Odvajanje ritmičkih elemenata od melodijskih u pjesmama

Crnogorac, Grgur

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 646

ODVAJANJE RITMIČKIH ELEMENATA OD MELODIJSKIH U PJESMAMA

Grgur Crnogorac

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 646

ODVAJANJE RITMIČKIH ELEMENATA OD MELODIJSKIH U PJESMAMA

Grgur Crnogorac

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

DIPLOMSKI ZADATAK br. 646

Pristupnik:	**Grgur Crnogorac (0036513611)**
Studij:	Računarstvo
Profil:	Programsko inženjerstvo i informacijski sustavi
Mentor:	izv. prof. dr. sc. Daniel Hofman

Zadatak: **Odvajanje ritmičkih elemenata od melodijskih u pjesmama**

Opis zadatka:

Pri izradi nove glazbe ponekad se koriste postojeće pjesme koje se na određeni način modificiraju. Kako bi se moglo dodati nove instrumente potrebno je neke izbaciti odnosno utišati. U ovom radu razvijat će se metode za izoliranje određenih instrumenata u pjesmi. Potrebno je proučiti načine prepoznavanja dijelova pjesama tako da se mogu odvojiti ritmički elementi od melodijskih. Proučiti načine razdvajanja koji se koriste kod mikseta koristeći frekvencijske filtere. Proučiti načine programskih implementacija izdvajanja instrumenata iz pjesama. Proučiti naprednije metode za razdvajanje harmonijskih i ritmičkih elemenata. Proučiti načine ubrzavanja odvajanja dijelova pjesama koristeći GPU. Implementirati u aplikaciji postojeće metode i proširiti ih. Testirati implementirane metode i usporediti kvalitetu i brzinu rada. Vizualno prikazati dekomponirane dijelove i omogućiti spremanje novonastalih zvukova u datoteke. Proširiti funkcionalnost aplikacije dodavanjem više metoda za separaciju instrumenata te usporediti kvalitetu različitih metoda. Predložiti načine razdvajanja vokalnih dijelova pjesama. Napraviti dokumentaciju i objaviti programski kôd na repozitoriju Git.

Rok za predaju rada: 28. lipnja 2024.

Zahvaljujem se mentoru, izv. prof. dr. sc. Danielu Hofmanu, na pomoći i susretljivosti kroz diplomski studij te prilikom izrade ovog rada. Također, zahvaljujem se prijateljima, kolegama i obitelji na podršci kroz život i tijekom studija.

SADRŽAJ

1. Uvod

Glazbeno djelo je složena umjetnička forma koja se sastoji od različitih elemenata, glavni od kojih su instrumenti i vokali, no osim njih za uređivanje na kraju dobivenog zvuka koriste se razne metode iz područja glazbenog/zvučnog inženjerstva.

Jedan instrument, glas ili kombinacija više njih povezanih određenim svojstvom u terminologiji se nazivaju izvorom (*musical source*). Izvorom se može smatrati skup svih instrumenata s vokalima koji proizvode melodiju, u kojem slučaju bi drugi izvor bili ujedinjeni ritmički elementi. Izvorom se također može smatrati i pojedini instrument, kao i glavni te popratni vokali, instrumenti koji proizvode zvuk nižih frekvencija (bas) te razni drugi.

U glazbenoj industriji pristup razdvojenim izvorima glazbenog djela vrlo je koristan u razne svrhe, od stvaranja *remix-a* pjesme, modernizacije starog glazbenog djela u boljoj kvaliteti, automatskog prepoznavanja teksta pjesme, pjevača ili korištene ljestvice do ubacivanja specifičnih elemenata u *miks* tijekom *"live"* DJ nastupa.

Proces razdvajanja pojedinih izvora glazbenog djela na engleskom se jeziku naziva *"musical source separation"*, te je glavna tematika ovog rada. U radu su navedene te objašnjene razne metode razdvajanja izvora, od jednostavnijih metoda temeljenim na pojedinačnim svojstvima instrumenata ili glazbenih djela, do kompleksnijih modela strojnog / dubokog učenja dizajniranih i naučenih upravo u tu svrhu. U sklopu rada realizirana je web aplikacija s grafičkim korisničkim sučeljem koja omogućuje učitavanje zvučne datoteke, razdvajanje izvora u datoteci odabranom metodom, preslušavanje i spremanje novonastalih zvučnih datoteka s pojedinim izvorima te pregled spektrograma unesene i novonastalih zvučnih signala, odnosno vizualizacija u domeni vrijeme-frekvencija.

2. Osnovni pojmovi

Neki od osnovnih pojmova često korištenih u domeni odvajanja glazbenih izvora navedeni su niže:

2.1. Smjesa

Skupina istovremenih individualnih zvukova - izvora, koja se u procesu razdvajanja koristi kao polazna točka naziva se **smjesa** (*mixture*). U matematičkom pogledu, signal smjese $y(t)$ koja se sastoji od N izvora definira se kao suma signala pojedinih izvora:

$$y(t) = \sum_{i=1}^{N} (x_i(t)) \tag{2.1}$$

te je cilj procesa razdvajanja izvora sa smjesom $y(t)$ kao jedinim ulaznim podacima, dobiti jedan ili više izvora $x_i(t)$

2.2. Izvor

Kao što je već spomenuto, definicija izvora u ovom kontekstu nije jednoznačna, te se izvorom može smatrati jedan ili više elemanata ukupnog zvučnog signala, odnosno smjese. Generalno, izvorom se smatra element ili kolekcija elemenata smjese grupiranih po određenim karakteristikama.

2.3. Valni oblik

Valni oblik (*waveform*) grafički je prikaz signala u vremenskoj domeni. Valni oblik pokazuje kako se amplituda signala mijenja tijekom vremena. Koristi se za vizualizaciju i analizu osnovnih karakteristika zvuka.

2.4. Spektrogram

Spektrogram je također grafički prikaz zvučnog signala, no dobiven transformacijom iz domene vremena (valni oblik) u domenu vrijeme-frekvencija. Ta se transformacija najčešće postiže pomoću STFT (*Short Time Fourier Transform*).

2.5. Maskiranje

Maskiranje je proces izrade matrice odgovarajućih dimenzija čijim se množenjem s digitiziranom reprezentacijom zvučnog signala dobiva neki njegov podskup. Maske se često izrađuju nad spektrogramom zvučnog signala, te se množenjem maske i spektrograma s obzirom na elemente (Hadamardov produkt) dobiva željeni izvor. Maske su najčešće binarne ili racionalne (mekane) s domenom [0, 1].

3. Metode razdvajanja

Razdvajanje glazbenih izvora kao aspekt umjetnosti glazbe, te kao meta znanstvenog istraživanja, postoji već desetljećima. Iako je u posljednje vrijeme većinu pažnje prisvojilo istraživanje softverskih metoda separacije, u svijetu elektroničke glazbe taj je proces sastavni dio nastupa svakog DJ-a.

