \frac{\partial v^{(2)}}{\partial W^{(1)}} \tag{45}$$

Korišćenjem teoreme o izvodu složene funkcije može se zaključiti da je:

$$\frac{\partial E}{\partial a^{(2)}} = \frac{\partial E}{\partial a^{(3)}} \frac{\partial a^{(3)}}{\partial v^{(3)}} \frac{\partial v^{(3)}}{\partial a^{(2)}} \tag{46}$$

Na osnovu formula (45) i (46) važi:

$$\frac{\partial E}{\partial W^{(1)}} = \frac{\partial E}{\partial a^{(3)}} \frac{\partial a^{(3)}}{\partial v^{(3)}} \frac{\partial v^{(3)}}{\partial a^{(2)}} \frac{\partial a^{(2)}}{\partial v^{(2)}} \frac{\partial v^{(2)}}{\partial W^{(1)}} \tag{47}$$

Prva dva činioca su već izračunata u formulama (41) i (42). Iz formule (36) slijedi da je:

$$\frac{\partial v^{(3)}}{\partial a^{(2)}} = W^{(2)} \tag{48}$$

Aktivaciona funkcija je sigmoid funkcija, pa je: $a^{(2)} = \varphi\left(v^{(2)}\right) = \frac{1}{1+e^{-v^{(2)}}}$. Važi da je:

$$\frac{\partial a^{(2)}}{\partial v^{(2)}} = \varphi'\left(v^{(2)}\right) = \varphi(v^{(2)}) \cdot (1 - \varphi\left(v^{(2)}\right)) \tag{49}$$

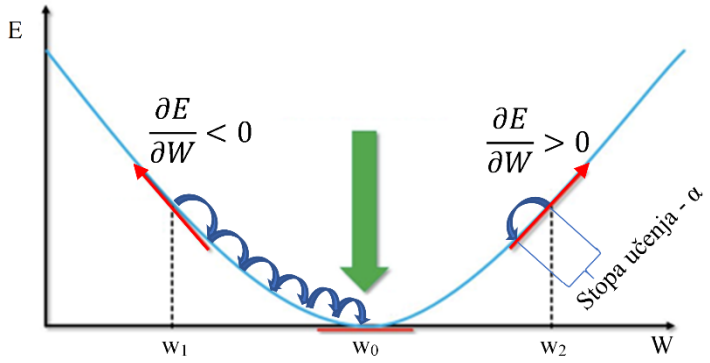
Iz formule (34) slijedi:

$$\frac{\partial v^{(2)}}{\partial W^{(1)}} = x \tag{50}$$

Sada je gradijent funkcije greške *E* po težinskoj matrici $W^{(1)}$:

$$\frac{\partial E}{\partial W^{(1)}} = (a^{(3)} - y) \cdot \varphi'\left(v^{(3)}\right) \cdot W^{(2)} \cdot \varphi'\left(v^{(2)}\right) \cdot x \tag{51}$$

Posljednji korak u algoritmu propagacije unazad je ažuriranje težinske matrice korišćenjem algoritma gradijentnog spuštanja. Gradijentno spuštanje je iterativni algoritam za pronalaženje minimuma funkcije, u ovom slučaju minimuma funkcije greške. Na slici 24 je prikazan grafik zavisnosti funkcije greške *E* od težinske matrice *W*. Tačka w_0 je tačka lokalnog minimuma.



Slika 24. Zavisnost funkcije greške od težinske matrice [39]

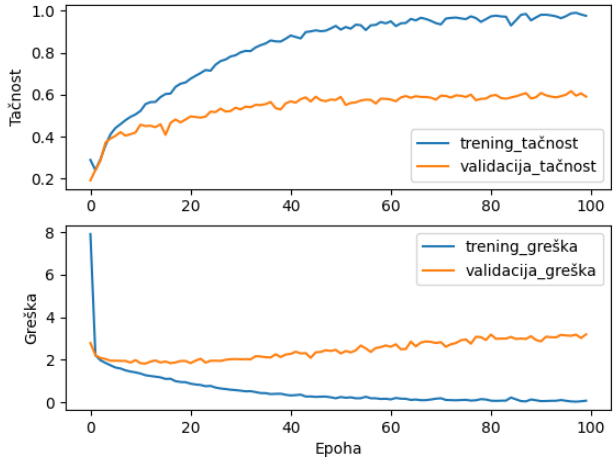
Neka je početna vrijednost težinske matrice $W = w_1$. Gradijent funkcije greške u toj tački je negativan. Dalje povećanje težinske matrice *W* dovodi do smanjenja greške *E*, pa je potrebno preduzeti korak u desnu stranu i povećati *W*. Neka je početna vrijednost težinske matrice $W = w_2$. Gradijent funkcije greške u toj tački je pozitivan. Dalje povećanje težinske matrice *W* dovodi do povećanja greške *E*, pa je potrebno preduzeti korak u lijevu stranu i smanjiti *W*.

Algoritam gradijentnog spuštanja se može definisati i formulom:

$$*W^{(n)} = W^{(n)} - \alpha \frac{\partial E}{\partial W^{(n)}} \ , \tag{52}$$

gdje je $*W^{(n)}$ nova vrijednost težinske matrice, $W^{(n)}$ stara vrijednost težinske matrice, a α je stopa učenja (eng. *learning rate*). Ako je gradijent pozitivan onda je potrebno umanjiti vrijednost težinske matrice, a ako je gradijent negativan onda je potrebno uvećati vrijednost težinske matrice. Stopa učenja ne smije biti previše mala vrijednost, jer će traženje minimuma funkcije trajati dugo. Sa druge strane, ukoliko je stopa učenja prevelika vrijednost, može se desiti da algoritam „preskoči" minimum. Takođe, stopu učenja je potrebno prilagoditi tako da algoritam gradijentnog spuštanja pronađe globalni, a ne lokalni minimum funkcije greške.

Čest slučaj prilikom treniranja neuralne mreže je preveliko prilagođavanje trening podacima ili pretreniranost neuralne mreže (eng. *overfitting*). „*Pretreniranje se odnosi na situaciju u kojoj model savršeno nauči da vrši predikciju za instance iz trening seta, ali ima veoma slabu sposobnost predikcije za instance koje se i malo razlikuju od naučenih*“ [40]. Na slici 25 je prikazana zavisnost tačnosti i greške neuralne mreže od broja epoha, prilikom treninga i validacije. Kada svi ulazni podaci prođu kroz neuralnu mrežu, završena je jedna epoha.



Slika 25. Problem pretreniranosti neuralne mreže

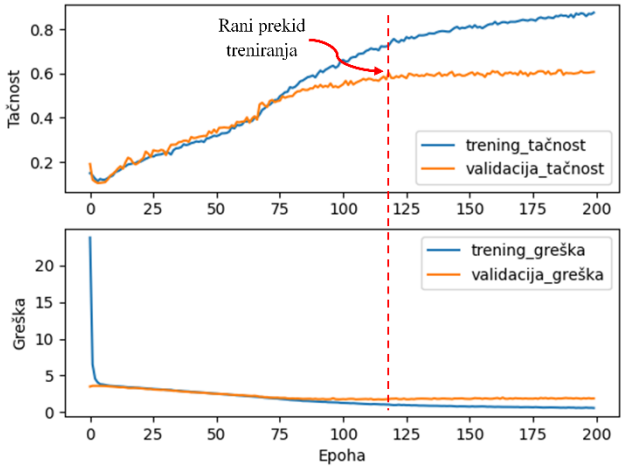
Suprotno od pretreniranosti neuralne mreže je podtreniranost neuralne mreže (eng. *underfitting*). Podtreniranost je scenario u kom neuralna mreža nije u stanju da formira odnos između ulaznih i izlaznih podataka, stvarajući veliku grešku kako na skupu podataka za trening, tako i na skupu podataka za testiranje.

Problem pretreniranosti neuralne mreže se može riješiti na dva načina: proširenjem skupa podataka i regularizacijom. Iako je proširenje skupa podataka najpoželjnija metoda, to često nije moguće. U rijetkim slučajevima kada je to moguće, ako je neuralna mreža previše kompleksna, može ponovo doći do pretreniranosti. Regularizacija ubraja sljedeće metode: jednostavnija arhitektura, vještačko generisanje podataka, rani prekid treniranja, *dropout*, λ_1 regularizacija, λ_2 regularizacija, itd.

Jednostavnija arhitektura podrazumijeva smanjenje broja skrivenih slojeva, kao i broja neurona u slojevima. Ne postoje definisana pravila, već se rješenje dobija eksperimentalnim putem. Međutim, kompleksne mreže imaju potencijal da budu uspješnije od jednostavnih mreža, pa se ova metoda rijetko koristi.

Vještačko generisanje podataka (eng. *data augmentation*) je efikasno sredstvo za smanjenje pretreniranosti neuralne mreže. Nad trening podacima se primjenjuju transformacije i ti podaci se dodaju u skup podataka za treniranje. Za slučaj audio signala moguće transformacije su: promjena trajanja audio signala bez promjene visine tona (eng. *time stretching*), promjena visine tona bez promjene trajanja audio signala (eng. *pitch scaling*), dodavanje šuma, primjena filtara u frekvencijskom domenu, nasumična promjena amplitude zvučnog signala (eng. *random gain*), itd. Bitno je voditi računa da se ne pretjera sa ovom metodom.

Rani prekid treniranja je vjerovatno najčešće korišćena metoda regularizacije neuralnih mreža, zbog svoje efikasnosti i jednostavnosti. Ova metoda detektuje trenutak kada neuralna mreža ulazi u pretreniranje (slika 26). Rani prekid treniranja se može koristiti samostalno ili u kombinaciji sa drugom metodom regularizacije.



Slika 26. Rani prekid treniranja

Dropout metoda se zasniva na proizvoljnom uklanjanju neurona iz skrivenih slojeva neuralne mreže prilikom treniranja. Na taj način se izbjegava prevelika zavisnost neuralne mreže od pojedinih neurona. Neuralna mreža će biti istrenirana tako da svim neuronima pruži jednaku mogućnost odlučivanja. Procenat uklonjenih neurona se definiše parametrom *dropout probability* koji se kreće od 0.1 do 0.5. *Dropout* se primjenjuje prilikom treniranja, dok se prilikom testiranja koristi mreža sa svim neuronima.

Prilikom treniranja, neuralna mreža podešava težine koristeći metod gradijentnog spuštanja. Nakon određenog vremena, težine će postati specijalizovane za trening podatke i njihova vrijednost raste. Samim tim, manje varijacije ili šum na ulazu će rezultirati velikim razlikama na izlazu. Velike težine su znak da je neuralna mreža previše specijalizovana za trening podatke, što je čini nestabilnom.

Algoritam treniranja se može modifikovati tako da podstakne neuralnu mrežu da koristi male težine. Jedan od načina je da se promijeni proračun greške dodavanjem trenutne vrijednosti svih težina u neuralnoj mreži. Na taj način se neuralna mreža „kažnjava“ proporcionalno veličini težina. Veće težine rezultiraju većom kaznom u vidu veće vrijednosti greške. Algoritam treniranja će podstaći neuralnu mrežu da ima manje vrijednosti težina.

U zavisnosti od dodatnog člana u formuli računanja greške razlikuju se λ_1 regularizacija i λ_2 regularizacija [36]. Kod λ_1 regularizacije dodatni član je suma apsolutnih vrijednosti težina:

$$E(p,y) = \frac{1}{2}(p-y)^2 + \lambda_1 \sum |W_i|, \tag{53}$$

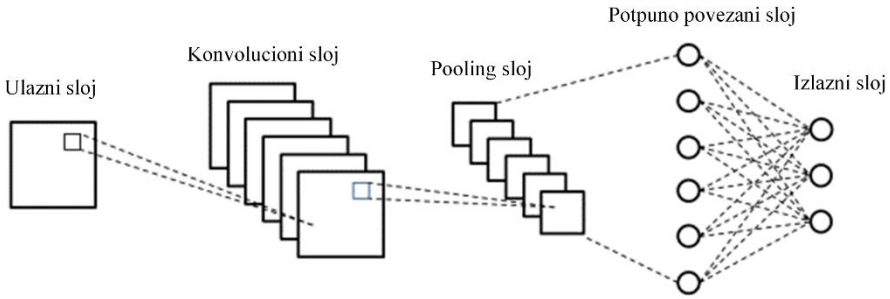
dok je kod λ_2 regularizacije dodatni član suma kvadratnih vrijednosti težina:

$$E(p,y) = \frac{1}{2}(p-y)^2 + \lambda_2 \sum {W_i}^2, \tag{54}$$

gdje su λ_1 i λ_2 parametri regularizacije. U oba izraza efekat regularizacije je smanjenje težina neuralne mreže, ali način na koji se težine smanjuju je drugačiji. U λ_1 regularizaciji, težine se smanjuju za konstantan iznos prema 0. U λ_2 regularizaciji, težine se smanjuju za količinu koja je proporcionalna težini. Kada težina ima veliku vrijednost, λ_1 regularizacija smanjuje težinu znatno manje od λ_2 regularizacije. Nasuprot tome, kada je težina mala, λ_1 regularizacija smanjuje težinu mnogo više od λ_2 regularizacije. Za razliku od λ_1 regularizacije, λ_2 regularizacija je osjetljiva na autlajere.

4.3 Konvolucione neuralne mreže

„*Konvolucione neuralne mreže su neuralne mreže koje koriste konvoluciju umjesto opšteg matričnog množenja u najmanje jednom od njihovih slojeva*“ [41]. Sastoje se od jednog ili više konvolucionih i *pooling* slojeva, praćenim sa jednim ili više potpuno povezanih slojeva, kao i izlaznog sloja (slika 27). Konvolucioni i *pooling* sloj vrše ekstrahovanje karakteristika, dok potpuno povezani sloj vrši klasifikaciju. CNN se najčešće primjenjuju na slike, tj. matrice piksela koje su predstavljene svojom širinom, visinom i vrijednostima piksela. Detekcija zvuka motorne testere se preko vizuelnih reprezentacija zvuka može svesti na klasifikaciju slika korišćenjem CNN-a.



