

# Razdvajanje izvora glazbe u zvukovnim zapisima primjenom dubokih neuronskih mreža

---

**Magat, Matej**

**Undergraduate thesis / Završni rad**

**2024**

*Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj:* **University of Zagreb, Faculty of Electrical Engineering and Computing / Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva**

*Permanent link / Trajna poveznica:* <https://urn.nsk.hr/urn:nbn:hr:168:371060>

*Rights / Prava:* [In copyright/Zaštićeno autorskim pravom.](#)

*Download date / Datum preuzimanja:* **2025-03-23**



*Repository / Repozitorij:*

[FER Repository - University of Zagreb Faculty of Electrical Engineering and Computing repository](#)



SVEUČILIŠTE U ZAGREBU  
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1518

**RAZDVAJANJE IZVORA GLAZBE U ZVUKOVNIM ZAPISIMA  
PRIMJENOM DUBOKIH NEURONSKIH MREŽA**

Matej Magat

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU  
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1518

**RAZDVAJANJE IZVORA GLAZBE U ZVUKOVNIM ZAPISIMA  
PRIMJENOM DUBOKIH NEURONSKIH MREŽA**

Matej Magat

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU  
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

ZAVRŠNI ZADATAK br. 1518

Pristupnik:	<b>Matej Magat (0036541718)</b>
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentor:	izv. prof. dr. sc. Goran Delač
Zadatak:	<b>Razdvajanje izvora glazbe u zvukovnim zapisima primjenom dubokih neuronskih mreža</b>

Opis zadatka:

Istražiti i opisati modele zasnovane na dubokim neuronskim mrežama prikladne za klasifikaciju izvora glazbe u zvukovnim zapisima. Poseban naglasak staviti na modele zasnovane na konvolucijskim neuronskim mrežama. Prikupiti prikladan skup zvukovnih zapisa u kojem je zabilježen zapis poznatog skupa mogućih izvora glazbe. Osmisliti i primjenski ostvariti sustav razdvajanja glazbe u zvukovnom zapisu. Opisati korišteni model i postupak učenja. Primjenom prikladno odabralih mjera za vrednovanje ispitati radna svojstva programskega ostvarenja.

Rok za predaju rada: 14. lipnja 2024.



# Sadržaj

<b>1. Uvod</b>	3
<b>2. Pristupi razdvajanju izvora zvuka</b>	4
2.1. Analiza neovisnih komponenti	4
2.2. Faktorizacija nenegativne matrice	5
2.3. Konvolucijske neuronske mreže	5
<b>3. U-Net arhitektura</b>	8
3.1. Dijelovi U-Net arhitekture	9
3.2. Koder	10
3.3. Usko grlo	11
3.4. Dekoder	12
<b>4. Zvuk kao ulaz u U-Net</b>	13
4.1. Kratkotrajna Fourierova transformacija	14
4.2. Griffin-Lim algoritam	15
<b>5. Programsко ostvarenje</b>	17
5.1. Programski jezik Python	17
5.2. PyTorch radni okvir	17
5.3. MUSDB18-HQ skup podataka	19
5.4. Google Colab	20
5.5. Ostvareno rješenje	21
5.5.1. Struktura projekta	21
5.5.2. Obrada zvuka	21
5.5.3. Implementacija U-Net arhitekture	23

5.5.4. Ulazni podaci . . . . .	25
5.5.5. Treniranje modela . . . . .	26
5.5.6. Predviđanje modela . . . . .	28
5.5.7. Uklanjanje šuma . . . . .	28
<b>6. Rezultati i rasprava . . . . .</b>	<b>29</b>
<b>7. Zaključak . . . . .</b>	<b>31</b>
<b>Literatura . . . . .</b>	<b>32</b>
<b>Sažetak . . . . .</b>	<b>34</b>
<b>Abstract . . . . .</b>	<b>35</b>

## 1. Uvod

Područje istraživanja pronalaženja glazbenih informacija (engl. Music Information Retrieval) bavi se na razvojem sustava klasifikacije glazbe, programske podrške za identifikaciju glazbe (npr. Shazam), generiranje glazbe (npr. Suno) i razdvajanje izvora zvuka (npr. Deezer Spleeter[1]). Budući da se glazba stvara na nedeterministički način, spanjanjem različitih zvukovnih zapisa, zadatci klasifikacije, razdvajanja i generiranja se ostvaruju različitim postupcima strojnog učenja i dubokog učenja. Višeslojne neuronske mreže kao što su konvolucijske neuronske mreže i transformatori se najčešće koriste u te svrhe.

Ovaj rad će opisati modele strojnog učenja zasnovane na konvolucijskim neuronskim mrežama prikladnim za klasifikaciju izvora glazbe u zvukovnim zapisima. Naglasak će se staviti na izgradnju, učenje i testiranje modela s U-Net arhitekturom.