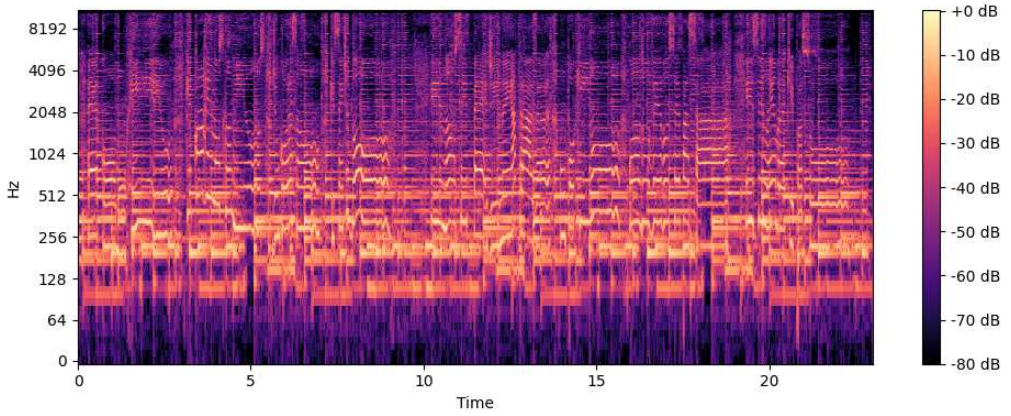
Tijekom tranzicije između pjesama redovna je aktivnost istovremena reprodukcija trenutne i iduće pjesme na redu (profesionalne miksete omogućuju reprodukciju 3, 4 ili više pjesama odjednom). Za pomoć u tom procesu moderne miksete sadrže razne funkcionalnosti kojima se zvuk mijenja da bi poprimio karakteristike koje postižu željeni glazbeni efekt, glavni od kojih su frekvencijski filteri, odnosno pojasni *equalizer*. Frekvencijski filteri su (najčešće 3) ručice kojima se kontrolira amplituda zvuka trenutno reporoducirane pjesme u određenom frekvencijskom pojasu. Najčešći rasponi pojaseva su: 20-200 Hz (*low*), 200-5000 Hz (*mid*) te 5000-20000 Hz (*high*). Kontrola glasnoće zvuka po određenom pojasu DJ-u omogućuje pojednostavljen oblik razdvajanja glazbenih izvora u stvarnome vremenu tijekom nastupa.

Osim takvih metoda separacije uživo, glazbeni producenti često se koriste softverom zvanim DAW (*Digital Audio Workstation*) namijenjenim produkciji glazbe, u koji se može učitati gotova pjesma ili neki njen isječak. Producent zatim "ručno" primjenjuje EQ, razne efekte ili dodatno filtira digitalni audio signal po određenim frekvencijama da bi prenamijenio, odnosno "*remix-ao*" pjesmu ili iz nje izolirao određeni podskup zvukova.

Softverski automatizirana separacija glazbenih izvora, tema ovog rada, tema je istraživanja mnogih znanstvenika, odjela na učilištima i organizacija posljednjih dvadesetak i više godina. Većina dosadađnjih metoda kreće od iste početne točke - digitiziranog glazbenog signala prebačenog iz vremenske u domenu vrijeme-frekvencija, najčešće metodom STFT. Važan aspekt STFT-a je inverzibilnost, odnosno mogućnost prebacivanja signala natrag u originalni, valni oblik. Posljednjih godina, razvijaju se i metode s valnim oblikom signala kao ulaznim podacima, većinom s podlogom u području strojnog i dubokog učenja.

3.1. Median filtering

Na slici 3.1 prikazan je tipičan spektrogram neke pjesme. Lako je primijetiti neke konzistentnosti na slici, u redovnim intervalima raspoređene vertikalne linije lokalizirane u nekoj "točki" u vremenu (ili nekom kratkom vremenskom periodu), te horizontalne linije koje odgovaraju različitim frekvencijama. Ove karakteristike predstavljaju harmoničke i ritmičke elemente pjesme. To svojstvo audio signala u domeni vrijeme-frekvencija je osnovica metode *median filtering*[4].



Slika 3.1: Spektrogram isječka pjesme

Median filtering nelinearna je metoda procesiranja digitalnog signala temeljena na određivanju medijana unutar prozora određene veličine i oblika u okolini svake točke u digitalnoj reprezentaciji signala, te zamjenom vrijednosti izračunatim medijanom. Često je korištena u procesu smanjenja šuma signala zbog svojstva očuvanja "rubova" odnosno strmih prijelaza uz smanjivanje šuma. U kontekstu separacije izvora, filtriranje se može koristiti primjenom prozora specifičnog oblika. Prvo se nad originalnim signalom filtriranje provodi pomoću prozora uskog s obzirom na frekvenciju, a širokog s obzirom na vrijeme. Time se izoliraju horizontalne strukture, odnosno aspekti zvučnog signala kontinuirane frekvencije kroz vrijeme. Zatim se isto radi s prozorom obrnutog oblika koji će očuvati vertikalne strukture. Razlog zašto metoda funkcionira leži u teoretskoj pozadini glazbe i zvuka koji instrumenti proizvode.

Ritmički instrumenti poput bubnjeva stvaraju zvuk kraćega trajanja raširen i kontinuiran po frekvencijskom spektru, svojstvo koje prikazom u spektrogramu rezultira užom, okomito orijentiranom "linijom" - lokaliziran je s obzirom na vrijeme, a zauzima velik dio frekvencijskog spektra. Harmonički instrument poput gitare, s druge strane, proizvest će zvuk duljega trajanja i sporije opadajuće amplitude kroz vrijeme, gdje glavnini amplitude doprinosi fundamentalna frekvencija, te frekvencije dobivene mno-

ženjem fundamentalne frekvencije cjelobrojnim faktorom - harmonijska serija. Prikazan spektrogramom, zvuk takvog instrumenta poprimit će oblik niza horizontalnih linija različitih frekvencija donekle konzistentnih kroz vrijeme vertikalno razmaknutih za cjelobrojni faktor fundamentalne frekvencije.

Rezultat prvog koraka algoritma su maske koje, množenjem s originalnim spektrogramom, stvaraju dva nova spektrograma, jedan od kojih bi u teoriji trebao sadržavati razdvojene harmoničke zvukove odnosno instrumente, dok će u drugom biti sadržani ritmički elementi. U praksi jednostavnost metode ne rezultira pretjeranom robusnošću, te kasnije razvijene metode pokzauju bolje rezultate.

3.2. REPET

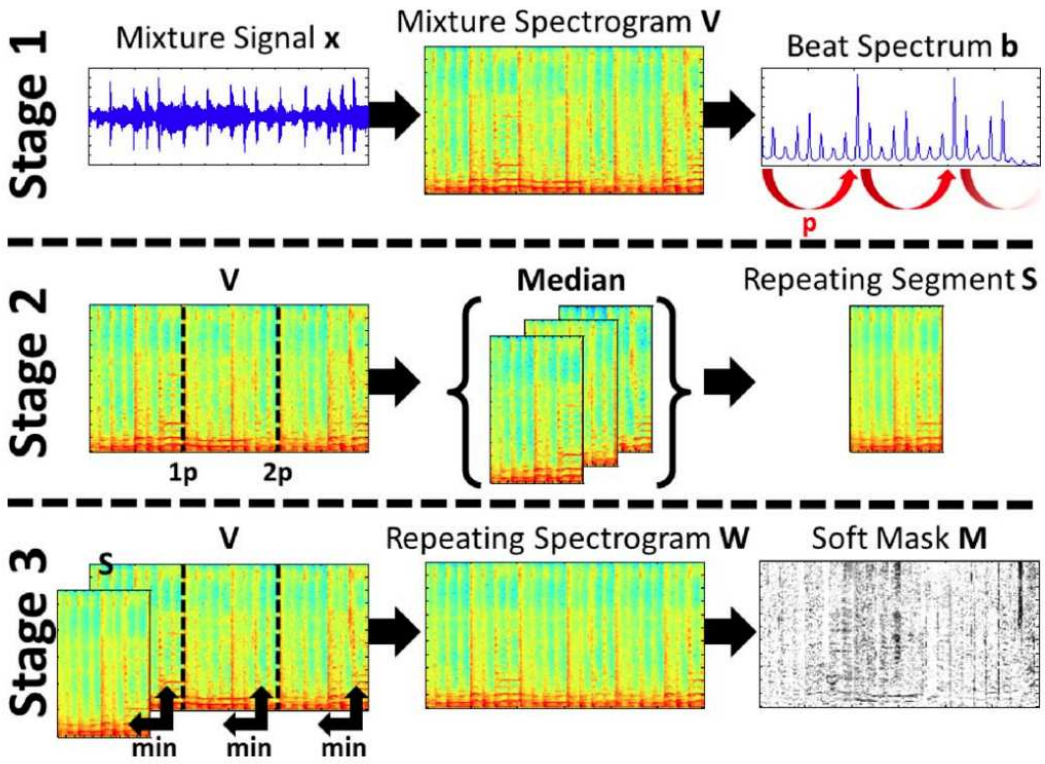
REPET (*REPeating Pattern Extraction Technique*) [14] još jedan je primjer metode separacije glazbenih izvora temeljene na jednostavnom svojstvu tipičnom za glazbeno djelo. Velika većina glazbenih djela sastoji se od kombinacije pozadinskih, ponavljajućih elemenata, te jednog ili više elemenata koji variraju kroz djelo te se nalaze u prvom planu. Jednostavan primjer toga je kombinacija instrumenata koji kroz djelo održavaju melodiju, dok vokalni dio pjesme različitim riječima i tonovima pjesmi daje strukturu i jedinstvenost.