Slika 27. Arhitektura konvolucione neuralne mreže [42]

Konvolucioni sloj sastoji se od konvolucionih kernela ili filtara koji se koriste za računanje različitih mapa karakteristika (eng. *features maps*). Mape karakteristika dobijaju se konvolucijom između ulazne slike ili mape karakteristika prethodnog sloja i kernela. Na rezultat konvolucije se zatim primjenjuje aktivaciona funkcija. Kernel se može opisati kao matrica težinskih koeficijenata, koja predstavlja detektor karakteristika. Neki od kernela su prikazani na slici 28:

Detektor dijagonale

1	0	0
0	1	0
0	0	1

Detektor horizontale

0	0	0
1	1	1
0	0	0

Detektor vertikale

0	1	0
0	1	0
0	1	0

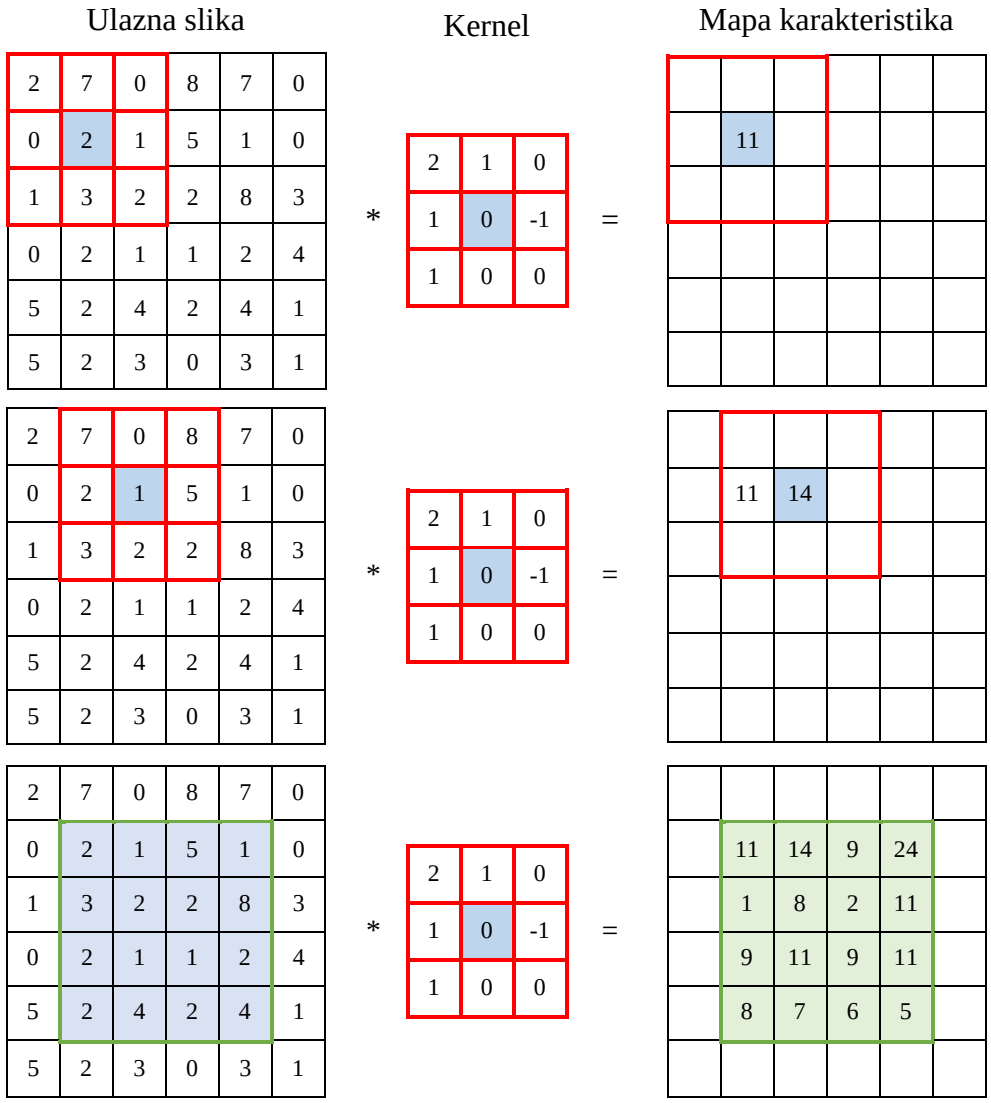
Slika 28. Vrste kernela

Međutim, tokom treniranja CNN modela ne koriste se predefinisani kerneli, već se težinskim koeficijentima kernela dodjeljuju proizvoljne vrijednosti. Sa svakom epohom treninga, težine se ažuriraju i kernel uči da izdvaja značajne karakteristike. Visina i širina kernela su obično neparni brojevi, zbog centralne vrijednosti koja je potrebna za računanje konvolucije. Za slučaj crno-bijele slike, dubina kernela je jednaka 1. Za sliku u boji, svaki piksel ima tri vrijednosti, pa je dubina kernela jednaka 3.

Broj piksela za koji se kernel pomjera u desnu stranu se naziva horizontalni korak (eng. *stride* - S). Osim horizontalnog koraka postoji i vertikalni korak koji ne mora biti isti kao i horizontalni. Povećanje koraka rezultira smanjenjem dimenzija mape karakteristika. Za razliku od ANN-a, gdje je ulaz u vektorskom formatu, ulaz u CNN je višekanalna slika, i to za RGB sliku trokanalna, a za crno bijelu sliku jednokanalna.

Neka je ulaz u CNN crno bijela slika dimenzija *NxN* piksela, kernel dimenzija *KxK* piksela, pri čemu je $N = 6$, $K = 3$. Pikseli crno bijele slike imaju vrijednost od 0 do 255, pri čemu je 0 potpuno bijela, a 255 potpuno crna boja. Primjenjuje se kernel na sliku i računa se konvolucija pomoću formule (55). Zatim se kernel pomjera u desnu stranu horizontalnim korakom i ponavlja se isti postupak sve dok kernel više ne može da se pomjeri ni horizontalno ni vertikalno. Postupak je prikazan na slici 29.

$$\sum_{i=1}^{P} slika_i \cdot kernel_i = 2 \cdot 2 + 7 \cdot 1 + \cdots + 2 \cdot 0 = 11 \tag{55}$$



Slika 29. Postupak računanja mape karakteristika

Mapa karakteristika je dimenzija 4x4, što dovodi do zaključka da je na ivicama i ćoškovima slike došlo do gubitka informacija. Način na koji se ovaj problem može riješiti je dodavanje piksela oko ivica ulazne slike (eng. *padding*). Vrijednost dodatih piksela se obično postavlja na 0 i to se naziva dopuna nulama (eng. *zero padding*) (slika 30). Dopuna nulama je važna kako bi se informacijama na ivicama ulazne slike dalo više na važnosti, u suprotnom, karakteristike na ivicama se gube. Broj piksela koji je potrebno dodati ulaznoj slici da bi dimenzije ostale nepromijenjene je $P = \frac{K-1}{2}$.

0	0	0	0	0	0	0	0
0	2	7	0	8	7	0	0
0	0	2	1	5	1	0	0
0	1	3	2	2	8	3	0
0	0	2	1	1	2	4	0
0	5	2	4	2	4	1	0
0	5	2	3	0	3	1	0
0	0	0	0	0	0	0	0

*

2	1	0
1	0	-1
1	0	0

=

-7	2	1	-6	13	8
0	11	14	9	24	23
-3	1	8	2	11	12
-1	9	11	9	11	25
-2	8	7	6	5	15
3	14	10	10	9	8

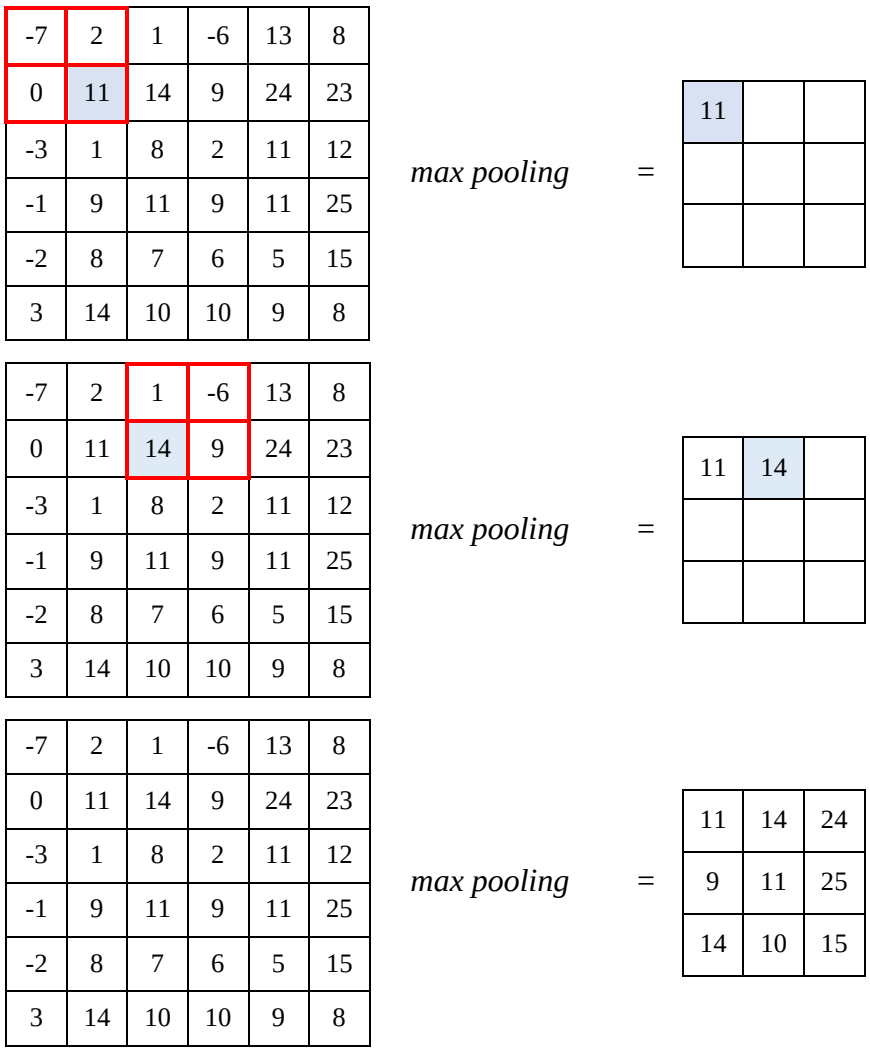
Slika 30. Dopuna nulama

Dimenzije mape karakteristika su:

$$\left[\frac{N+2P-K}{S}+1, \frac{N+2P-K}{S}+1\right] \tag{56}$$

Pooling slojevi smanjuju dimenzije mape karakteristika poduzorkovanjem (eng. *downsampling*). Međutim, najdominantnije karakteristike će biti sačuvane. Postoje različite vrste *pooling*-a, ali se najčešće koriste *average* i *max pooling*. Kod *pooling* operacije potrebno je definisati veličinu *pooling* prozora i korak, slično kao kod konvolucije. *Average pooling* računa prosječnu, a *max pooling* maksimalnu vrijednost iz regiona mape karakteristika određenog *pooling* prozorom. Neka su dimenzije *pooling* prozora *MxM* piksela, a korak *S*, gdje je $M = 2$, $S = 2$. Primjena *max pooling* prozora na mapu karakteristika prikazana je na slici 31.

Nakon jednog ili više konvolucionih i *pooling* slojeva, rezultat se transformiše u jednodimenzioni niz, a zatim slijedi jedan ili više potpuno povezanih slojeva. U poređenju sa ANN, CNN imaju mnogo manje parametara, što ih čini lakšim za treniranje. Danas se CNN koriste kao alat za postizanje obećavajućih rezultata u oblastima kompjuterskog vida, kao što su klasifikacija slika, detekcija objekata, detekcija lica, prepoznavanje izraza lica, prepoznavanje vozila, prepoznavanje teksta, itd. Takođe, CNN se mogu primjenjivati i na vizuelne reprezentacije audio signala, poput spektrograma, mel spektrograma i MFCC-a.



Slika 31. Primjena *max pooling* prozora na mapu karakteristika

5 IMPLEMENTACIJA SISTEMA

U ovom poglavlju je opisana implementacija sistema za detekciju sječe šuma na osnovu metoda dubokog učenja. Detekcija se vrši na osnovu uzoraka zvuka iz dva prikupljena skupa podataka. Prvi skup čine audio signali preuzeti sa javnih platformi, dok drugi skup čine vlastiti audio signali snimljeni u realnim uslovima. Predloženi modeli će biti trenirani, validirani i testirani koristeći ove skupove podataka. U radu će se analizirati performanse dva metoda dubokog učenja: potpuno povezana neuralna mreža i CNN. Takođe, predloženi sistem će biti testiran u realnim uslovima. Konačnu mjeru validnosti predloženih metoda detekcije zvuka motorne testere će dati razvijeni uređaj – mikroprocesorska platforma, na kojoj će ove metode biti implementirane.

5.1 Skupovi podataka

Skup podataka 1 čine audio signali dužine trajanja 30 sekundi u *wave audio* formatu (*.wav). Programi koji su korišćeni za pripremu audio signala su Audacity [43] i WavePad Sound Editor [44]. Audio signali su podijeljeni na djelove za trening, validaciju i testiranje. Tokom pripreme ovog skupa podataka, svaki audio signal se dijeli na 10 segmenata dužine trajanja 3 sekunde. Samim tim, nije ispravno koristiti strategiju podjele skupa podataka na trening i test skupove koja može dovesti do scenarija gdje se jedan segment audio signala koristi za treniranje, a već sljedeći koji je gotovo identičan kao i prethodni, za testiranje modela. Na ovaj način bi tačnost neuralne mreže bila drastično veća, jer je mreža već vidjela sličan audio segment. Podaci koji se koriste za trening, validaciju i testiranje dolaze iz različitih izvora.

Zatim se trening, test i validacioni podaci dijele u tri kategorije. Prva kategorija se odnosi na zvukove okoline, kao što su: vjetar, kiša, grmljavina, snijeg, rijeka, talasi jezera, ptice, ostale životinje koje su prisutne u šumama, hod po šumi, insekti, itd. Druga kategorija se odnosi na zvuk motorne testere, raznih proizvođača, u različitim fazama rada (uključivanje motorne testere, rad na leru, sječa drveta, gašenje motorne testere), kao i na različitim udaljenostima. Takođe, ova kategorija je proširena audio signalima koji kombinuju zvukove motorne testere i zvukove okoline (npr. motorna testera i grmljavina, motorna testera i kiša, motorna testera i vjetar, itd.). Treća kategorija se odnosi na zvukove koji su veoma slični motornoj testeri: motorne sanke, motocikli (dvotaktni i četvorotaktni), trimer za travu i kosačica. Ova kategorija je takođe proširena audio signalima koji predstavljaju kombinaciju zvukova veoma sličnih motornoj testeri i zvukova okoline. Sve tri kategorije su približno jednake veličine, kako bi ih metode treniranja smatrale ravnopravnim.

Ukupan broj audio signala, u trajanju od 30 sekundi, u skupu podataka 1 je 9897, što znači 98970 ulaznih signala nakon podjele na segmente. Od toga se za treniranje neuralne mreže koristi 59520 signala (60.14%), za validaciju 19260 signala (19.46%), a za testiranje 20190 signala (20.40%). Svi audio signali u skupu podataka 1 su preuzeti sa javne platforme www.youtube.com, a linkovi preko kojih se može pristupiti audio fajlovima su dati u prilogu 1. Ovaj skup podataka korišćen je za treniranje neuralnih mreža i odabir optimalnog modela.

Raznolikost audio signala u skupu podataka 1, kao i sama veličina (48.1 GB ili skoro 100.000 ulaznih signala) će smanjiti šanse za pretreniranje neuralne mreže. Distribucija ulaznih signala u skupu podataka 1 prikazana je tabelom I.

Tabela I. Distribucija ulaznih signala u skupu podataka 1

Kategorija		**Broj ulaza**		
		Trening	*Test*	*Validacija*
Motorna testera		19490	6560	6460
Okolina	Ptice	4600	1340	1380
	Hod	760	270	270
	Vjetar	2850	770	770
	Padavine	2990	1210	1260
	Insekti	1410	600	590
	Noć u šumi	2560	760	710
	Rijeke i jezera	2330	990	520
	Ukupno	17500	5940	5500
Slični	Trimer	4740	1670	1590
	Kosačica	6200	2050	1950
	Motorne sanke	4970	1700	1590
	Motocikli	6620	2270	2170
	Ukupno	22530	7690	7300

Skup podataka 2 se sastoji od snimljenih audio signala u realnim uslovima (slika 32), trajanja 30 sekundi, u *wave audio* formatu (*.wav). Ovaj skup podataka je korišćen za testiranje razvijenog modela u realnim uslovima. Audio signali su podijeljeni u tri kategorije: zvukovi okoline, zvukovi motorne testere i zvukovi slični motornoj testeri. Ukupan broj audio signala je 425, što znači 4250 ulaznih signala nakon podjele na segmente. Distribucija ulaznih podataka u skupu podataka 2 prikazana je tabelom II.

Slika 32. Snimanje audio signala u realnim uslovima

Tabela II. Distribucija ulaznih signala u skupu podataka 2

Kategorija		Broj ulaza
Motorna testera		1240
Okolina	Hod	165
	Vjetar	615
	Kiša i grmljavina	565
	Insekti	365
	Ukupno	1710
Slični	Trimer	700
	Kosačica	600
	Ukupno	1300

Priprema skupova podataka u programskom jeziku Python podrazumijeva ekstrahovanje karakteristika audio signala i njihovo skladištenje u odgovarajućem formatu. Prvi korak prilikom ekstrahovanja karakteristika audio signala je učitavanje potrebnih modula (programski kod 8).

Programski kod 8.

```
import os
import librosa
import math
import deepdish as dd
import numpy as np
```

Librosa i *numpy* modul su objašnjeni u poglavlju 3.3. *OS* modul nudi funkcije za interakciju sa operativnim sistemom, otvaranje foldera, podfoldera i prolazak kroz fajlove. *Math* modul omogućava pristup matematičkim funkcijama. *Deepdish* modul omogućava rad sa *hdf5* fajlovima u koje će se smještati ekstrahovane karakteristike. Kao alternativu moguće je koristiti i *json* modul koji omogućava rad sa *json* fajlovima. Ipak, zbog veličine skupa podataka, prednost je data *hdf5* fajlovima, koji zauzimaju mnogo manje prostora u odnosu na *json* fajlove.

U programu se potom definišu putanje do foldera u kojima se nalaze audio signali za trening, validaciju i test, ime *hdf5* fajlova u koje će se smještati karakteristike audio signala, frekvencija odabiranja i trajanje audio signala. Ukupan broj odbiraka u audio signalu se računa kao proizvod frekvencije odabiranja i trajanja audio signala (programski kod 9).

Programski kod 9.

```
TRAIN_DATASET_PATH = "DATA SET/train"
VALIDATION_DATASET_PATH = "DATA SET/validation"
TEST_DATASET_PATH = "DATA SET/test"

TRAIN_FILE_NAME = " features/train_features.hdf5"
VALIDATION_FILE_NAME = " features/validation_features.hdf5"
TEST_FILE_NAME = " features/test_features.hdf5"

SAMPLE_RATE = 22050
DURATION = 30

SAMPLES_PER_TRACK = SAMPLE_RATE * DURATION
```

Za ekstrahovanje karakteristika audio signala je korišćena funkcija *save_features*, definisana programskim kodom 10. Ulazni argumenti funkcije su putanja do foldera sa audio signalima, ime fajla u koje će se smještati karakteristike audio signala, veličina frejma, dužina skoka i broj segmenata na koje se dijeli audio signal (po *default*-u 10).

Funkcija *save_features*, kreira rječnik *data* koji sadrži tri podniza: *features*, *labels* i *mapping*. Imena podfoldera (*okolina, motorna testera, slični*) se smještaju u podniz *mapping*. Štampa se poruka za korisnika u kom se podfolderu trenutno nalazi. Pomoću *librosa.load* funkcije se vrši učitavanje audio signala. Audio signali se dijele na segmente, i to 10 segmenata dužine trajanja 3s. Broj odbiraka po segmentu je količnik ukupnog broja odbiraka u audio signalu i broja segmenata. Broj vektora sa karakteristikama se računa kao količnik broja odbiraka po segmentu i dužine skoka. Rezultat količnika je decimalni broj, pa je izvršeno zaokruživanje na prvi sljedeći cijeli broj.

Pomoću *for* petlje prolazi se kroz segmente audio signala, određuje se početak i kraj segmenta i računaju se karakteristike za taj segment. Zatim se vrši upisivanje karakteristika u podniz *features*. Podniz *label* označava kom podfolderu pripada audio signal, pa elementi uzimaju vrijednosti 0 za *okolina*, 1 za *motorna testera* ili 2 za *slični*. Nakon prolaska kroz sve audio signale, rječnik *data* se pomoću funkcije *deepdish.io.save* čuva kao *hdf5* fajl.

Programski kod 10.

```
def save_features(dataset_path, file_name, n_fft=2048, hop_length=512,
num_segments=10):

    data = {"mapping": [],"labels": [],"features": []}
    samples_per_segment = int(SAMPLES_PER_TRACK/num_segments)
    num_feature_vectors_per_segment=math.ceil(samples_per_segment/hop_length)

    for i, (dirpath, dirnames, filenames) in enumerate(os.walk(dataset_path)):
        if dirpath is not dataset_path:
            semantic_label = dirpath.split("/")[-1]
            data["mapping"].append(semantic_label)
            print("\nProcessing: {}".format(semantic_label))

            for f in filenames:
                file_path = os.path.join(dirpath, f)
                signal, sample_rate = librosa.load(file_path, sr=SAMPLE_RATE)

                for d in range(num_segments):
                    start = samples_per_segment * d
                    finish = start + samples_per_segment

                   # ekstraktovanje karakteristike audio signala
                    feature = librosa.feature.rms(signal[start:finish],
             sample_rate, n_fft=n_fft, hop_length=hop_length)[0]
                    feature = feature.T

                    if len(feature) == num_ feature_vectors_per_segment:
                        data["features "].append(feature.tolist())
                        data["labels"].append(i - 1)
                        print("{}, segment:{}".format(file_path, d + 1))

    data["features "] = np.array(data["features "], dtype=np.float32)
    data["labels"] = np.array(data["labels"], dtype=np.int32)
    dd.io.save(file_name, data)
```

Ekstrahovanje karakteristika audio signala iz dva skupa podataka rađeno je na računaru konfiguracije: AMD Ryzen 5 5600G 6 cores, RAM 16GB i NVMe SSD SeaGate Baracuda Q5 500GB. Ukupno vrijeme potrebno za ekstrahovanje karakteristika je oko 180 minuta za prvi, a 8 minuta za drugi skup podataka. Veličina *hdf5* fajla zavisi od karakteristika koje se ekstrahuju, ali varira od 200MB za vremenske i frekvencijske karakteristike do 5GB za vremensko-frekvencijske karakteristike.

5.2 Implementacija potpuno povezane neuralne mreže

Implementacija potpuno povezane neuralne mreže u programskom jeziku Python započinje učitavanjem potrebnih modula (programski kod 11).

Programski kod 11.

```
import time, deepdish as dd, numpy as np
import tensorflow.keras as keras, matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
```

Moduli *numpy* i *deepdish* su objašnjeni u poglavljima 3.3 i 5.1. *Time* modul pruža različite funkcije vezane za vrijeme. Modul *keras* se koristi za jednostavnu izgradnju i treniranje neuralnih mreža. *Matplotlib* je modul za vizuelizaciju u Python-u. Iz modula *sklearn,* vjerovatno jednog od najkorisnijih modula za mašinsko učenje u Python-u, učitava se paket *metrics* i funkcije *confusion_matrix* i *confusionMatrixDisplay* za računanje i prikaz matrica konfuzije. Matrica konfuzije je matrica dimenzija *NxN* koja se koristi za procjenu performansi klasifikacionog modela, gde je *N* broj klasa.

Definisana je putanja do *hdf5* fajlova koji sadrže karakteristike audio signala, vrijednost stope učenja, λ_2 parametar regularizacije, *dropout*, *batch size* i broj epoha (programski kod 12).

Programski kod 12.

```
TRAIN_FILE_NAME = "features/train_features.hdf5"
VALIDATION_FILE_NAME = "features/validation_features.hdf5"
TEST_FILE_NAME = "features/test_features.hdf5"
L2=0.00001; DROPOUT=0.2; LEARNING_RATE=0.0001; BATCH_SIZE=256; EPOCHS=1000;
```

Programski kod 13 se odnosi na glavni dio programa:

Programski kod 13.

```
if __name__ == "__main__":
    X_train, X_validation, X_test, y_train, y_validation, y_test, mapping =
    load_data(TRAIN_FILE_NAME, VALIDATION_FILE_NAME, TEST_FILE_NAME);
    input_shape = (X_train.shape[1], 1)
    model = build_model(input_shape)
    optimizer = keras.optimizers.Adam(learning_rate=LEARNING_RATE)
    model.compile(optimizer=optimizer,loss='sparse_categorical_crossentropy',
    metrics= ["accuracy"])
    model.summary()
    start = time.time()
    es_loss = keras.callbacks.EarlyStopping(monitor='val_loss', mode='min',
    verbose=1, patience=5)
    history = model.fit(X_train, y_train, validation_data=(X_validation,
    y_validation), batch_size=BATCH_SIZE, epochs=EPOCHS,callbacks=[es_loss])
```

Programski kod 13.(nastavak)

```
end = time.time()
plot_history(history)
model.save('model')
print('\nVrijeme:', end-start)

test_loss, test_acc = model.evaluate(X_test, y_test, verbose=2)
prediction = model.predict(X_test);
y_prediction = prediction.argmax(axis=1);
cm = confusion_matrix(y_test, y_prediction);
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=mapping)
disp.plot()
plt.show()
```

Funkcija *load_data*, definisana programskim kodom 14, učitava podatke za treniranje, validaciju i testiranje. Ulazni argumenti funkcije su putanje do *hdf5* fajlova gdje su smještene karakteristike audio signala.

Programski kod 14.

```
def load_data(train_file_name, validation_file_name, test_file_name):
    train_data = dd.io.load(train_file_name)
    validation_data = dd.io.load(validation_file_name)
    test_data = dd.io.load(test_file_name)

    train_inputs = (np.array(train_data["features"]))
    validation_inputs = (np.array(validation_data["features"]))
    test_inputs = (np.array(test_data["features"]))

    train_targets = np.array(train_data["labels"])
    validation_targets = np.array(validation_data["labels"])
    test_targets = np.array(test_data["labels"])

    mapping = np.array(train_data["mapping"])
    return train_inputs, validation_inputs, test_inputs, train_targets,
    validation_targets, test_targets, mapping
```

Dimenzije ulaznih podataka su [*num_features* * *num_feature_vectors_per_segment*, 1], pri čemu se *num_features* odnosi na broj karakteristika koje se koriste, a *num_ feature_vectors_per_segment* na broj vektor karakteristika za jedan segment. Kreiranje neuralne mreže je izvršeno pomoću funkcije *build_model* kojoj se prosljeđuje oblik ulaznih podataka (programski kod 15).

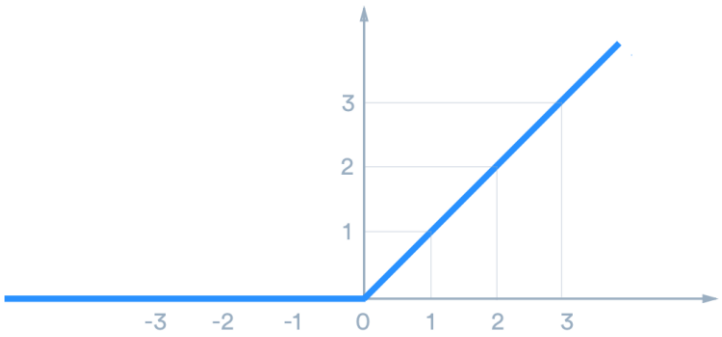
Predložena arhitektura potpuno povezane neuralne mreže se sastoji od ulaznog sloja, tri skrivena sloja i jednog izlaznog sloja. Broj neurona po skrivenim slojevima je 512, 256 i 64. *Sequential* model neuralne mreže iz *keras* biblioteke spaja sve slojeve jedan za drugim. Ulazni sloj je tipa *Flatten*, koji će izravniti ulazne podatke i pretvoriti ih u vektor. Skriveni slojevi su tipa *Dense*, koji ulazne podatke množe sa težinskom matricom, vrše aktivaciju i rezultat prosljeđuju narednom sloju. U cilju sprečavanja pretreniranja neuralne mreže korišćene su metode *dropout*-a, λ_2 regularizacije i rani prekid treniranja. Kao aktivaciona funkcija je korišćena ReLU (eng. *Rectified Linear activation Unit*) funkcija, definisana formulom:

$$\varphi(v) = \begin{cases} 0, v < 0 \\ v, v \geq 0 \end{cases} \tag{57}$$

Programski kod 15.

```
def build_model(input_shape):
    model = keras.Sequential([
        keras.layers.Flatten(input_shape=input_shape),
        keras.layers.Dense(512, activation="relu",
        kernel_regularizer=keras.regularizers.l2(L2)),
        keras.layers.Dropout(DROPOUT),

        keras.layers.Dense(256, activation="relu",
        kernel_regularizer=keras.regularizers.l2(L2)),
        keras.layers.Dropout(DROPOUT),

        keras.layers.Dense(64, activation="relu",
        kernel_regularizer=keras.regularizers.l2(L2)),
        keras.layers.Dropout(DROPOUT),

        keras.layers.Dense(3, activation="softmax")])
    return model
```

ReLU funkcija, prikazana na slici 33, omogućava brže treniranje neuralne mreže u odnosu na sigmoid funkciju. Takođe, smanjena je vjerovatnoća pojave iščezavajućeg gradijenta. Algoritam propagacije unazad objašnjen u poglavlju 4.2, podrazumijeva računanje gradijenta funkcije greške nad težinskom matricom. U formulama (44) i (51) je izvršeno množenje sa izvodom aktivacione funkcije. Maksimalna vrijednost izvoda sigmoid funkcije je 0.25 [46]. Ukoliko neuralna mreža ima veliki broj slojeva, uzastopnim množenjem sa 0.25 dobiće se gradijent koji je jednak ili približno jednak 0. Ta pojava se naziva iščezavajući gradijent i može se eliminisati korišćenjem ReLU funkcije.



Slika 33. ReLU funkcija [45]

Izlazni sloj sadrži tri neurona, koji odgovaraju broju kategorija audio signala: motorna testera, lažno pozitivni i okolina. Aktivaciona funkcija izlaznog sloja je *softmax*. *Softmax* funkcija pretvara vektor od *K* stvarnih vrijednosti u vektor od *K* vjerovatnoća čiji je zbir jednak 1:

$$\varphi(v) = \frac{e^{v}}{\sum_{j=1}^{K} e^{v_j}} \tag{58}$$

Adam algoritam (eng. *Adaptive Moment Estimation*) se koristi za optimizaciju težina neuralne mreže, i predstavlja proširenje algoritma gradijentnog spuštanja. Za razliku od algoritma gradijentnog spuštanja koji održava jednu te istu stopu učenja, *Adam* algoritam ažurira stopu učenja za svaku težinu neuralne mreže pojedinačno. Preporučuje se kao podrazumijevani algoritam optimizacije, jer je jednostavan za implementaciju, ima brže vrijeme treniranja, niske zahtjeve za memorijom i zahtijeva manje podešavanja nego bilo koji drugi algoritam optimizacije [47].

Sparse categorical crossentropy (SCC) je korišćena kao funkcija greške, koja računa unakrsnu entropiju između stvarnih izlaza i predikcija. Za razliku od *categorical crossentropy* kod koje su izlazi numerisani binarno, kod SCC izlazi su cijeli brojevi (0, 1 ili 2).

Implementiran je rani prekid treniranja, *callback* funkcijom iz *keras* biblioteke sa nazivom *EarlyStopping*, koji zaustavlja trening neuralne mreže ako se greška validacije ne smanji nakon 5 epoha. Pomoću *time.time()* funkcije u *start* se smješta vrijeme početka treniranja neuralne mreže. Slijedi treniranje neuralne mreže pomoću *model.fit* funkcije. Funkciji se prosleđuju trening podaci, validacioni podaci, *batch_size*, broj epoha i *callback* funkcija. *Batch size* je broj ulaza nakon kojih se vrši ažuriranje težinske matrice. Informacije o tačnosti i grešci treninga i validacije se smještaju u promjenjivu *history*. U *stop* se smješta vrijeme završetka treniranja neuralne mreže. Vizualizacija tačnosti i greške vrši se pomoću funkcije *plot_history*, definisane programskim kodom 16.