Ugrubo, metoda REPET sastoji se od tri koraka: pronalazak ponavljajućeg perioda, modeliranje ponavljajućeg segmenta, te izvlačenje ponavljajućih uzoraka. Grafički prikaz procesa prikazan je na slici 3.2

Prvi korak, pronalazak ponavljajućeg perioda, obavlja se uz pomoć autokorelacije, mjere sličnosti između nekog vremenskog segmenta i segmenata istog oblika pomaknutih za određeni vremenski skok. To rezultira matricom, na temelju koje se izračunava "spektar otkucaja" b. Spektar otkucaja daje uvid u akustičnu ponavljajuću strukturu pjesme, te se iterativnom metodom odabira perioda p pronalazi optimalni period.

U drugom koraku pjesma se podijeli u ponavljajuće segmente pomoću ranije izračunatog perioda p te se elemnt po element izračunava njihov medijan. Rezultat toga je model ponavljajućeg segmenta koji se koristi u trećem koraku.

U završnom koraku REPET metode izračunava se ponavljajući spektrogram na temelju minimuma između svakog elementa modela ponavljajućeg segmenta i svakog od perioda duljine p u originalnom spektrogramu. Nakon toga izračunava se maska čijom se primjenom na originalni spektrogram dobiva spektrogram pozadinskih zvukova. Spektrogram zvukova u prvom planu (vokala ili glavnih instrumenata) dobiva se



Slika 3.2: Grafički prikaz algoritma REPET[14]

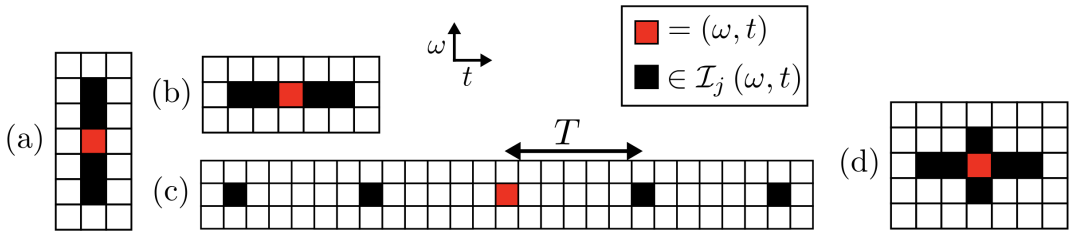
jednostavno oduzimanjem spektrograma pozadinskih zvukova od originalnog spektrograma.

Slično metodi *median filtering*, rezultati koje jednostavna metoda kao REPET proizvodi ukazuju na validnost ideje u određenoj mjeri, no u usporedbi s novim metodama razvijenim posljednjih godina rezultati bez dodatne dorade nemaju kvalitetu koja bi dozvolila daljnje korištenje dobivenih zvučnih datoteka.

3.3. Kernel Additive Models

Vratimo se nakratko na metodu *median filtering*. Metoda se zasniva na činjenici da su harmonički instrumenti horizontalne komponente na spektrogramu, dok su ritmički instrumenti vertikalne komponente. U suštini, harmonički instrumenti pokazuju kontinuitet kroz vrijeme, dok ritmički instrumenti pokazuju kontinuitet kroz frekvenciju. Metoda *kernel additive models* koristi se lokalnim značajkama koje spektrogrami glazbe pokazuju poput upravo navedenog kontinuiteta, periodičnosti kroz vrijeme (slično principu kojim se vodi metoda REPET) i drugima[3]. Metoda *kernel additive models* zasniva se na odabiru *kernel-a* kojim se modelira neka od ranije nave-

denih značajki glazbenog izvora, određivanjem amplitude pojedinog izvora u svakom trenutku premještanjem *kernel-a* na različite vremensko-frekvencijske košare te pospješivanjem procjena kroz više iteracija. Nekoliko primjera različitih *kernel-a*, svakog sa svojom svrhom pri obavljanju separacije, prikazano je na slici 3.3. Na ovaj način, metoda *kernel additive models* može se smatrati nadogradnjom na jednostavnije metode poput *median filtering* i REPET, pošto se osnovne pretpostavke tih metoda mogu reprezentirati odgovarajućim *kernel-ima*.



Slika 3.3: Grafički prikaz nekoliko različitih *kernel-a*[1]

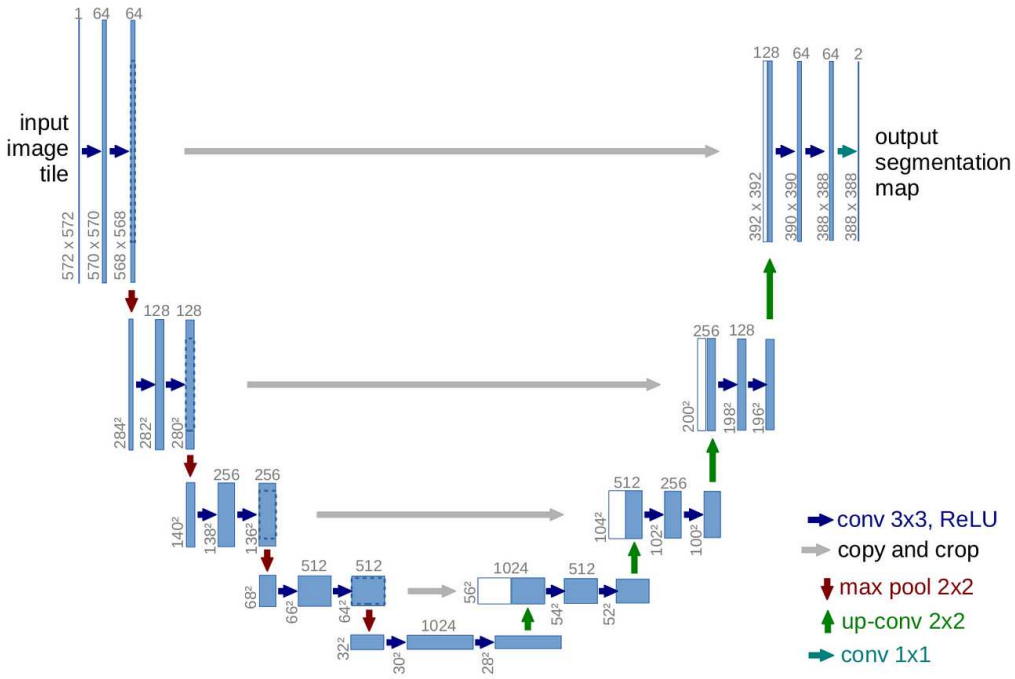
3.4. Metode dubokog učenja

Posljednjih godina strojno učenje, a posebno duboko učenje i neuronske mreže, česta su tema razgovora, a i odabir za potencijalna rješenja kompleksnih problema pomoću računala. Naravno, u području procesiranja zvuka, specifično u tematici razdvajanja glazbenih izvora, strojno učenje predloženo je kao potencijalno rješenje tog kompleksnog problema, te su posljednjih godina razne arhitekture neuronskih mreža te modeli strojnog i dubokog učenja korišteni upravo u tu svrhu. Danas, duboko učenje producira trenutno najviše obećavajuće rezultate pri rješavanju tog problema, a postojeće arhitekture i rješenja konstantno evoluiraju i dodatno se razvijaju. Tipično, slično dosad razmatranim metodama, kao ulazna informacija za neuronske mreže koristi se signal u domeni vrijeme-frekvencija (spektrogram), no neka rješenja razvijena su sa valnim oblikom kao ulaznom točkom. Neke od poznatijih razvijenih arhitektura i rješenja navedena su i objašnjena niže.

3.4.1. U-nets

U-nets originalno su razvijene pri rješavanju zadatka segmentacije slika, no kasnije su prenamijenjene u različite svrhe, uključujući pri rješavanju problema separacije glazbenih izvora. U-mreže sastoje se od 2 glavna dijela: enkodera i dekodera. Prvi dio

mreže sastoji se od niza 2D konvolucija koje korak po korak smanjuju dimenzionalnost mreže, dok se u drugom dijelu obavlja suprotan zadatak - niz 2D dekonvolucija povećava dimenzionalnost dok se na kraju ne dobije maska koja, pomnožena originalnim spektrogramom zvučnog signala, daje aproksimaciju zvučnog signala željenog izvora. Grafički prikaz arhitekture U-mreže vidljiv je na slici 3.4. Razvijene su i varijante U-mreža koje mogu odvojiti više izvora odjednom[6].

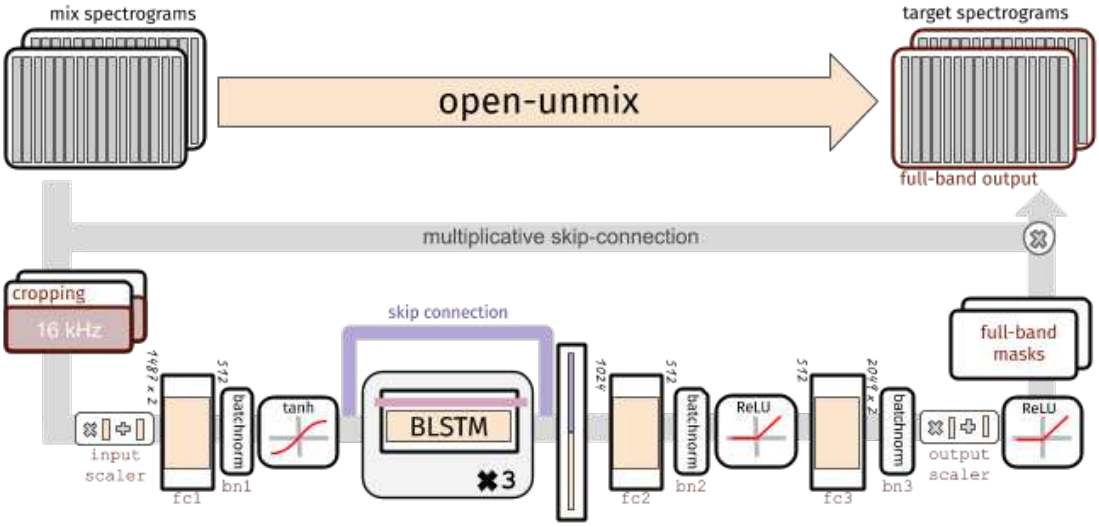


Slika 3.4: Arhitektura U-mreže [15]

Poznat komercijalni primjer rješenja zasnovanom na U-mrežama je *Spleeter* koji je razvila tvrtka Deezer.

3.4.2. Open-Unmix

Open-Unmix novija je aritektura zasnovana na kombinaciji potpuno povezanih (*fully connected*) slojeva te BLTSM (*bidirectional long short term memory*) slojeva sa *skip* vezama. Navedena arhitektura prikazana je na slici 3.5. Open-Unmix, uz *median filtering* i REPET, jedna je od dostupnih metoda za separaciju implementiranih u sklopu web aplikacije za separaciju izvora realizirane u sklopu ovog rada.



Slika 3.5: Arhitektura Open-Unmix [6]

3.4.3. Mask inference

Još jedna arhitektura koja koristi BLTSM slojeve, *mask inference* oblik je mreže koji se tipično sastoji od nekoliko BLTSM slojeva povezanih s potpuno povezanim slojem koji po završetku aproksimacije stvara masku dimenzija koje odgovaraju ulaznom spektrogramu.

3.4.4. Deep clustering

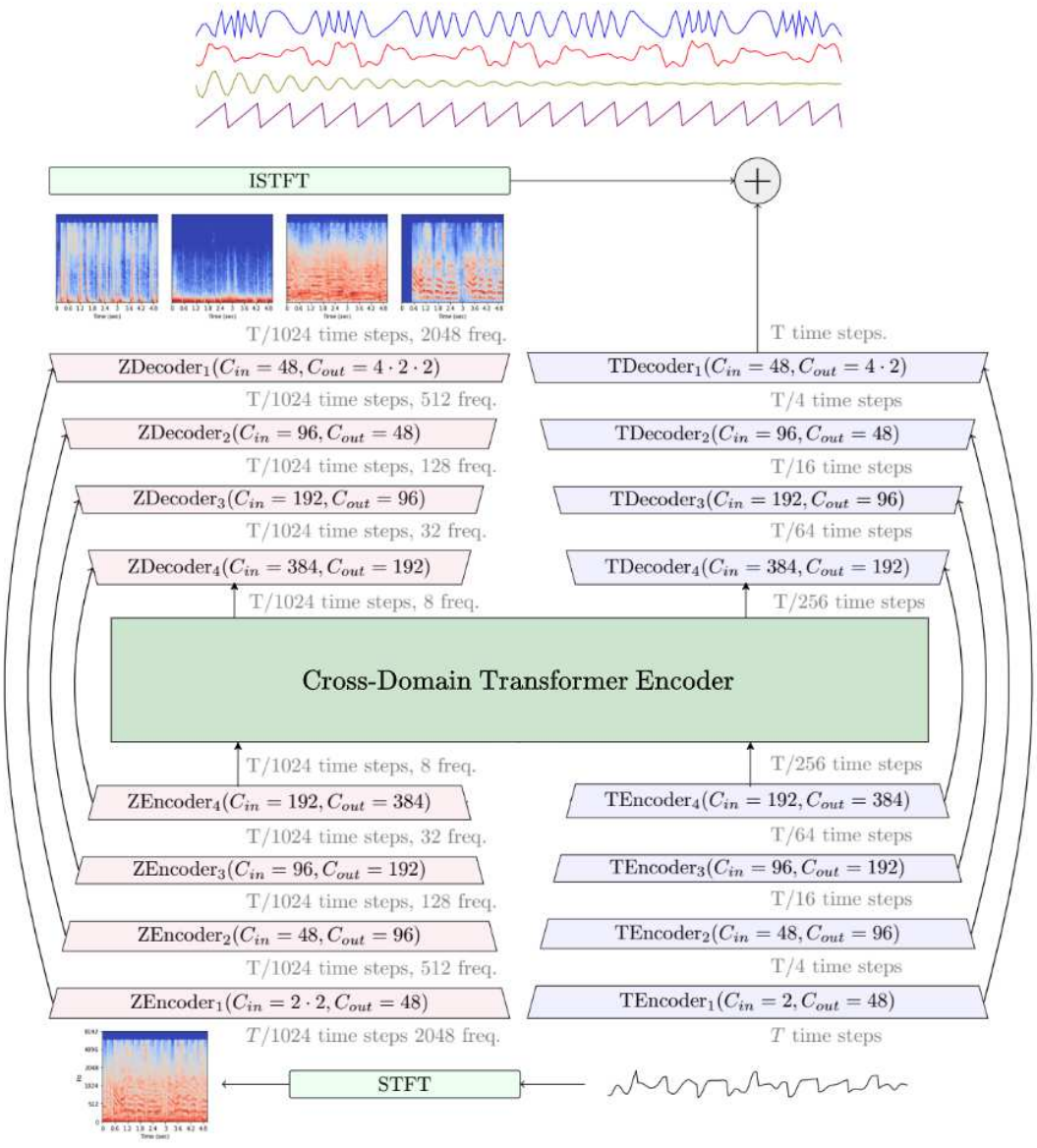
Deep clustering metoda je koja mapira vremensko-frekvencijske košare iz spektrograma u višedimenzionalni prostor, te ih grupira prema njihovoj međusobnoj udaljenosti u novom prostoru. Arhitektura je slična arhitekturi *mask inference* metode, a grupiranje u maske se tipično provodi odvojenim algoritmom (poput *k-means*) u posljednjem koraku.