Programski kod 16.

```
def plot_history(history):
    fig, axs = plt.subplots(2)

    axs[0].plot(history.history["accuracy"], label="Trening")
    axs[0].plot(history.history["val_accuracy"], label="Validacija")
    axs[0].set_ylabel("Tačnost")
    axs[0].legend(loc="lower right")

    axs[1].plot(history.history["loss"], label="Trening")
    axs[1].plot(history.history["val_loss"], label="Validacija")
    axs[1].set_ylabel("Greška")
    axs[1].set_xlabel("Epoha")
    axs[1].legend(loc="upper right")

    plt.show()
```

5.3 Implementacija konvolucione neuralne mreže

Za implementaciju CNN-a u Python-u mogu se iskoristiti programski kodovi navedeni u 5.2, sa modifikacijom funkcije *build_model* (programski kod 17). Predložena CNN arhitektura se sastoji od tri konvoluciona sloja, tri *pooling* sloja, jednog potpuno povezanog sloja i jednog izlaznog sloja. *Sequential* model neuralne mreže iz *keras* biblioteke spaja sve slojeve jedan za drugim. Tip konvolucionih slojeva je Conv2D, što znači da se kernel prilikom konvolucije pomjera po dvijema osama. Broj kernela u konvolucionim slojevima je 32, dimenzije kernela su 3x3 u prva dva, tj. 2x2 u trećem konvolucionom sloju, vertikalni i horizontalni korak je 1, a aktivaciona funkcija je ReLU.

Tip *pooling* slojeva je MaxPooling2D, što znači da se *max pooling* prozor pomjera po dvijema osama. Dimenzije *max pooling* prozora su 3x3 u prva dva, tj. 2x2 u trećem *pooling* sloju, a vertikalni i horizontalni korak je 2. Argument *padding = 'same'* znači da se dopuna nulama dodaje po potrebi kako bi se nadoknadila preklapanja kada se veličina ulaza i veličina kernela ne uklapaju.

Slojevi tipa *BatchNormalization* iz *keras* biblioteke normalizuju svoje ulaze. Sloj primjenjuje transformaciju koja održava izlaz blizu 0 što ubrzava treniranje neuralne mreže. Da bi se spriječilo pretreniranje konvolucione neuralne mreže korišćene su metode *dropout* i rani prekid treniranja. U posljednja dva potpuno povezana sloja, broj neurona je 64 i 3. Aktivaciona funkcija u izlaznom sloju je *softmax* koja će kao izlaz dati vektor vjerovatnoća.

Programski kod 17.

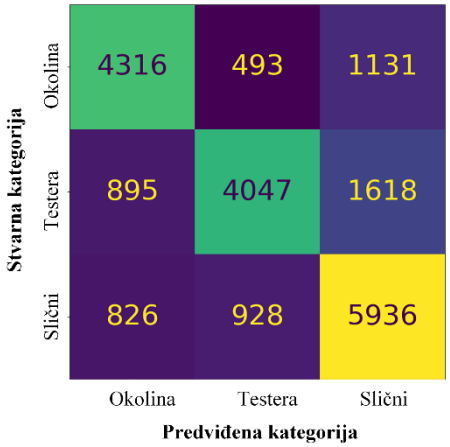
```
def build_model(input_shape):

    model = keras.Sequential()
    model.add(keras.layers.Conv2D(32, (3, 3), strides= (1,1),
    activation='relu', input_shape=input_shape))
    model.add(keras.layers.MaxPooling2D((3, 3), strides=(2, 2),
    padding='same'));
    model.add(keras.layers.BatchNormalization())

    model.add(keras.layers.Conv2D(32, (3, 3), strides= (1,1),
    activation='relu'))
    model.add(keras.layers.MaxPooling2D((3, 3), strides=(2, 2),
    padding='same'));

    model.add(keras.layers.BatchNormalization())

    model.add(keras.layers.Conv2D(32, (2, 2), strides= (1,1),
    activation='relu'))
    model.add(keras.layers.MaxPooling2D((2, 2), strides=(2, 2),
    padding='same'));

    model.add(keras.layers.BatchNormalization())

    model.add(keras.layers.Flatten())

    model.add(keras.layers.Dense(64, activation='relu'))
    model.add(keras.layers.Dropout(DROPOUT))

    model.add(keras.layers.Dense(3, activation='softmax'))

    return model
```

5.4 Rezultati

Prilikom izbora optimalnog modela uzete su u obzir tačnost i greška koju je model postigao na test podacima iz skupa podataka 1. Tačnost je procenat dobijenih tačnih predviđanja, a greška je vrijednost koja ukazuje na rastojanje od željenog stanja. Ukoliko se predviđanje modela približava željenom stanju, to neće uticati na tačnost, ali će se greška smanjiti. Model postaje sve robusniji i bolji. Sa druge strane, jedno pogrešno predviđanje dovodi do značajne razlike u vrijednostima greške bez veće promjene u vrijednostima tačnosti. Za neuravnotežene skupove podataka, gdje jedna klasa čini veći dio skupa podataka, tačnost bi generalno bila visoka. U takvim scenarijima, nije dovoljno uzeti u obzir samo tačnost modela. U slučaju dva različita modela sa istim rezultatom tačnosti, prednost je data onom koji ima manju vrijednost greške.

Na ulaz potpuno povezane neuralne mreže se dovode vremenske i frekvencijske karakteristike: AE, ZCR, RMSE, SC i SB. Karakteristike su nadovezane jedna na drugu i normalizovane skaliranjem između -1 i 1. Na ovaj način, sve karakteristike će biti podjednako uzete u obzir prilikom treniranja neuralne mreže. Parametri koji se mijenjaju prilikom treniranja potpuno povezane neuralne mreže su: stopa učenja, *dropout*, λ_2 parametar regularizacije i *batch size*. Stopa učenja uzima vrijednosti između 1e-5 i 1e-2, pri čemu se nova vrijednost dobija množenjem prethodne sa 1e1. *Dropout* uzima vrijednosti 0.1, 0.2 i 0.3. Parametar regularizacije λ_2 uzima vrijednosti od 1e-7 do 1e-2, pri čemu se nova vrijednost dobija množenjem prethodne sa 1e1. *Batch size* uzima vrijednosti iz skupa elemenata [8, 16, 32, 64, 128, 256, 512]. Broj epoha zavisi od trenutka u kom se izvrši rani prekid treniranja.

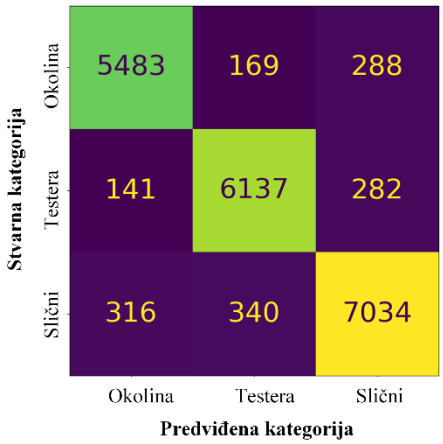
Najbolji rezultat potpuno povezane neuralne mreže na testnim podacima skupa podataka 1 postignut je za vrijednosti: stopa učenja = 1e-4, *batch size* = 512, *dropout* = 0.2 i λ_2 = 1e-7. Od ukupno 20190 ulaznih podataka, njih 14299 ili oko 71% je ispravno klasifikovano. Rani prekid treniranja je izvršen nakon 22 epohe, a vrijeme treniranja je iznosilo 22.57 sekundi. Slika 34 prikazuje matricu konfuzije potpuno povezane neuralne mreže.



Slika 34. Matrica konfuzije potpuno povezane neuralne mreže

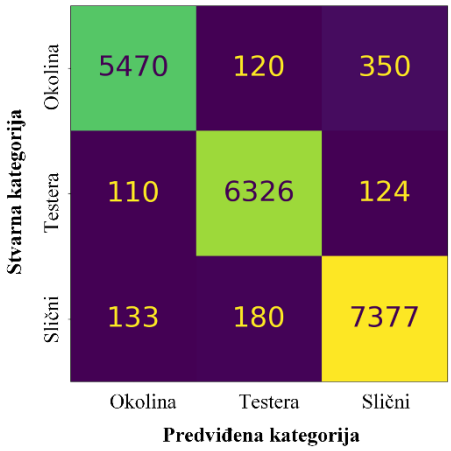
Ulazi CNN modela su vremensko-frekvencijske karakteristike: MFCC i mel sprektrogram. Broj MFCC-eva testiranih u ovom radu je 13, 26 i 39. Broj *mel bands* u mel spektrogramu koji je testiran u ovom radu je 40, 80 i 130. Varijabilni parametri u treningu CNN-a su: stopa učenja, *dropout* i *batch size*. Stopa učenja uzima vrijednosti između 1e-5 i 1e-2, pri čemu se nova vrijednost dobija množenjem prethodne sa 1e1. *Dropout* uzima vrijednosti 0.1, 0.2 i 0.3. *Batch size* uzima vrijednosti iz skupa elemenata [64, 128, 256, 512]. Broj epoha zavisi od ranog prekida treniranja.

Parametri CNN modela sa najboljim rezultatom kada se na ulaz dovode MFCC su: stopa učenja = 1e-4, *batch size* = 256, *dropout* = 0.1 i broj MFCC-eva je 39. Na testnim podacima iz skupa podataka 1, je postignuta tačnost od 92.39%. Rani prekid treniranja je obavljen nakon 14 epoha, a vrijeme treninga je iznosilo 1107.57 sekundi. Slika 35 prikazuje matricu konfuzije CNN-a za MFCC ulaze.



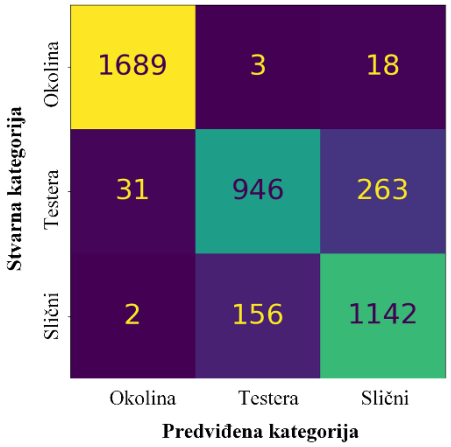
Slika 35. Matrica konfuzije CNN-a za MFCC ulaze

Parametri CNN modela sa najboljim rezultatom kada se mel spektrogram koristi kao ulaz su: stopa učenja = 1e-4, *batch size* = 512, *dropout* = 0.3 i broj *mel bands* = 80. Na testnim podacima iz skupa podataka 1, je postignuta tačnost od 94.96%. Rani prekid treniranja je izvršen nakon 20 epoha, a vrijeme treniranja je iznosilo 3252.61 sekunde. Ovo je takođe model koji je postigao najbolji rezultat na testnim podacima iz skupa podataka 1 i biće izabran kao optimalni model. Slika 36 prikazuje matricu konfuzije za ovu CNN.



Slika 36. Matrica konfuzije CNN-a za mel spektrogram ulaze

Visoka tačnost optimalnog modela na skupu podataka 1 pokazuje da se metode dubokog učenja mogu koristiti za pouzdanu detekciju sječe šuma motornom testerom, čak i u prisustvu zvukova koji su veoma slični motornoj testeri. Međutim, predloženi sistem je potrebno testirati i u realnim uslovima, na snimljenim audio signalima. Optimalni model, treniran na skupu podataka 1, je postigao tačnost od 88.87% na snimljenim audio signalima iz skupa podataka 2. Slika 37 prikazuje matricu konfuzije CNN modela na skupu podataka 2.



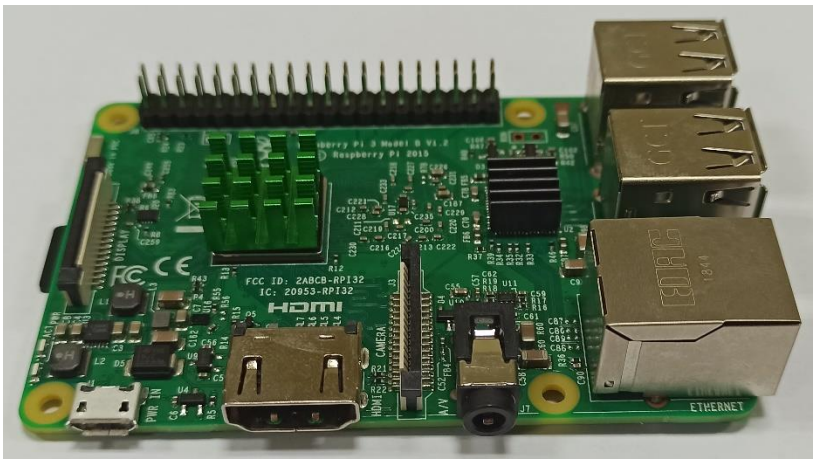
Slika 37. Matrica konfuzije CNN modela na skupu podataka 2

CNN se pokazuju kao efikasnije od potpuno povezanih neuralnih mreža kad je u pitanju detekcija zvuka motorne testere. Razlika u tačnosti detekcije je 24%. Takođe, pokazano je da se sistem za detekciju zvuka motorne testere istreniran u laboratorijskim uslovima može direktno primijeniti u realnim uslovima.

5.5 Hardverska implementacija sistema

Raspberry Pi (RPi) je serija malih računara na jednoj ploči (eng. *Single Board Computer* - SBC) razvijena od strane Raspberry Pi fondacije u saradnji sa Broadcom-om [48]. SBC je kompletan računar izgrađen na jednoj pločici veličine kreditne kartice, sa mikroprocesorom, memorijom, ulazno/izlaznim portovima (I/O) i drugim karakteristikama koje su potrebne za funkcionalan računar [49]. Postoje tri serije Raspberry Pi-a: Raspberry Pi, Raspberry Pi Zero, Raspberry Pi Pico, a svaka od njih ima nekoliko generacija.

U ovom radu je korišćena serija Raspberry Pi 3 model B v1.2 (slika 38), objavljena 2016. godine. Raspberry Pi 3 Model B v1.2 posjeduje 64-bitni četvorojezgarni ARM Cortex-A53 procesor koji radi na frekvenciji 1.2 GHz, RAM memoriju veličine 1GB i dvojezgarni VideoCore IV Multimedia grafički koprocesor. Od interfejsa sadrži 4 USB porta, Ethernet port, HDMI, 3.5 mm audio konektor, DSI Display port, CSI Camera port, 40 GPIO (eng. *General-Purpose Input/Output*) pinova i slot za memorijsku karticu. Kao operativni sistem najčešće se koristi Raspbian, besplatni operativni sistem zasnovan na Debian Linux-u i prilagođen RPi platformi. U ponudi su i drugi operativni sistemi: Pidora, OpenElec, Windows 10 IoT, RaspBMC, RISC OS i Arch Linux. Raspberry Pi 3 model B v1.2 posjeduje WiFi i *Bluetooth* konektivnost. Za komunikaciju između korisnika i RPi-a je potreban HDMI kabal, monitor, tastatura i miš. Osim ovog standardnog načina za komunikaciju, moguće je i povezivanje RPi-a sa računarom putem Ethernet-a. Više o tome se može pronaći na [50].



Slika 38. Raspberry Pi 3 model B v1.2

MicroSD memorijska kartica kapaciteta 32GB je korišćena za instalaciju operativnog sistema i skladištenje podataka (slika 39). Za Raspberry Pi 3 model B v1.2 je potrebno da memorijska kartica bude kapaciteta između 8 i 32 GB.

Slika 39. MicroSD memorijska kartica

Električno napajanje (slika 40) transformiše ulazni naizmjenični napon od 220V u jednosmjerni napon od 5V, sa maksimalnom strujom od 2.5A.

Slika 40. Električno napajanje

Kućište (slika 41) pruža zaštitu RPi uređaja od spoljašnjih uticaja, prvenstveno mehaničkih oštećenja. Takođe, obezbjeđuje i jednostavan pristup svim konektorima.

Slika 41. Kućište za RPi

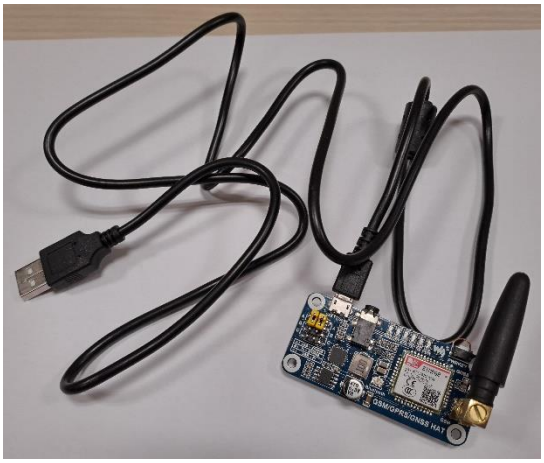
Acme MK-200 mikrofon (slika 42), korišćen za snimanje zvuka, ima frekvencijski opseg od 20Hz do 16kHz i osjetljivost od -58dB.

Slika 42. Acme MK-200 mikrofon

Povezivanje Acme MK-200 mikrofona se vrši posredstvom 3.5mm audio konektora. Iako RPi posjeduje odgovarajuću 3.5mm audio utičnicu, ona je namijenjena za audio izlaz. Zbog toga je potrebno koristiti eksternu zvučnu karticu koja omogućava povezivanje Acme MK-200 mikrofona putem USB-a. Na slici 43 je prikazana UGREEN US205 eksterna USB zvučna kartica.

Slika 43. UGREEN US205 eksterna USB zvučna kartica

SIM 868 RPi HAT (eng. *Hardware Attached on Top*) (slika 44) je platforma male snage koji ima višestruke komunikacione funkcije: GSM, GPRS, GNSS i *Bluetooth*. Omogućava RPi-u da uputi telefonski poziv, pošalje poruku, poveže se na mobilni Internet, odredi GPS koordinate, prenese podatke preko *Bluetooth*-a, itd. Za komunikaciju sa RPi-em je moguće koristiti USB ili UART interfejs, a upravljanje se vrši preko AT komandi. SIM 868 RPi HAT sadrži USB to UART konvertor, slot za SIM karticu, 3.5mm audio utičnicu, konektore za GNSS, *Bluetooth* i GSM antenu, LED indikatore, itd. Za povezivanje SIM 868 RPi HAT na mobilni Internet i slanje podataka o detekciji zvuka na *cloud* platformu je potrebna SIM kartica sa poznatim ili uklonjenim PIN-om.

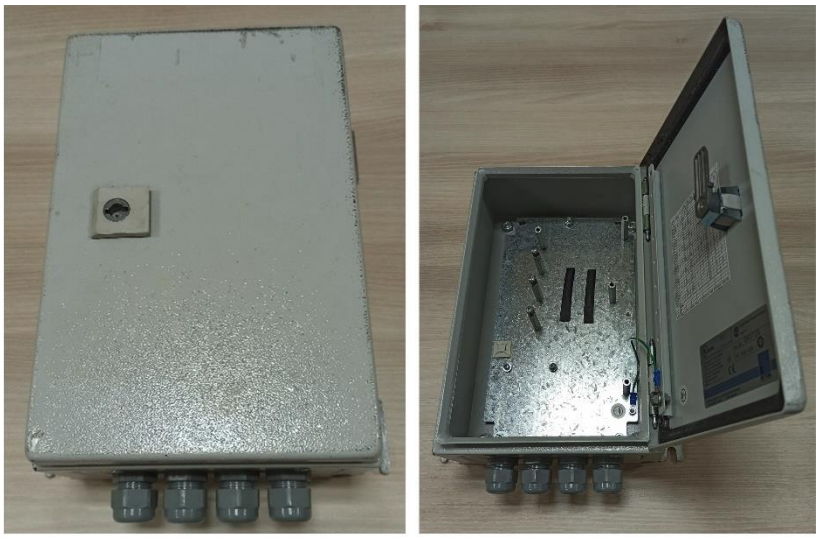


Slika 44. SIM 868 platforma

Svjetleća dioda (LED), otpornik od 220Ω i kablovi **ž/ž** (eng. *jumper cables*), prikazani na slici 45 se koriste za signaliziranje korisniku u kom trenutku se vrši snimanje, kao i da li je detektovan zvuk motorne testere.

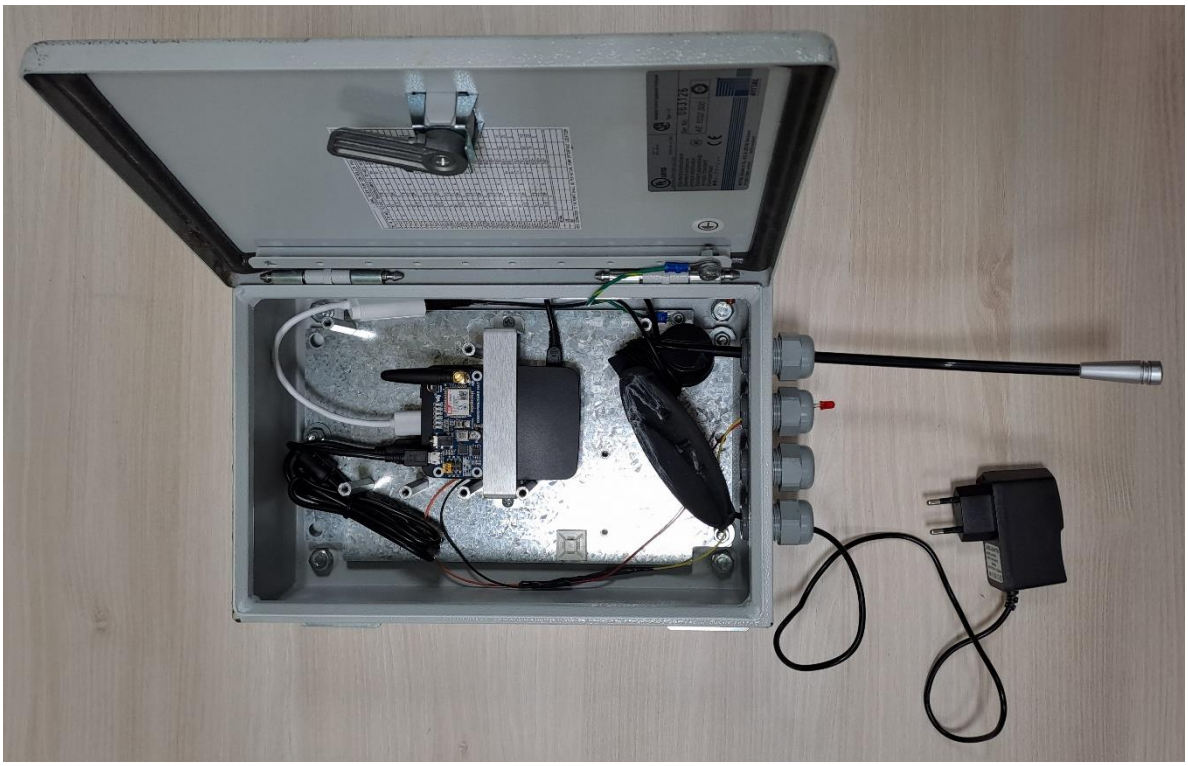
Slika 45. LED, otpornik i kablovi

Vodootporno kućište pruža zaštitu od vremenskih uticaja (slika 46).



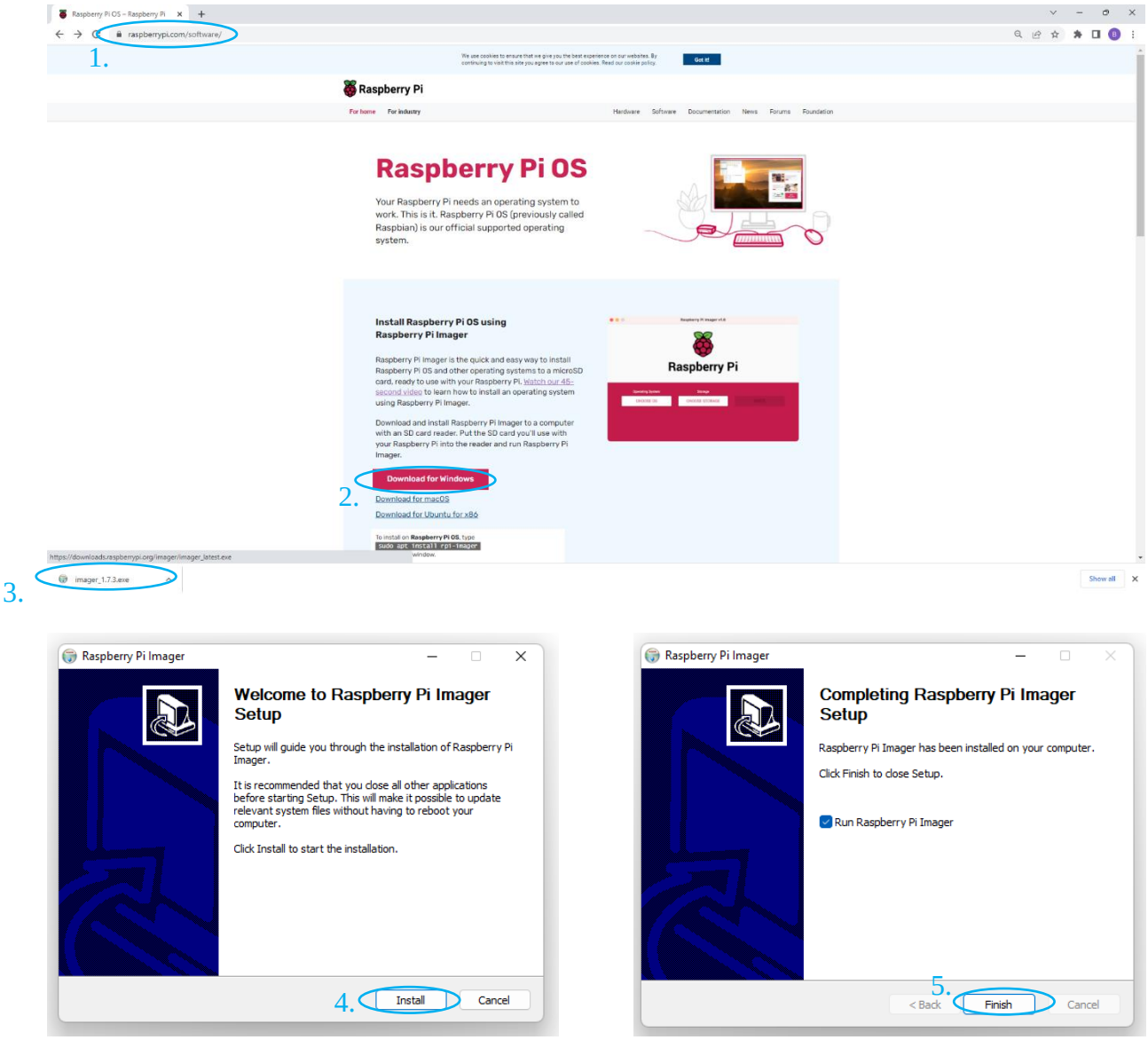
Slika 46. Vodootporno kućište

Na slici 47 je prikazan finalni prototip sistema za detekciju sječe šuma.



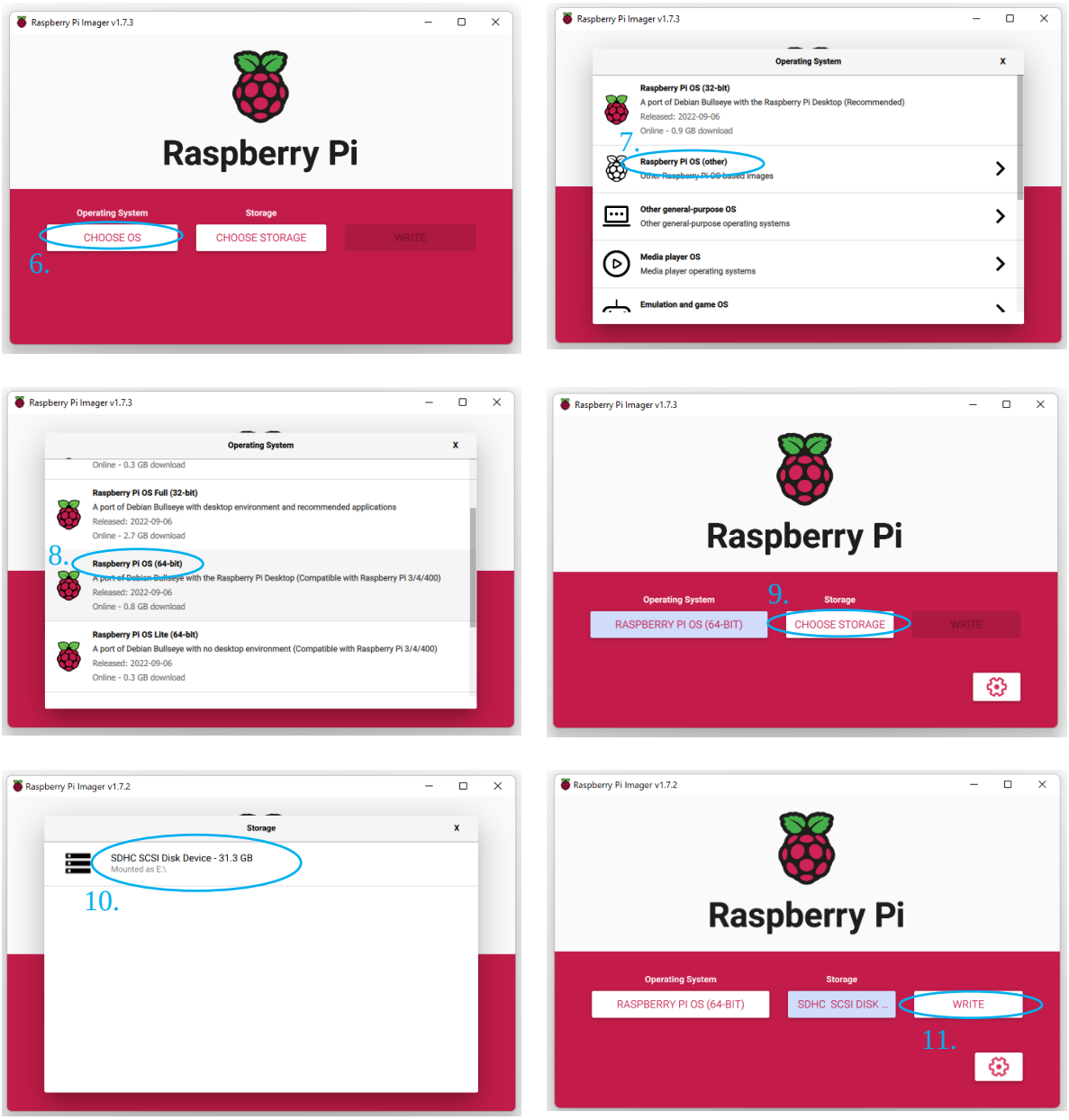
Slika 47. Finalni prototip sistema za detekciju sječe šuma

Prije prvog pokretanja Raspberry Pi-a, neophodno je instalirati operativni sistem. Za ovo je potreban računar, microSD memorijska kartica kapaciteta između 8GB i 32GB, čitač microSD memorijskih kartica i pristup Internetu. Putem računara treba pristupiti *web* stranici https://www.raspberrypi.com/software/ i preuzeti instalacioni fajl za Raspberry Pi Imager klikom na dugme *Download for Windows*. Raspberry Pi Imager je jednostavna aplikacija za instaliranje operativnog sistema na microSD memorijsku karticu. Nakon što je završeno preuzimanje, pokrenuti instalacioni fajl. Postupak je prikazan na slici 48.