3.4.5. Wave U-net

Wave U-net nadogradnja je na već spomenutu arhitekturu U-mreže, koja umjesto dvodimenzionalnih konvolucija koristi jednodimenzionalne konvolucije, primijenjene izravno na valni oblik (*waveform*) ulaznog zvučnog signala, svojstvo po kojem je arhitektura i dobila naziv.

3.4.6. Demucs

Demucs je moderna mreža razvijena od strane Facebook Research Lab, arhitekture koja dijeli sličnosti s U-net mrežom. Originalna verzija također se sastojala od konvolucijskih i dekonvolucijskih slojeva, no u sredini mreže nalazila su se 2 BLTSM sloja. Novija verzija Demucs arhitekture hibridna je, gdje se u ulaznom dijelu iskorištavaju i valni oblik i spektrogram, a najnovije verzije koriste se transformerima razvijenim i populariziranim u području obrade prirodnog jezika. Arhitektura je prikazana na slici 3.6



Slika 3.6: Arhitektura Hybrid Transformer Demucs[12]

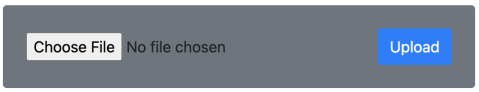
4. Aplikacija za razdvajanje glazbenih izvora

U sklopu rada razvijena je aplikacija s nekoliko funkcionalnosti:

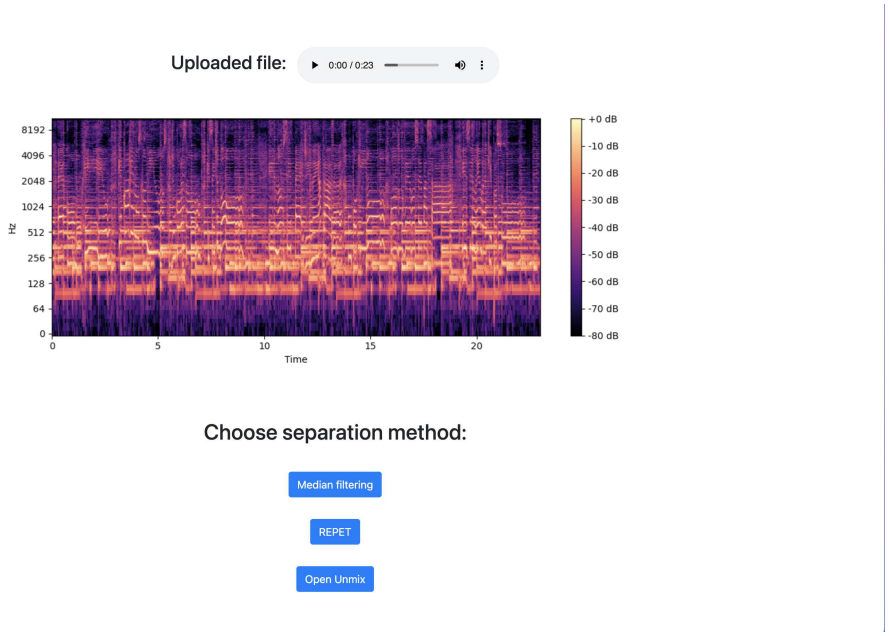
- učitavanje lokalnih audio datoteka u aplikaciju vidljivo na slici 4.1
- prikaz spektrograma datoteke nakon učitavanja vidljivo na slici 4.2
- nekoliko ponuđenih opcija za metodu separacije vidljivo na slici 4.2
- pregled spektrograma odvojenih elemenata pjesme vidljivo na slici 4.3
- preslušavanje i spremanje odvojenih elemenata pjesme vidljivo na slici 4.3

Aplikacija je realizirana u obliku web aplikacije na sličan način kao druge, postojeće, komercijalne aplikacije za separaciju glazbenih izvora. Uobičajeni tijek rada je sljedeći:

1. korisnik otvara web stranicu u internetskom pregledniku
2. korisnik učitava željenu audio datoteku u sučelje za učitavanje
3. originalna audio datoteka se procesira, generira se spektrogram i prikazuje korisniku uz sučelje za reprodukciju datoteke
4. korisnik klikom na željeni gumb odabire metodu separacije datoteke u različite elemente
5. nakon procesiranja, prikazuju se spektrogrami generirani za svaki izvor dobiven odabranom metodom separacije uz sučelja za reprodukciju i spremanje novonastalih audio datoteka za svaki izvor
6. za istu originalnu audio datoteku moguće je obaviti separaciju svakom od ponuđenih metoda
7. za ponavljanje cijelog procesa dovoljno je ponovno učitati web stranicu na kojoj se nalazi aplikacija