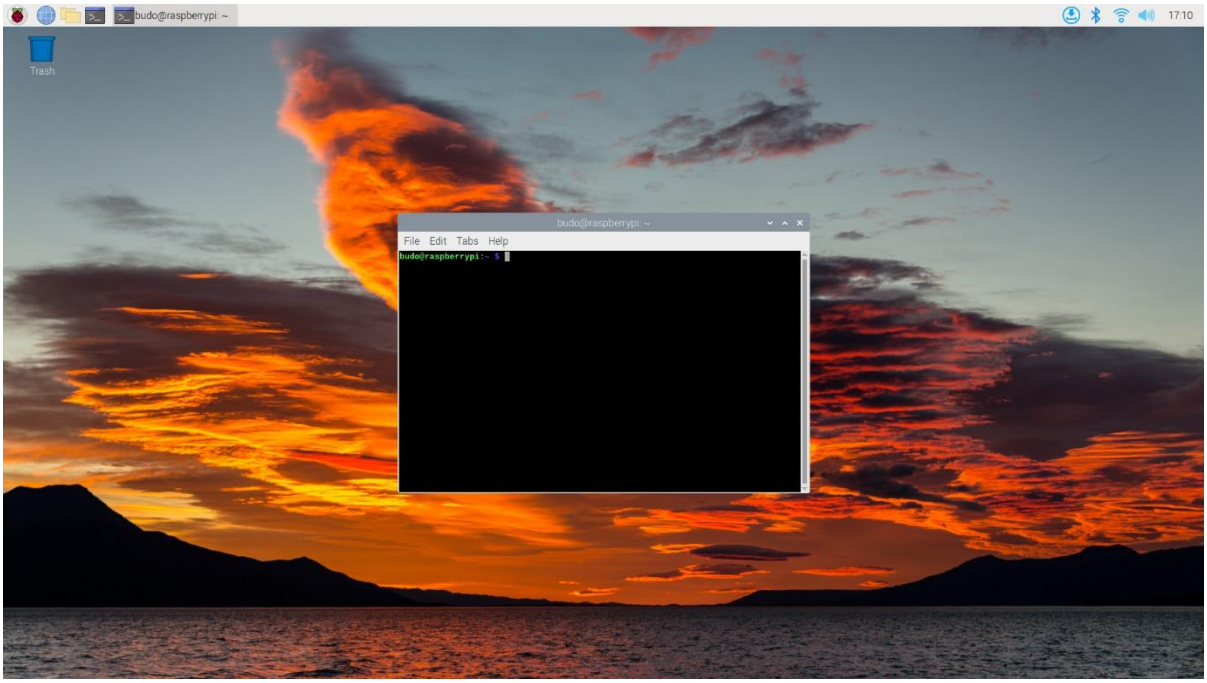
Slika 48. Instalacija aplikacije Raspberry Pi Imager

Potom treba pokrenuti aplikaciju Raspberry Pi Imager, odabrati operativni sistem i microSD memorijsku karticu kao medijum za skladištenje. Klikom na *write* pokrenuće se instalacija operativnog sistema na odabrani medijum. Za potrebe ovog rada instaliran je Raspberry PI OS (Raspbian) 64 bitna verzija, koja je kompatibilna sa Raspberry Pi 3/4/400. Postupak je prikazan na slici 49.



Slika 49. Instalacija operativnog sistema za Raspberry Pi

Memorijsku karticu sa operativnim sistemom potom treba ubaciti u odgovarajući slot Raspberry Pi-a. Prilikom prvog pokretanja operativnog sistema, potrebno je izvršiti nekoliko podešavanja poput odabira jezika, vremenske zone, postavljanja lozinke, imena korisnika, itd. Takođe, preporučuje se i ažuriranje operativnog sistema. Raspberry Pi OS je izuzetno jednostavan za korišćenje uz pomoć grafičkog korisničkog okruženja (eng. *Graphical User Interface* – GUI) koje je prikazano na slici 50.



Slika 50. Raspberry Pi OS GUI

Programski jezik koji se snažno promoviše za upotrebu kod RPi-a, a koji smo i koristili za potrebe ovog istraživanja je Python. Instalacija Python modula se vrši uz pomoć alata *pip*. U komandnom prozoru je potrebno ispisati liniju za instalaciju *pip*-a:

```
sudo apt-get install python3-pip
```

Kako bi se izbjeglo globalno instaliranje Python modula koje bi moglo da utiče na sistemske alate ili druge projekte, potrebno je instalirati *virtualenv*. *Virtualenv* alat kreira izolovano Python okruženje koje sadrži sopstvenu kopiju Python-a, *pip*-a i posebno mjesto za čuvanje instaliranih modula. Omogućava rad na više projekata sa različitim konfiguracijama, u isto vrijeme, na istoj mašini. Komanda za instaliranje *virtualenv* alata je:

```
sudo apt-get install python-virtualenv
```

Kreirano je i aktivirano novo virtuelno okruženje pomoću programskog koda 18.

Programski kod 18.

```
virtualenv --system-site-packages -p python3 naziv_virtenv
source ~/naziv_virtenv/bin/activate
```

Prije instalacije Python modula, potrebno je ažurirati *pip* na posljednju verziju komandom:

```
easy install -U pip
```

Instalacija Python modula u okviru virtuelnog okruženja se vrši uz pomoć komande:

```
pip3 install naziv_modula
```

Verzija Pythona korišćena za potrebe ovog rada je 3.9.2, *pip*-a 22.2.2, *virtualenv*-a 20.4.0, a verzije korišćenih modula su:

- *Tensorflow* – 2.9.1
- *Librosa* – 0.9.2
- *SciPy* – 1.9.0
- *PySerial* – 3.5b0
- *NumPy* – 1.22.4
- *DateTime* – 4.5
- *SoundDevice* – 0.4.4
- *RPi.GPIO* – 0.7.0

Uz pomoć *Thonny* besplatnog Python razvojnog okruženja, kreiran je programski kod 19 sa nazivom *record_predict.py* koji vrši detekciju zvuka motorne testere i šalje informacije na *cloud*.

Programski kod 19.

```
import tensorflow.keras as keras
import librosa, math, sounddevice, os, serial
import numpy as np
from scipy.io.wavfile import write
from datetime import datetime
import RPi.GPIO as GPIO

SAMPLE_RATE= 22050
DURATION = 30
n_mels = 80
n_fft = 2048
hop_length = 512
num_segments = 10
SAMPLES_PER_TRACK = SAMPLE_RATE * DURATION

detection_true = '1'
reboot = '1'
num_of_detections=0 #broj detekcija u audio signalu

log_path = '/home/user/Desktop/output.txt' #putanja do log fajla
MAX_FILE_SIZE_B = 1024 #maksimalna veličina log fajla

ser = serial.Serial("/dev/ttyUSB0", 115200)
W_buff=['AT\r\n',"AT+CPIN?\r\n","AT+CGATT=1\r\n","AT+SAPBR=3,1,\"APN\",\"tmcg-
4g\"\r\n","AT+SAPBR=3,1,\"USER\",\"38267\"\r\n","AT+SAPBR=3,1,\"PWD\",\"38267\"\r\n","A
T+SAPBR=1,1\r\n","AT+SAPBR=2,1\r\n","AT+HTTPINIT\r\n","AT+HTTPPARA=\"CID\",1\r\n","AT+H
TTPPARA=\"URL\",\"iot.ucg.ac.me/Data.php?wkey=23def23d87ed646e0d04ae941df1eaed&field2=0
\"\r\n","AT+HTTPACTION=0\r\n"]

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(8, GPIO.OUT, initial = GPIO.LOW)

def led_detection():
    GPIO.output(8,GPIO.HIGH)
    time.sleep(0.1)
    GPIO.output(8,GPIO.LOW)
    time.sleep(0.1)
    GPIO.output(8,GPIO.HIGH)
    time.sleep(0.1)
    GPIO.output(8,GPIO.LOW)

def led_no_detection():
    GPIO.output(8,GPIO.HIGH)
    time.sleep(0.1)
    GPIO.output(8,GPIO.LOW)
    time.sleep(0.1)
```

Programski kod 19. (nastavak)

```
def slanje_na_cloud(num, value_to_send, field_num):
    print('Slanje na cloud')
    ser.write(str.encode(w_buff[0]))
    ser.flushInput()
    response = "".encode()
    time.sleep(1)

    while True:
        while ser.inWaiting() > 0:
           response += ser.read(ser.inWaiting())
        if response!= "":
            print (str(response))

            if num < 9:
                print(str(num))
                print ('Salje: ' + str(w_buff[num+1]))
                ser.write(str.encode(w_buff[num+1]))
                time.sleep(1.5)

            if num == 9:
                print(str(num))
                print ('Salje: ' +'AT+HTTPPARA=\"URL\",\"iot.ucg.ac.me/
                Data.php?wkey=23def23d87ed646e0d04ae941df1eaed&'+
                                    field_num +'=')
                ser.write(str.encode('AT+HTTPPARA=\"URL\",\"iot.ucg.ac.me/
                Data.php?wkey=23def23d87ed646e0d04ae941df1eaed&'+
                                    field_num + '='))
                print ('vrijednost: ' + value_to_send)
                ser.write(str.encode(value_to_send))
                ser.write(str.encode('\"\r\n'))
                time.sleep(2)

            if num == 10:
                print(str(num))
                print ('Salje: ' + str(w_buff[num+1]))
                ser.write(str.encode(w_buff[num+1]))
                time.sleep(3)

            if num == 11:
                print(response na kraju: ')
                print (str(response))
                if '200' in str(response):print("OK")
                else:print("Error")

            num = num + 1
            response = "".encode()

            if num == 12:
                print ('KRAJ')
                break

#slanje informacije o ponovnom pokretanju
slanje_na_cloud(0, reboot, 'field2')

#učitavanje modela
model_path = "model"
model = keras.models.load_model(model_path)
```

Programski kod 19. (nastavak)

```
while(1):
    #snimanje zvuka
    print("Snimanje...")
    GPIO.output(8,GPIO.HIGH)
    recording = sounddevice.rec(int(DURATION*SAMPLE_RATE),
    samplerate=SAMPLE_RATE, channels=2)
    sounddevice.wait()
    print("Zavrseno!")
    GPIO.output(8,GPIO.LOW)
    write("recording.wav",SAMPLE_RATE,recording)
    signal_path = 'recording.wav'

    samples_per_segment = int(SAMPLES_PER_TRACK / num_segments)
    num_mel_spectrogram_vectors_per_segment = math.ceil(samples_per_segment /
    hop_length)

    data = {"mel_spectrogram": []}

    #učitavanje snimljenog audio signala
    signal, sr = librosa.load(signal_path, sr = SAMPLE_RATE)

    for d in range(num_segments):
        start = samples_per_segment * d
        finish = start + samples_per_segment

        #ekstraktovanje mel spektrograma
        mel_spectrogram = librosa.feature.melspectrogram(signal[start:finish],
        sr=SAMPLE_RATE, n_fft=n_fft, hop_length=hop_length, n_mels=n_mels)
        log_mel_spectrogram = librosa.power_to_db(mel_spectrogram)
        log_mel_spectrogram = log_mel_spectrogram.T

        if len(log_mel_spectrogram) ==num_mel_spectrogram_vectors_per_segment:
            data["mel_spectrogram"].append(log_mel_spectrogram.tolist())

    #predikcija
    prediction = model.predict(data["mel_spectrogram"])
    predicted_index = np.argmax(prediction, axis=1)

    #trenutno vrijeme
    now = datetime.now()
    timestamp = now.strftime('%Y-%m-%d_%H-%M-%S')
    write_mode = 'a'

    #upisivanje log fajla
    if os.path.isfile(log_path):
        size = os.path.getsize(log_path)
        if size >= MAX_FILE_SIZE_B:
            write_mode = 'w'
    with open(log_path, mode = write_mode) as output_file:
        output_file.write(timestamp + ' Predikcija je: ' +
        np.array2string(predicted_index) + '\n')

    print(predicted_index)

    #obavještavanje korisnika
    for i in range(0,10):
        if (predicted_index[i]==1):
           num_of_detections = num_of_detections + 1;
    if(num_of_detections>=6):
        print("Detektovana motorna testera")
        led_detection()
        slanje_na_cloud(9, detection_true, 'field1')
    else:
        print("Nije detektovana motorna testera")
        led_no_detection()

   num_of_detections = 0
```

Linije koda koje započinju sa ključnom riječju *import* se odnose na module koje je potrebno učitati. Moduli *keras, os, librosa, numpy* i *math* su objašnjeni u poglavljima 3.3, 5.1 i 5.2. *Sounddevice* modul omogućava snimanje i reprodukovanje audio signala, *scipy.io.wavfile.write* čuvanje *NumPy* niza u *.wav formatu, *datetime* rad sa datumima i vremenom, *serial* pruža podršku za serijsku komunikaciju, a *RPi.GPIO* služi za kontrolu GPIO na RPi-u.

U nastavku koda je definisana frekvencija odabiranja, trajanje audio signala, veličina frejma, dužina skoka, broj segmenata na koje se dijeli audio signal, broj *mel bands* u mel spektrogramu i ukupan broj odbiraka u audio signalu. Na ulaz modela neuralne mreže se prosljeđuje snimljeni audio signal dužine trajanja 30s, podijeljen na 10 segmenata. Za svaki segment audio signala neuralna mreža će dati predikciju, pa će izlaz modela biti vektor od 10 elemenata koji mogu imati vrijednost 0, 1 ili 2 (ambijentalni zvukovi, zvukovi motorne testere ili zvukovi slični motornoj testeri).

Osim slanja informacije o detekciji zvuka motorne testere na *cloud* platformu (1 – detektovana motorna testera), program će predikcije neuralne mreže ispisivati i u lokalni *log* fajl. Definisana je putanja i maksimalna veličina *log* fajla u bajtovima. Kao alternativa skladištenju informacija o detekciji na *cloud* platformu, moguće je poslati SMS poruku ili inicirati GSM poziv.