Slika 4.1: Sučelje za učitavanje audio datoteka

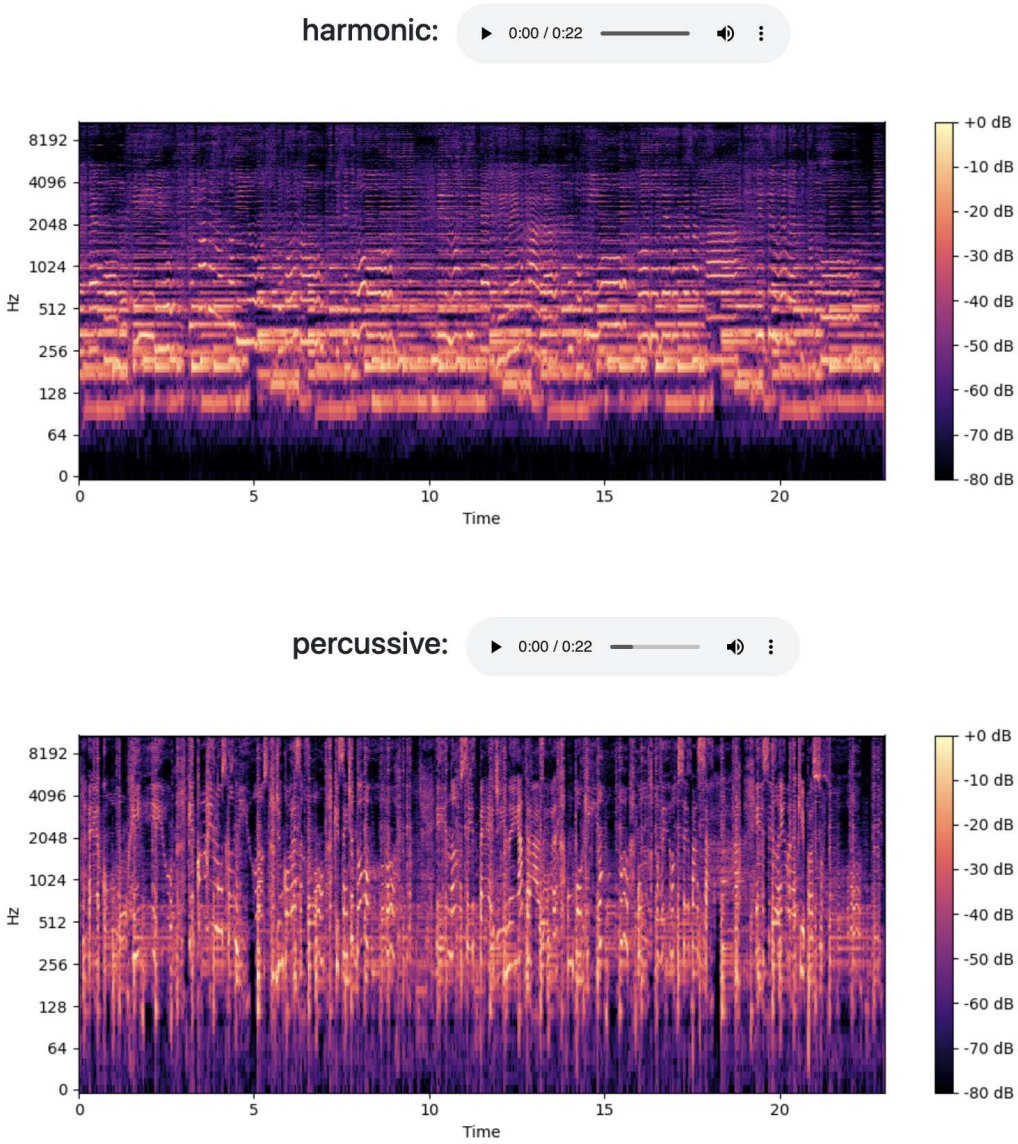


Slika 4.2: Rezultat učitavanja i sučelje za odabir metode razdvajanja

Kao što je već rečeno, rješenje je implementirano u obliku web aplikacije. Aplikacija se sastoji od nekoliko sastavnih dijelova: poslužiteljske strane, klijentske strane te skripti za separaciju i generiranje spektrograma na poslužiteljskoj strani. U nastavku su pobliže objašnjeni dijelovi aplikacije, način na koji su implementirani i tehnologije koje su korištene.

4.1. Poslužitelj

Poslužiteljski dio aplikacije realiziran je pomoću programskog jezika Golang. Golang je statički tipiziran, kompajliran programski jezik razvijen u Google-u 2009. Njegove glavne značajke su memorijska sigurnost zahvaljujući *garbage collection*-u, jednostavna sintaksa koja naliči na C, te odlična podrška za konkurentost programa pomoću *goroutines*.



Slika 4.3: Spektrogrami i audio zapisi nakon razdvajanja metodom median filtering

4.1.1. URL preusmeravanje

Za URL *routing* koristi se paket chi[2]. Paket chi omogućuje jednostavno usmjeravanje temeljeno na URL-u, grupiranje ruta te dinamički *routing* temeljen na vrijednostima URL-a. U ovom slučaju, dinamički *routing* koristan je za GET dohvate koji započinju separaciju odabranom metodom. Naziv metode ubacuje se u URL s kojeg se dohvaćaju podaci, što poziva odgovarajuću skriptu na poslužitelju koja obavlja separaciju te vraća tražene podatke.

4.1.2. Struktura Python

Ostatak poslovne logike na poslužiteljskoj strani uobičajena je implementacija *backend*-a u programskom jeziku Go, gdje se koriste sučelja i strukture koja implementiraju ta sučelja da bi se postigla željena razina apstrakcije u aplikaciji. Primjer toga je struktura *Python* koja u sebi sadrži putove do lokacija Python skripti te omogućuje pokretanje istih u svrhu obavljanja separacije i generiranja spektrograma. Programski kod strukture prikazan je na slici 4.4

```go
type Python struct {
    execPath        string // path to python executable
    scriptsPath     string // path to scripts folder
    uploadsPath     string // path to uploads folder
    spectrogramPath string // path to spectrogram.py
}

func NewPython(basePath string) *Python {
    execPath := path.Join(basePath, "env", "bin", "python3")
    scriptsPath := path.Join(basePath, "scripts")
    uploadsPath := path.Join(basePath, "uploads")
    spectrogramPath := path.Join(scriptsPath, "spectrogram.py")

    return &Python{
        execPath:        execPath,
        scriptsPath:     scriptsPath,
        uploadsPath:     uploadsPath,
        spectrogramPath: spectrogramPath,
    }
}

func (p *Python) Spectrogram(wavPath string) error {
    cmd := exec.Command(p.execPath, p.spectrogramPath, wavPath)

    out, err := cmd.CombinedOutput()
    log.Println(string(out))
    log.Println(err)
    if err != nil {
        return err
    }

    return nil
}

func (p *Python) Separate(method string) error {
    script := path.Join(p.scriptsPath, "separate", method+".py")

    cmd := exec.Command(p.execPath, script, p.uploadsPath)

    out, err := cmd.CombinedOutput()
    log.Println(string(out))
    log.Println(err)
    if err != nil {
        return err
    }

    return nil
}

```

Slika 4.4: Implementacija strukture Python

4.1.3. Konkurentno generiranje spektrograma

Jedna zanimljiva optimizacija koju omogućuje podrška za konkurentnost programskog jezika Go je višestruko pokretanje određene Python skripte u isto vrijeme. Tu opciju omogućuju *goroutines*, strukture u programskom jeziku Go nalik dretvama koje omogućuju konkurentno izvođenje zadataka. U trenutku kad skripta za separaciju završi svoj posao te su na poslužiteljskoj strani generirane i spremljene odvojene audio datoteke s pojedinim izvorima, aplikacija prolazi po specificiranom direktoriju čije ime odgovara metodi separacije, te za svaku pronađenu audio datoteku u direktoriju pokreće zasebu Go rutinu unutar koje se poziva skripta za generiranje spektrograma. Ta "paralelizacija" generiranja spektrograma skraćuje ukupno vrijeme potrebno za procesiranje korisničkog zahtjeva za separacijom te pospješuje bolje korisničko iskustvo korištenja aplikacije. Sinkronizacija rutina po završetku posla ostvarena je pomoću strukture *errgroup* iz proširenja standardne biblioteke programskog jezika Go naziva *x*. Programski kod s implementacijom generiranja spektrograma u zasebnim dretvama prikazan je na slici 4.5

```
59  func (s *Separator) generateSpectrograms(method string) error {
60      dirPath := filepath.Join(s.uploadsPath, method)
61      errGroup := new(errgroup.Group)
62
63      filepath.WalkDir(dirPath, func(path string, d fs.DirEntry, err error) error {
64          fmt.Println("PATH:", path)
65          if util.IsWav(d) {
66              errGroup.Go(func() error {
67                  fmt.Println("CALLING SPECTROGRAM:", path)
68                  err := s.python.Spectrogram(path)
69                  return err
70              })
71          }
72          return nil
73      })
74
75      if err := errGroup.Wait(); err != nil {
76          return err
77      }
78
79      return nil
80  }
81
```

Slika 4.5: Generiranje spektrograma u zasebnim rutinama

4.2. Klijent

Klijentska strana aplikacije, korisničko sučelje pomoću kojeg korisnik obavlja funkcionalnosti aplikacije, ostvareno je kombinacijom dviju tehnologija: go HTML predlošci i *htmx*.

4.2.1. Go predlošci - *templates*

Prvi dio, HTML predlošci (*templates*), funkcionalnost su ugrađena u programski jezik Go. Predlošci se definiraju kao uobičajene HTML datoteke, koje se pomoću paketa *html/template* u standardnoj biblioteci programskog jezika Go parsiraju pri kompajliranju samog programa. U predloške se pri posluživanju klijentu mogu dinamički unijeti željeni podaci da se odgovarajući sadržaj prikaže na klijentskoj strani. Primjer HTML predloška korištenog u radu prikazan je na slici 4.6.

```
<div class="col-md-12 d-flex justify-content-center align-items-center m-3">
    <p class="h3 mr-3">Uploaded file:</p>
    <audio controls>
        <source src="{{.AudioPath}}" type="audio/wav">
        Your browser does not support the audio element.
    </audio>
</div>
<div class="col-md-12 d-flex justify-content-center align-items-center m-3">
    <img src="{{.ImagePath}}" alt="Spectrogram" class="img-fluid">
</div>

<hr />

<div class="col-md-12 d-flex justify-content-center align-items-center m-3">
    <p class="h2">Choose separation method:</p>
</div>

<div id="median" class="col-md-12 d-flex justify-content-center align-items-center m-3">
    <button class="btn btn-primary" hx-get="/separate/median" hx-target="#median" hx-swap="outerHTML">Median filtering</button>
</div>

<div id="repet" class="col-md-12 d-flex justify-content-center align-items-center m-3">
    <button class="btn btn-primary" hx-get="/separate/repet" hx-target="#repet" hx-swap="outerHTML">REPET</button>
</div>

<div id="umx" class="col-md-12 d-flex justify-content-center align-items-center m-3">
    <button class="btn btn-primary" hx-get="/separate/unmix" hx-target="#umx" hx-swap="outerHTML">Open Unmix</button>
</div>
```