Serijski port za komunikaciju sa SIM 868 se otvara pomoću *serial* klase iz istoimenog modula, pri čemu je potrebno definisati port (*/dev/ttyUSB0*) i *baudrate* (115200 *baud*-a). Definiše se niz *W_buff* koji sadrži AT komande za komunikaciju RPi-a i SIM 868 modula:

AT - provjera komunikacije; vraća rezultat OK ako su RPi i SIM 868 modul pravilno povezani; ako modul ili SIM kartica ne radi, vratiće rezultat ERROR;

AT+CPIN=**** - slanje PIN-a SIM kartice; ukoliko je kartica otključana, onda je dovoljno pomoću AT+CPIN? komande provjeriti zahtjev za lozinkom; ako modul vrati rezultat READY, nije potrebno unositi PIN SIM kartice;

AT+CGATT=1 – priključivanje uređaja na GPRS servis;

AT+SAPBR=3,1,"APN","tmcg4g" – podešavanje naziva pristupne tačke za mobilni Internet;

AT+SAPBR=3,1,"USER","38267" - podešavanje korisničkog imena za odabranu pristupnu tačku;

AT+SAPBR=3,1,"PWD","38267" - podešavanje lozinke za odabranu pristupnu tačku i korisničko ime; naziv pristupne tačke, korisničko ime i lozinku za Telekom Internet servis provajdera je moguće pronaći na [51];

AT+SAPBR=1,1 – omogućava GPRS servis;

AT+SAPBR=2,1 – provjerava da li je veza ispravno podešena; kao rezultat vraća IP adresu;

AT+HTTPINIT – inicijalizuje HTTP sesiju.

AT+HTTPPARA="CID",1 – podešavanje HTTP parametara; dodjeljuje ID;

AT+HTTPPARA="URL","iot.ucg.ac.me/Data.php?wkey=23def23d87ed646e0d04ae941df1eaed&field2=0"; dodjeljuje *url*;

AT+HTTPACTION = 0 – komanda za izvršenje HTTP GET zahtjeva; rezultat komande je u formatu *+HTTPACTION*: *<Method><StatusCode>,<DataLen>*; zahtjev je uspješno izvršen ako je *StatusCode* = 200;

Detaljnije o AT komandama se može pronaći na [52] i [53].

Pomoću *GPIO.setwarnings*(*False*) funkcije, onemogućena su upozorenja GPIO modula. Postoje dva načina numerisanja IO pinova na RPi-u. Prvi je korišćenje sistema numerisanja BOARD. Ovo se odnosi na brojeve pinova odštampane na RPi pločici. Drugi sistem numerisanja su BCM (eng. *Broadcom SOC channel*) brojevi. U ovom radu je korišćen BOARD sistem numerisanja.

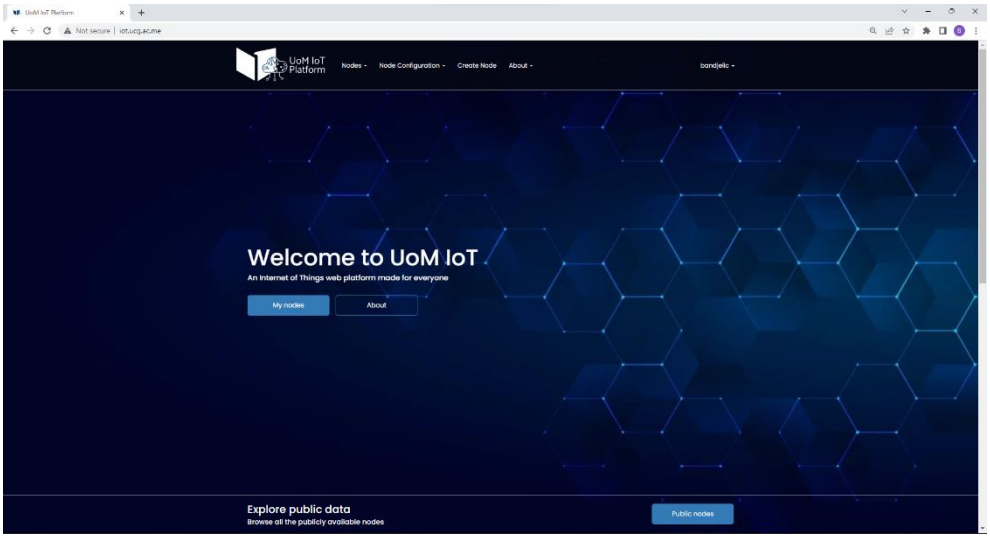
Snimanje audio signala i detekcija motorne testere će se signalizirati pomoću LED-a, pa je potrebno konfigurisati pin na koji je LED povezana kao izlazni i postaviti inicijalnu vrijednost na LOW – isključena dioda. Pozitivna detekcija motorne testere se signalizira sa dva treptaja LED-a – funkcija *led_detection*(), a u suprotnom se vrši jedan treptaj LED-a – funkcija *led_no_detection*(). Tokom snimanja audio signala LED će biti stalno uključena.

Funkcija *slanje_na_cloud* uzima tri argumenta: *num* – početnu poziciju AT komande iz niza *W_buff* , *value_to_send* – informaciju koju treba da pošalje na *cloud* i *field_num* – polje u koje upisuje informaciju. Funkcija prolazi kroz niz *W_buff* počev od pozicije *num*, šalje AT komande SIM 868 modulu, odgovor od modula skladišti u promjenljivu *response* i prikazuje ga korisniku. Prilikom prvog pokretanja programa, potrebno je proći kroz cijeli niz *W_buff*, što se obezbjeđuje pozivanjem funkcije *slanje_na_cloud*(*0, reboot, 'field2'*) u glavnom dijelu programa. Na ovaj način se SIM 868 modulu prosljeđuju neophodni podaci za povezivanje na Internet i vrši upisivanje informacije o prvom paljenju uređaja u *field2*. Samim tim, moguće je znati da li se i koliko puta resetovao uređaj. Svakim sljedećim upisivanjem na *cloud* je dovoljno proći kroz niz *W_buff* počev od pozicije *num* = *9*, koja će korigovati URL HTTP sesije.

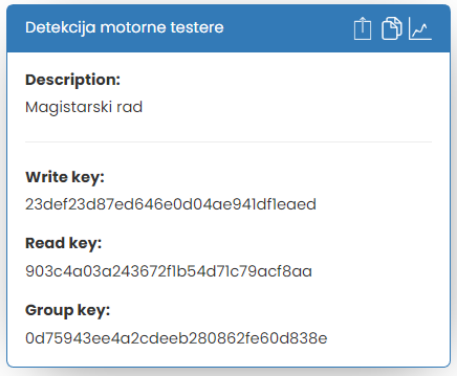
Podaci se šalju na UoM IoT platformu (slika 51) kojoj je moguće pristupiti preko linka http://iot.ucg.ac.me/ ili http://89.188.32.140/. Potrebno je registrovati se na platformu i kreirati tzv. *node*. Svaki *node* ima svoj pristupni ključ (slika 52), koji je potrebno uključiti u *url* prilikom podešavanja HTTP parametara:

```
iot.ucg.ac.me/Data.php?wkey=23def23d87ed646e0d04ae941df1eaed&field1=1
```

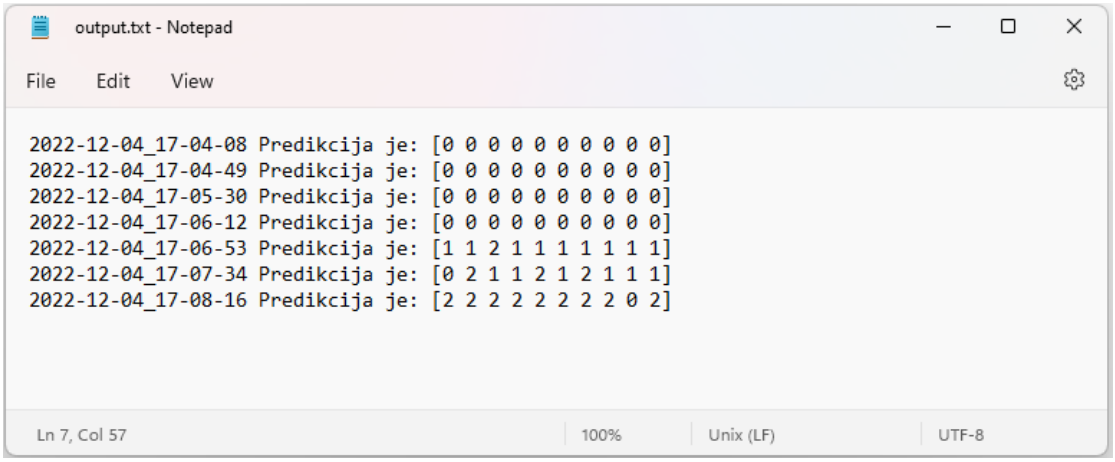
U glavnom dijelu programa se vrši učitavanje modela neuralne mreže koji je u poglavlju 5.4 izabran kao optimalni. Slijedi snimanje audio signala, što se signalizira uključivanjem LED-a, podjela na segmente snimljenog signala, računanje karakteristika za svaki segment i prosljeđivanje na ulaz modela. Model vrši predikciju i kao rezultat vraća vektor od 10 elemenata sa vrijednostima 0, 1 ili 2. Pomoću *datetime.now()* funkcije dobija se informacija o datumu i trenutnom vremenu, koja se zajedno sa predikcijama modela upisuje u log fajl *output.txt* (slika 53). Ako bar 6 elemenata u vektoru ima vrijednost 1, onda je detektovana motorna testera. Upisivanje na UoM IoT platformu se vrši pozivanjem funkcije *slanje_na_cloud*(*9, detection_true, 'field1'*).



Slika 51. UoM IoT platforma – početni ekran



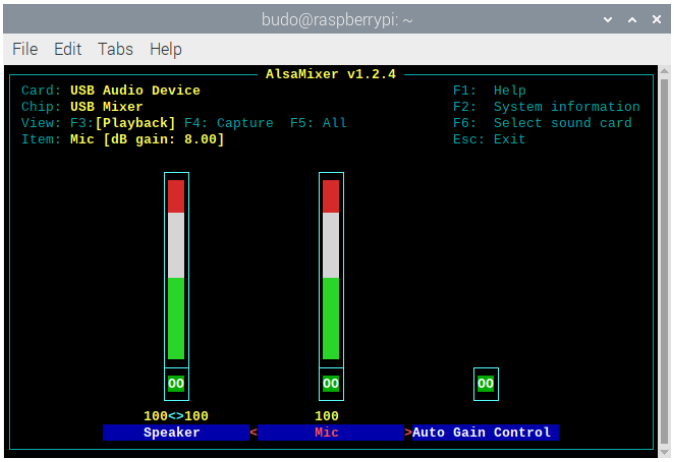
Slika 52. *Node* pristupni ključevi



Slika 53. Log fajl

Za konfiguraciju postavki zvuka i podešavanje pojačanja mikrofona koristi se *alsamixer* (slika 54), koji dolazi sa operativnim sistemom i pokreće se iz komandnog prozora linijom:

```
alsamixer
```



Slika 54. AlsaMixer

Pokretanje *record_predict.py* fajla iz virtuelnog okruženja je moguće uz pomoć komande:

```
/home/user/naziv_virtenv/bin/python /home/user/naziv_virtenv/record_predict.py
```

Prvi dio komande se odnosi na putanju do Python-a koji se koristi u virtuelnom okruženju, a drugi dio na putanju do *record_predict.py* fajla. Ova komanda je sačuvana kao *bash* skripta pod nazivom *startup.sh*. Za automatsko pokretanje skripte sa podizanjem operativnog sistema je potrebno modifikovati *.bashrc* fajl komandom:

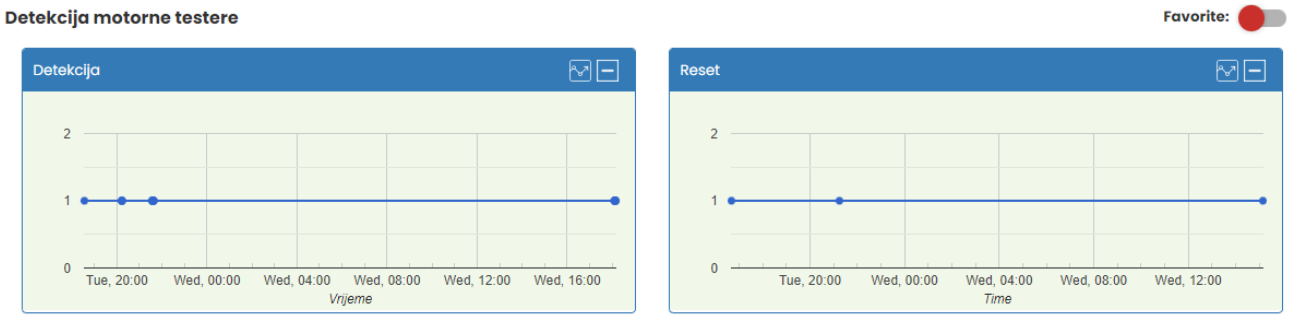
```
sudo nano /home/user/.bashrc
```

U posljednjoj liniji dodati putanju do prethodno kreirane *startup.sh* skripte:

```
sh /home/user/startup.sh
```

Prije izlaska iz *.bashrc* fajla treba zapamtiti sve promjene. Prilikom prekida u napajanju RPi-a, program će se automatski pokrenuti sa narednim podizanjem OS-a.

Dijagram sa pristiglim informacijama o detekciji zvuka motorne testere na *node* na UoM IoT platformi prikazan je na slici 55. Polje sa nazivom *Detekcija* se odnosi na trenutke u kojima je detektovan zvuk motorne testere, dok se polje sa nazivom *Reset* odnosi na trenutke u kojima je došlo do resetovanja uređaja.



Slika 55. Prikaz podataka sa *node*-a na UoM IoT platformi

6 ZAKLJUČAK

U ovom radu je obrađena detekcija sječe šuma zasnovana na dubokom učenju. Predložena su dva modela za detekciju zvuka motorne testere: potpuno povezana neuralna mreža i konvoluciona neuralna mreža. CNN model, koji se pokazao preciznijim, postiže tačnost klasifikacije od 94.96% na skupu podataka prikupljenom sa YouTube-a i 88.87% na skupu podataka snimljenom za potrebe ovog istraživanja. Predloženi model detekcije je implementiran na pristupačnoj mikroprocesorskoj platformi opremljenoj sa jednostavnim mikrofonom, GPS i GSM/GPRS modulom. Ovaj sistem za detekciju sječe šuma bi se u obliku senzorskih čvorova mogao instalirati na stablima u šumi, dok bi se napajanje vršilo preko solarnih panela. Kada dođe do detekcije zvuka motorne testere, sistem obavještava korisnika o događaju slanjem informacije na *cloud*. Osim toga, moguće je poslati i SMS ili inicirati GSM poziv. Na ovaj način se može doprinijeti smanjenju bespravne sječe šuma te tako smanjiti dugoročne posljedice po okolinu, kvalitet života ljudi i ekonomiju. Takođe, ovakav sistem sa odgovarajućim skupom podataka bi mogao naći primjenu u različitim oblastima.

Ograničavajući faktori ovog sistema se odnose na razlike u kvalitetu upotrebljenog mikrofona koje mogu onemogućiti uređaje da snime suptilne audio detalje, neophodne za robusno otkrivanje određenih zvukova, izdržljivost sistema u nepovoljnim ambijentalnim uslovima, kao i slabu pokrivenost 4G signalom u ruralnim oblastima. Mikrofoni vrhunskog kvaliteta imaju bolje komponente u cijeloj svojoj konstrukciji, od materijala koje koriste za prihvatanje zvučnih vibracija, do kvaliteta elektronike i dizajna. Ograničen frekvencijski opseg, neujednačen frekvencijski odziv, šum, ograničen dinamički opseg i distorzija su među aspektima koji mikrofon čine manje kvalitetnim. Dobar mikrofon će podjednako prenositi sve frekvencije zvuka. Mikrofoni lošeg kvaliteta izobličavaju zvuk, izraženo je prisustvo šuma, neke frekvencije su naglašenije od drugih, a neke mogu biti podložne promjenama faza, itd.

Prilikom dizajniranja sistema potrebno je uzeti u obzir i ambijentalne uslove poput temperature, vlažnosti, vibracija i elektromagnetnih smetnji. Povećanje temperature degradira performanse senzorskih čvorova, usporava mikrokontrolerske jedinice, smanjuje životni vijek baterije, smanjuje performanse elektronskih komponenti i ubrzava njihovo starenje. Visoka vlažnost smanjuje kvalitet veze i povećava vjerovatnoću kratkog spoja u senzorskim čvorovima. Slično tome, jake elektromagnetne smetnje povećavaju stopu gubitka podataka. Aktivnosti bespravne sječe šuma se najčešće sprovode u ruralnim i udaljenim oblastima. S obzirom da je pristup 4G signalu neravnomjerno raspoređen između ruralnih i urbanih područja, rad sistema će biti otežan.

Naučni doprinos se ogleda u inovativnim metodama koje su korišćene u radu, kao i u poboljšanju postojećih rezultata u ovoj oblasti. Osim toga, svojevrsni doprinos se ogleda i u dva skupa podataka koji su prikupljeni u svrhe ovog istraživanja i koji se mogu koristiti za dalje istraživanje u ovoj ili sličnim oblastima. Ovaj rad pruža pregled tehnika detekcije zvuka neuralnim mrežama, karakteristika audio signala koje se mogu koristiti za klasifikaciju/detekciju zvuka, kao i arhitektura neuralnih mreža koje su najefikasnije za konkretnu primjenu.

Prilikom izbora optimalnog modela, razmatrane su različite kombinacije parametara neuralne mreže, što može značajno olakšati buduća istraživanja na ovu temu. Takođe, dostupni su i kodovi u programskom jeziku Python koji se koriste za ekstrahovanje karakteristika, implementaciju neuralnih mreža, snimanje audio signala, detekciju i obavještavanje korisnika.

Ovo istraživanje je otvorilo veliki broj pravaca za dalji rad. Buduća istraživanja se mogu odnositi na problematiku napajanja predloženog sistema, potrošnju električne energije, dužinu trajanja baterije i *sleep/active* šeme. Platforme bežičnih senzorskih čvorova su ograničene resursima kada je u pitanju napajanje, računarske i skladišne mogućnosti. Tradicionalno, senzorske čvorove napajaju male baterije ograničenog kapaciteta. Za različite scenarije primjene, predviđeni životni vijek senzorskih čvorova može biti kratak ili i do nekoliko godina, a zamjena baterije je skupa, nezgodna ili čak nemoguća, posebno u velikim, udaljenim i opasnim okruženjima. Stoga, dizajneri predlažu energetski efikasne senzorske čvorove, uključujući komunikacione protokole sa niskom potrošnjom energije (ZigBee, SigFox, *Bluetooth Low Energy,* itd.) i *sleep/active* šeme.

Procesori u senzorskim čvorovima su sposobni da funkcionišu u različitim operativnim režimima kao što su aktivni, neaktivni i mirni režim, što smanjuje nepotrebnu potrošnju energije. Sakupljanje energije iz potencijalnih izvora kao što su RF (eng. *Radio Frequency*) signali, solarna energija, vibracije i vjetar može produžiti životni vijek baterije. Sa druge strane, veličina sistema napajanja utiče na ukupnu fizičku veličinu, cijenu i težinu senzorskog čvora, pa je potrebno naći odgovarajući kompromis.

Buduća istraživanja mogu uzeti u obzir uticaj udaljenosti izvora zvuka, način na koji mikroprocesorska platforma vrši snimanje zvuka i uticaj kvaliteta mikrofonskih senzora na tačnost sistema. Utvrđivanjem maksimalne udaljenosti do koje mikrofon registruje zvuk će pomoći da se odredi ukupan broj senzorskih čvorova koje treba postaviti za određenu površinu. Potrebno je uzeti u obzir i gustinu šume, što može uticati na prostiranje zvuka.

Poboljšanje skupova podataka primjenom metoda vještačkog generisanja podataka kao što su promjena trajanja audio signala bez promjene visine tona, promjena visine tona bez promjene trajanja audio signala, dodavanje šuma, primjena filtara u frekvencijskom domenu, nasumična promjena amplitude zvučnog signala, zatim prikupljanje većeg broja različitih podataka, eksperimentisanje sa arhitekturama neuralnih mreža, dodatna optimizacija parametara, povećanje broja karakteristika, korišćenje potpuno drugih karakteristika, ekstrakcija karakteristika pomoću neuralne mreže, samo su neki od mnogobrojnih pravaca za dalje istraživanje.

LITERATURA

[1] [online] Forest resource definition, Law Insider, Dostupno na: https://www.lawinsider.com/dictionary/forest-resource Pristupano: 25. Jul 2022. godine

[2] [online] Importance of Forest - Introduction, uses and economic benefit, Dostupno na: https://www.vedantu.com/chemistry/importance-of-forest Pristupano: 25. Jul 2022. godine

[3] Gazdić, M., Gajević, M., Zarubica, D., & Iković, V. (2020). Šumarstvo, alternativa razvoja Crne Gore, Dostupno na: https://www.researchgate.net/publication/353795133_Sumarstvo_alternativa_razvoja_Crne_Gore Pristupano: 26. Jul 2022. godine

[4] [online] Šumarstvo Metapodaci, Periodika: Godišnja, Godina: 2016, Dostupno na: https://data.stat.gov.rs/Metadata/13_Poljoprivreda/Html/1304_ESMS_G0_2016_2.html Pristupano: 26. Jul 2022. godine

[5] [online] Izvještaj o zdravstvenom stanju šuma u 2015. godini (za verifikaciju), Dostupno na: https://www.gov.me/dokumenta/f2103a6f-fc04-478e-af2d-ce711bbeb18f Pristupano: 26. Jul 2022. godine

[6] [online] Izvještaj o zdravstvenom stanju šuma za 2018. godinu, Dostupno na: https://www.gov.me/dokumenta/31777e1c-922c-47c7-a34f-82729658ebc2 Pristupano: 26. Jul 2022. godine

[7] [online] Izvještaj o procjeni zdravstvenog stanja šuma i održivosti gazdovanja šumama za 2019. godinu, Dostupno na: https://www.gov.me/dokumenta/02d45e0e-0287-44e2-bdda-9bd6b2d62f1d Pristupano: 26. Jul 2022. godine

[8] [online] Izvještaj o zdravstvenom stanju šuma u 2020. godini, Dostupno na: https://www.gov.me/dokumenta/38c46e52-7e71-4bc7-aad2-e4e774c839fd Pristupano: 26. Jul 2022. godine

[9] [online] Izvještaj o zdravstvenom stanju šuma u 2021. godini (bez rasprave), Dostupno na: https://www.gov.me/dokumenta/b5481ac6-bbaa-456c-9e47-3739090513de Pristupano: 26. Jul 2022. godine

[10] Muhammad, Nur, Haniff, Mohd, Noora, Rokiah, Kadirb, Suriyani, & Muhamadc (2020). Timber Theft: Examining the Factors of Illegal Logging.

[11] Ertel, W. (2017) Introduction to Artificial Intelligence. 10.1007/978-3-319-58487-4.

[12] Moons, B., Bankman, D., & Verhelst, M. (2019). Embedded Deep Neural Networks: Algorithms, Architectures and Circuits for Always-on Neural Network Processing. 10.1007/978-3-319-99223-5_1.

[13] Colonna, J., Gatto, B., Santos, E., & Nakamura, E. (2016). A Framework for Chainsaw Detection Using One-Class Kernel and Wireless Acoustic Sensor Networks into the Amazon Rainforest. 34-36. 10.1109/MDM.2016.86.

[14] Nicolae, G., Gaita, A., Radoi, A., & Burileanu, C. (2018). A Method for Chainsaw Sound Detection Based on Haar-like Features. 1-5. 10.1109/TSP.2018.8441379.

[15] Gnamélé, N., Ouattara, Y., Kobea, T., Baudoin, G., & Laheurte, J.M. (2019). KNN and SVM Classification for Chainsaw Identification in the Forest Areas. International Journal of Advanced Computer Science and Applications. 10. 10.14569/IJACSA.2019.0101270.

[16] Tang, Y., Han, P., Wang, Z., Longcan, H., Gao, Y., & Li, H. (2012). Based on intelligent voice recognition of forest illegal felling of detecting methods. 1153-1156. 10.1109/CCIS.2012.6664564.

[17] Soisoonthorn, T., & Rujipattanapong, S. (2007). Deforestation detection algorithm for wireless sensor networks. 10.1109/ISCIT.2007.4392237.

[18] Liu, Y., Cheng, Z., Liu, J., Yassin, B., Nan, Z., & Luo, J. (2019). AI for Earth: Rainforest Conservation by Acoustic Surveillance.

[19] Rizky R.B., Abdurohman, M., & Prabowo, S. (2019). Accuracy Enhancement of Feature Extraction Scheme in Detection of Chainsaw Sound to Prevent Illegal Logging. 56-61. 10.1109/ICSIGSYS.2019.8811072.

[20] [online] ESC-50 dataset, Dostupno na: https://github.com/karolpiczak/ESC-50 Pristupano 07. Jul 2022. godine

[21] Copeland, B., & Proudfoot, D. (2007). Artificial Intelligence. 10.1016/B978-044451540-7/50032-3.

[22] Ogunsote, O.O. (2007). Propagation of sound, Dostupno na: http://sdngnet.com/Files/Lectures/FUTA-ARC-507/Assignments/2007%20Assignments/Term%20Papers/Propagation%20of%20Sound.pdf Pristupano: 27. Feb 2022. godine

[23] [online] Sound Wave Demonstrating Compression And Rarefaction, Dostupno na: https://www.seekpng.com/ima/u2r5q8o0u2e6y3y3/ Pristupano: 27. Feb 2022. godine

[24] [online] Pitch, Loudness and Timbre, Dostupno na: https://www.compadre.org/osp/EJSS/4485/270.htm#:~:text=Perceived%20loudness%20is%20related%20to,sound%20wave%20as%20independent%20properties Pristupano: 01. Mart 2022. godine

[25] Müller, M. (2015). Fundamentals of Music Processing. 10.1007/978-3-319-21945-5.

[26] Radusinović, I., Predavanja iz predmeta Osnove digitalnih telekomunikacija, Dostupno na: https://drive.google.com/file/d/1FNSY2Ji0IwS0Ug8VsTVQeKTLcxNxhKII/view?usp=sharing Pristupano 02. Mart 2022. godine

[27] [online] Audio Feature Extraction, Dostupno na: https://devopedia.org/audio-feature-extraction#:~:text=Generally%20audio%20features%20are%20categorised,%2C%20segment%2Dlevel%20and%20global_ Pristupano 02. Mart 2022. godine

[28] [online] Understanding FFTs and Windowing, Dostupno na: https://download.ni.com/evaluation/pxi/Understanding%20FFTs%20and%20Windowing.pdf Pristupano 04. Mart 2022. godine

[29] [online] Download Python, Dostupno na: https://www.python.org/downloads/ Pristupano 11. April 2022. godine

[30] [online] Download PyCharm Python IDE for Professional Developers by JetBrains, Dostupno na: https://www.jetbrains.com/pycharm/download/#section=windows Pristupano 11. April 2022. godine

[31] Stankovic, Lj. (2015). Digital Signal Processing with Selected Topics.

[32] [online] Mel scale – Wikipedia, Dostupno na: https://en.wikipedia.org/wiki/Mel_scale Pristupano 14. Mart 2022. godine

[33] [online] Speech Processing for Machine Learning: Filter banks, Mel-Frequency Cepstral Coefficients (MFCCs) and What's In-Between, Haytham Fayek, Dostupno na: https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html Pristupano 14. Mart 2022. godine

[34] [online] Cepstrum – Wikipedia, Dostupno na: https://en.wikipedia.org/wiki/Cepstrum#:~:text=The%20cepstrum%20is%20a%20representation,human%20voice%20and%20musical%20signals. Pristupano 06. Jul 2022. godine

[35] Norton, M., & Karczub, D. (2003). Fundamentals of Noise and Vibration Analysis for Engineers (2nd ed.). Cambridge: Cambridge University Press. doi:10.1017/CBO9781139163927

[36] Haykin, S. S. (2009). Neural networks and learning machines. Upper Saddle River, NJ: Pearson Education.

[37] Negnevitsky, M. (2002). Artificial intelligence: A guide to intelligent systems. New York: Addison Wesley.

[38] [online] Izvod složene funkcije – Wikipedia, Dostupno na: https://sr.wikipedia.org/sr-el/%D0%98%D0%B7%D0%B2%D0%BE%D0%B4_%D1%81%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B5_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%98%D0%B5 Pristupano 09. April 2022. godine

[39] Varma, S., & Das, S. (2018). Introduction to Deep Learning, Dostupno na: https://srdas.github.io/DLBook/GradientDescentTechniques.html#issues-with-gradient-descent Pristupano 10. April 2022. godine

[40] [online] Jovanović, J., Mašinsko učenje, Dostupno na: http://ai.fon.bg.ac.rs/wp-content/uploads/2016/10/ML_intro_2016.pdf Pristupano 11. April 2022. godine

[41] Goodfellow, I., Courville, A., & Bengio, Y. (2016). Deep Learning. United Kingdom: MIT Press.

[42] [online] Basic CNN Architecture: Explaining 5 Layers of Convolutional Neural Network, Dostupno na: https://www.upgrad.com/blog/basic-cnn-architecture/ Pristupano 26. April 2022. godine

[43] [online] Download Audacity, Dostupno na: https://www.audacityteam.org/download/ Pristupano 11. April 2022. godine

[44] [online] Audio Editing Software. Sound, Music, Voice & MP3 Editor. Best Audio Editor for 2022, Dostupno na: https://www.nch.com.au/wavepad/index.html Pristupano 11. April 2022. godine

[45] [online] Statistical modeling and deep learning image analysis for lung cancer risk prediction, Dostupno na: https://marskar.github.io/frm/#16 Pristupano 11. April 2022. godine

[46] Veit, D. (2012). Neural networks and their application to textile technology. Simulation in Textile Technology: Theory and Applications. 9-71. 10.1533/9780857097088.9.

[47] [online] A Comprehensive Guide on Deep Learning Optimizers - Analytics Vidhya, Dostupno na: https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/ Pristupano 11. April 2022. godine

[48] [online] Raspberry Pi - Wikipedia, Dostupno na: https://en.wikipedia.org/wiki/Raspberry_Pi Pristupano 09. Jun 2022. godine

[49] [online] Single-board computer - Wikipedia, Dostupno na: https://en.wikipedia.org/wiki/Single-board_computer Pristupano 09. Jun 2022. godine

[50] Mitrović, A. (2019). Primjena metoda mašinskog učenja kod klasifikacije zvuka u kućnom okruženju

[51] [online] Kako da podesim mobilni telefon za internet - Crnogorski Telekom, Dostupno na: https://telekom.me/poslovni-korisnici/korisnicka-zona/clanak/kako-da-podesim-mobilni-telefon-za-internet Pristupano 06. Dec 2022. godine

[52] [online] AT Commands Reference Guide, Dostupno na: https://www.sparkfun.com/datasheets/Cellular%20Modules/AT_Commands_Reference_Guide_r0.pdf Pristupano 06. Dec 2022. godine

[53] [online] FTP HTTP AT Commands User Guide, Dostupno na: https://researchdesignlab.com/projects/AN_SIM900_FTP_HTTP_AT_COMMANDS_USER_GUIDE_beta_V1.00.pdf Pristupano 06. Dec 2022. godine

PRILOG

Prilog 1 – Izvori sa kojih su preuzeti audio signali (*Prilog uz magistarski rad.pdf*)