Slika 4.6: HTML preložak za korisničko sučelje s opcijama za separaciju

4.2.2. htmx

Druga tehnologija koja omogućuje dinamičko korisničko sučelje u obliku SPA (*Single Page Application*) je biblioteka *htmx*. *Htmx* je nova, moderna biblioteka alternativnog dizajna u usporedbi s u posljednje vrijeme često korištenim JavaScript bibliotekama i *framework-ovima* za izradu korisničkog sučelja koje se temelje na prijenosu podataka s poslužitelja u obliku JSON ili drugih formata za prijenos podataka. *Htmx* umjesto

toga s poslužitelja direktno prenosi HTML, koji se pomoću atributa definiranim u elementima web stranice dinamički prikazuju u korisničkom sučelju, nadomještavajući, dodavajući nove ili zamjenjujući postojeće elemente.

Primjer: u aplikaciji se biblioteka *htmx* koristi na gumbima koji nakon klika poslužitelju šalju zahtjev za procesiranjem učitane audio datoteke, obavljanjem separacije nad istom, generiranjem spektrograma za svaki razdvojeni izvor te prikazom dobivenih informacija korisniku. Po završetku procesiranja korisničkog zahtjeva, poslužitelj klijentu dostavlja HTML koji se ubacuje u postojeći raspored web stranice, zamjenjujući gumb koji je pritisnut na početku zahtjeva. Time je omogućen dinamički prikaz podataka na klijentskoj strani. Primjeri korištenja *htmx* atributa za postizanje navedene funkcionalnosti vidljivi su na slici 4.6 na linijama 19, 23 i 27.

4.3. Python skripte

Centralni dio aplikacije, odvajanje glazbenih izvora iz predane audio datoteke, ostvareno je u obliku niza skripti napisanih u programskom jeziku Python, svaka od kojih je zaslužna za obavljanje separacije metodom prema kojoj je skripta nazvana. Osim skripti koje obavljaju separaciju, stvorena je i skripta koja iz predane audio datoteke generira spektrogram te sprema sliku spektrograma na odgovarajuću lokaciju u datotečnom sustavu poslužitelja.

Pri izradi skripti korišteni su različiti Python paketi koji su omogućili pojedine dijelove njihove funkcionalnosti. Korišteni paketi navedeni su i objašnjeni niže.

4.3.1. librosa

Librosa[5] je Python paket čija je primarna namjena analiza i procesiranje zvučnih datoteka. Sadrži razne funkcije koje olakšavaju implementaciju kompleksnijih algoritama bez potrebe za direktnom implementacijom algoritama niže razine koji se često pojavljuju tijekom rada sa zvučnim signalima. U kontekstu ovog rada, paket librosa korišten je u skripti za generiranje spektrograma za učitavanje audio datoteke, obradu pomoću STFT te za pretvaranje Y osi spektrograma iz amplitudne skale u decibelnu. Paket librosa također sadrži implementaciju metode *median filtering* koja je korištena u skripti za separaciju istoimenom metodom.

4.3.2. Matplotlib

Matplotlib[7] je poznata Python biblioteka za stvaranje vizualizacija raznih vrsta i oblika. U ovom radu koristi se prilikom generiranja i spremanja spektrograma originalne učitane audio datoteke i audio datoteka nastalih separacijom.

4.3.3. NumPy

NumPy[8] je Python biblioteka koja omogućuje rad s višedimenzionalnim listama i matricama, obavljanje raznih matematičkih operacija nad njima te razne druge funkcionalnosti. U sklopu ovog rada korištena je na mnogo mjesta s obzirom da je osnovna reprezentacija učitane audio datoteke reprezentirana u obliku NumPy matrice.

4.3.4. PyTorch

PyTorch[11] je još jedna Python biblioteka korištena u sklopu rada čija je primarna namjena pomoć pri implementaciji rješenja koja koriste strojno učenje. Sadrži razne pomoćne funkcije korištene u području strojnog učenja te se pomoću te biblioteke često realiziraju modeli strojnog i dubokog učenja, odnosno neuronske mreže.

4.3.5. nussl

Nussl[9] je sveobuhvatna biblioteka stvorena u *Interactive Audio Lab* na Northwestern University, Il, SAD. U biblioteci su sadržane implementacije raznih metoda razdvajanja izvora, od osnovnijih poput *median filtering* i REPET, do gotovih, naučenih modela strojnog učenja. Osim gotovih implementacija, u sklopu biblioteke sadržane su i razne mogućnosti niže razine za pomoć pri učenju vlastitih modela, evaluaciji metoda, procesiranju zvučnih datoteka i mnoge druge. U ovom radu iz biblioteke nussl korištena je implementacija REPET metode za razdvajanje izvora.

4.3.6. OpenUnmix

openunmix-pytorch[10] je implementacija Open-Unmix pomoću biblioteke PyTorch dostupna na javnom GitHub repozitoriju. Korištena je kao implementacija istoimene metode za separaciju u sklopu projekta.

5. Testiranje rezultata

Aplikacija razvijena u sklopu rada testirana je prema dvjema različitim metrikama. Prva metrika, performanse, tiče se izvođenja samog programa te obavljanja separacije. Druga metrika, kvaliteta separacije, tiče se metoda razdvajanja glazbenih izvora koje su u sklopu aplikacije implementirane.

5.1. Performanse

Performanse aplikacije izmjerene su na poslužiteljskoj strani. Pri dolasku klijentskog zahtjeva za pokretanjem separacije bilo kojom od ponuđenih metoda zapisano je trenutno sistemsko vrijeme na poslužitelju. Nakon što je skripta za separaciju završila obradu te su na poslužitelju spremljenje novonastale audio datoteke, izračunato je proteklo vrijeme od pokretanja skripte. Performanse metoda su mjerene za 2 različite ulazne audio datoteke, jednu duljine 23 sekune, te drugu duljine 3 minute i 7 sekundi. Cilj mjerenja s razičitim duljinama zvučnih zapisa i samim time veličinama audio datoteka je bio bolje prikazivanje razlika u performansama pojedinih metoda. Svaka od metoda, za svaku od dvije datoteke, pokrenuta je 5 puta, te su rezultati zapisani u tablici niže, uz dodatno izračunato prosječno trajanje izvođenja za svaku kombinaciju metode separacije i audio datoteke.

Iz prikazane tablice vidljive su velike razlike u performansama, neke očekivane a neke manje. Metoda *median filtering* računski je najjednostavnija od implementiranih te izmjerena vremena to jasno ističu. Metoda je redovito najbrže obavljena neovisno o duljini ulazne audio datoteke. Algoritam metode REPET sastoji se od više iteracija pri odabiru optimalnog perioda ponavljanja, što utječe na performanse metode. Potencijalni faktor koji također negativno utječe na brzinu je implementacija metode u sklopu biblioteke *nussl*. Metoda *OpenUnmix* očekivano je sporija od metode *median filtering* zbog potrebe za učitavanjem modela u memoriju pri pokretanju, te činjenici da se skripta izvršavala bez GPU akceleracije. U daljnjem razvoju aplikacije bilo bi poželjno postići GPU akceleraciju kojom bi se ova i slične metode temeljene na mo-

Tablica 5.1: Mjerenja trajanja separacije implementiranim metodama

Trajanje	23s			3m7s		
Mjerenje	Median	REPET	Unmix	Median	REPET	Unmix
1.	4.17s	14.03s	16.00s	15.80s	149.27s	150.03s
2.	3.73s	10.79s	14.81s	11.88s	147.17s	103.18s
3.	3.81s	11.00s	14.69s	15.29s	144.49s	108.22s
4.	3.71s	10.87s	14.73s	12.10s	149.27s	103.83s
5.	3.86s	10.91s	14.90s	15.09s	145.93s	103.68s
Prosjek	3.86s	11.52s	15.03s	14.03s	147.22s	113.79s

delima strojnog ili dubokog učenja znatno ubrzale.

5.2. Kvaliteta separacije

Drugi aspekt po kojem su metode evaluirane je kvaliteta zvuka, odnosno kvaliteta separacije samih izvora. U znanstvenoj zajednici kroz godine su se koristilo i razvilo više metoda i metrika evaluacije kvalitete razdvojenih izvora, neke od kojih su SDR (*Signal to Distortion Ratio*), SIR (*Signal to Interference Ratio*) te kompleksnije metode koje ujedinjuju više metrika te su korištene u sklopu otvorenih natjecanja u razvoju metoda razdvajanja izvora, poput *bsseval*[13].

S obzirom da su metode implementirane u sklopu ovog rada poznate, te su nad njima više puta kroz povijest obavljene i objavljene evaluacije raznim metodama uključujući prije navedene, u sklopu ovog rada nije provedena takva vrsta evaluacije metoda. Umjesto toga, kvaliteta metoda separacije procijenjena je subjektivnim, slušnim testom u sklopu kojeg su preslušane originalna te sve pojedinačne novonastale audio datoteke s razdvojenim izvorima. Rezultati testa navedeni su niže.

Za potrebe slušnog testa kao ulazna audio datoteka odabrane su dvije audio datoteke, identične onima korištenima pri testiranju performansi aplikacije. Dulja od njih je pjesma duljine 3 minute i 7 sekundi, stvorena u DAW programu FL Studio. Pjesma pripada *house* žanru s uobičajenom pripadajućom glazbenom strukturom koja se sastoji od uvoda, prve strofe, prvog refrena, druge strofe, drugog refrena te outro-a. Pjesma sadrži kombinaciju raznih ritmičkih instrumenata poput bubnjeva, kastanjeta i klikova, uz druge instrumente poput klavira i truba, te ženskog vokala. Druga pjesma duljine 23 sekunde je javno dostupan isječak iz pjesme *Que Pena Tanto Faz* preuzet iz GitHub repozitorija implementacije REPET[14] metode u Python programskom jeziku

korištene u sklopu rada. Ova je pjesma podosta jednostavnija, sastoji se od melodije svirane na gitari te ženskog vokala.

Slušnim testom potvrđene su pretpostavke iz poglavlja 3 ovog rada vezane za implementirane metode separacije - jednostavnije, ranije razvijene metode, u ovom slučaju *median filtering* i REPET, u usporedbi s modernim metodama temeljenim na strojnom učenju poput *Open Unmix* ne pokazuju dovoljnu robustnost s obzirom na veliku varijantnost i kompleksnost gotovog glazbenog djela, te same po sebi ne produciraju rezultate pogodne za daljnje primjene kao što je poboljšavanje kvalitete i renovacije zastarjelih snimaka pjesama ili stvaranje *remix-eva*. Moderne metode poput *Open Unmix*, nasuprot tomu, produciraju zvučne datoteke s razdvojenim glzbenim izvorima zadovoljavajuće kvalitete, koji se potencijalno, naravno uz dodatnu obradu radi poboljšanja kvalitete, mogu koristiti u netom prije navedene svrhe.

6. Zaključak

U ovom radu proučena je problematika razdvajanja glazbenih izvora, proučene i objašnjene su metode postizanja iste, te je realizirana i testirana web aplikacija čija je funkcionalnost omogućavanje separacije izvora koristeći više ranije istraženih metoda. Testiranjem aplikacije utvrđeno je da ranije razvijene metode separacije ne ostvaruju zadovoljavajuće rezultate za daljnju primjenu nastalih zvučnih datoteka, no novije metode temeljene na strojnom učenju daju obećavajuće rezultate.

Pregledom ostvarenih rezultata vidljivo je više potencijalnih unaprijeđenja u funkcionalnosti aplikacije. Jedno od mogućih unaprijeđenja korištenje računala s diskretnom grafičkom karticom kao poslužitelja te korištenjem paralalelizacije u računski intenzivnim dijelovima algoritama. Nadalje, potencijalno korisna opcija bi bila trajno spremanje učitanih i novonastalih zvučnih datoteka te omogućavanje korisniku pristupa u kasnijem trenutku u slučaju potrebe. Osim navedenog, glavno unaprijeđenje koje se ističe jest implementacija dodatnih metoda za razdvajanje glazbenih izvora koje bi korisnicima dalo veće mogućnosti i slobodu pri odabiru odgovarajuće metode.

LITERATURA

[1] Zafar Rafii Bryan Pardo Laurent Daudet Antoine Liutkus, Derry Fitzgerald. Kernel additive models for source separation. U *IEEE Transactions on Signal Processing*, 2014.

[2] chi router. https://github.com/go-chi/chi.

[3] Antoine Liutkus Mark D. Plumbley Fabian-Robert Stöter Estefania Cano, Derry Fitzgerald. Source separation: An introduction. *IEEE Signal Processing Magazine*, 2019.

[4] D. FitzGerald. Harmonic/percusssive separation using median filtering. U *Proceedings of the 13th International Conference on Digital Audio Effects*, 2010.

[5] librosa. https://librosa.org/.

[6] Ethan Manilow, Prem Seetharman, i Justin Salamon. *Open Source Tools & Data for Music Source Separation*. https://source-separation.github.io/tutorial, 2020.

[7] Matplotlib. https://matplotlib.org/.

[8] NumPy. https://numpy.org/.

[9] nussl. https://github.com/nussl/nussl.

[10] Open-Unmix PyTorch. https://github.com/sigsep/open-unmix-pytorch.

[11] PyTorch. https://pytorch.org/.

[12] Simon Rouard, Francisco Massa, i Alexandre Défossez. Hybrid transformers for music source separation. U *ICASSP 23*, 2023.

[13] Fabian-Robert Stöter, Antoine Liutkus, i Nobutaka Ito. The 2018 signal separation evaluation campaign. U *Latent Variable Analysis and Signal Separation:*

14th International Conference, LVA/ICA 2018, Surrey, UK, stranice 293–305, 2018.

[14] B. Pardo Z. Rafii. Repeating pattern extraction technique (repet): A simple method for music / voice separation. U *IEEE Trans. on Audio, Speech and Language Processing*, 2013.

[15] Jeremy Zhang. Unet — line by line explanation. *Towards Data Science*, 2019.

Odvajanje ritmičkih elemenata od melodijskih u pjesmama

Sažetak

U sklopu rada proučene su i predstavljene neke od metoda razdvajanja glabenih izvora. U uvodu je objašnjena pozadina, osnovni problem i cilj separacije glazbenih izvora. Dalje je u radu naveden i pobliže objašnjen niz metoda i algoritama za postizanje separacije. U idućim poglavljima objašnjena je arhitektura web aplikacije koja omogućuje razdvajanje glazbenih izvora raznim metodama stvorene u sklopu ovog rada. Na poslijetku su objašnjene metode testiranja rezultata rada, rezultati testiranja te su navedena potencijalna buduća unaprijeđenja.

Ključne riječi: razdvajanje glazbenih izvora, glazba, spektrogram, neuronske mreže, analiza zvuka

Separating rhythmic and melodic elements of songs

Abstract

In this work, a number of music source separation methods are named and explained. The introduction of the work consists of an explanation of the area of music source separation. Further on, different methods for achieving music source separation are explained. A detailed explanation of a web application realized as part of this work that enables separation of music sources using various methods follows. Finally, the method of testing the result of the work is explained, together with the results themselves and ideas to improve on the work are listed.

Keywords: music source separation, music, spectrogram, neural networks, sound analysis