## 2. Pristupi razdvajanju izvora zvuka

Neka skup K senzora, a  $\mathbf{x}(n)$  vektor mješavine snimljenih zvukova. [2]

$$\mathbf{x}(n) = [x_1(n), \dots, x_K(n)]^T, n = 1, 2, \dots, K \quad (2.1)$$

Skup K senzora promatra skup od L izvora zvuka.

$$\mathbf{s}(n) = [s_1(n), \dots, s_L(n)]^T, n = 1, 2, \dots, L \quad (2.2)$$

Prepostavlja se da svaki senzor dohvaca skaliranu verziju svakog signala bez kašnjenja u prijenosu. Tada se model miješanja izvora opisuje s:

$$\mathbf{x}(n) = \mathbf{A}\mathbf{s}(n) \quad (2.3)$$

$\mathbf{A}$  predstavlja matricu dimenzija K x L te se cilj razdvajanja izvora postize pronalaskom inverzne matrice  $\mathbf{A}^{-1}$  dimenzija L x K

$$\hat{\mathbf{s}}(n) = \mathbf{A}^{-1}\mathbf{x}(n) \quad (2.4)$$

gdje je  $\hat{\mathbf{s}}(n)$  skup procijenjenih izvora zvuka.

### 2.1. Analiza neovisnih komponenti

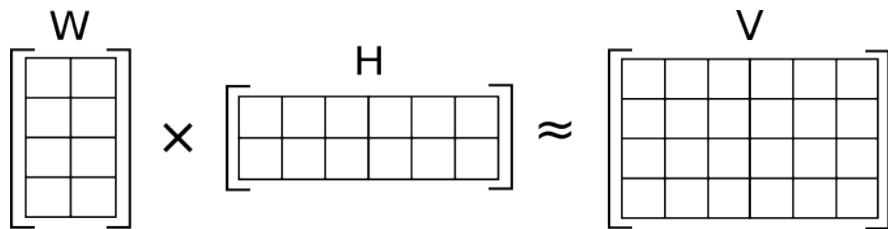
Analiza neovisnih komponenti prepostavlja neovisnost izvora i da podaci izvora ne pripadaju Gaussovoj razdiobi. Metoda pokušava riješiti jednadžbu (2.4) pronalaskom matrice  $U \approx A^{-1}$

Primjena analize neovisnih komponenti je "problem koktel zabave", gdje se temeljni govorni signali odvajaju od uzorka podataka koji se sastoji od ljudi koji istovremeno razgovaraju u prostoriji. Problem se pojednostavljuje pretpostavkom da nema vremenskih kašnjenja ili odjeka.

## 2.2. Faktorizacija nenegativne matrice

Metoda faktorizacije nenegativne matrice računa procjenu nenegativne matrice  $V$  dimenzija  $L \times K$ .

$$V = WH \quad (2.5)$$



**Slika 2.1.** Ilustracija približne faktorizacije nenegativne matrice [3]

**W** je matrica dimenzija  $L \times R$ , a **H** dimenzija  $R \times W$ . Matrice **W** i **H** su inicijalizirane nasumično te se vrijednosti matrice iterativno popravljaju minimiziranjem funkcije gubitka. Matrica **V** je spektrogram zvukovnog zapisa koji se dobije primjenom vremenski kratkotrajne Fourierove transformacije (engl. short-time Fourier transform). Konačno izračunate matrice **W** i **H** se obrnutom kratkotrajnom Fourierovom transformacijom pretvaraju u zvukovni zapis.

## 2.3. Konvolucijske neuronske mreže

Strojno učenje zasniva se na mijenjanju parametara s ciljem optimizacije kriterija uspješnosti. Ulazni podatci iterativno prolaze kroz model strojnog učenja, te se izlaz vrednuje s pomoću funkcije gubitka. Osnovni cilj modela strojnog učenja je generalizacija, sposobnost da točno izvede nove zaključke na neviđenom skupu podataka.

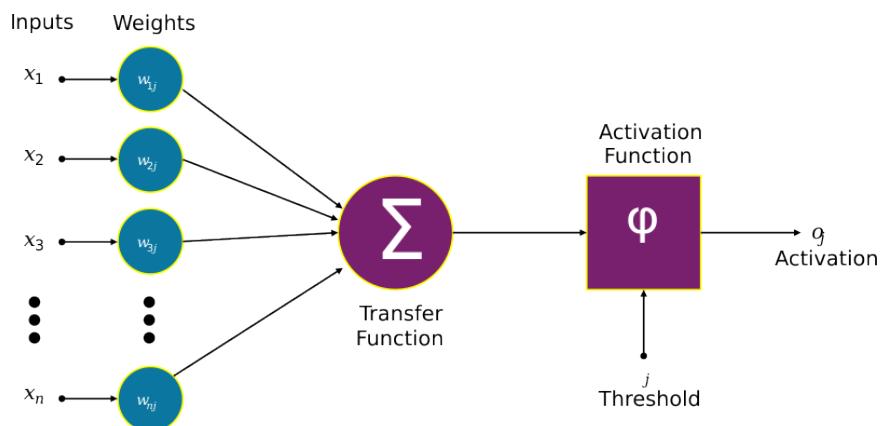
Postoje tri pristupa strojnom učenju:

- Nadzirano učenje — podatci su u obliku (ulaz, izlaz) =  $(\mathbf{x}, y)$ , a cilj stroja za učenje je pronalazak funkcije  $\hat{y} = f(\mathbf{x})$ . Ako je cilj predviđanje brojčane vrijednosti, riječ je o regresiji, ako se predviđa diskretna ili nebrojčana vrijednost onda je klasifikacija.
- Nenadzirano učenje — pronalazak pravilnosti u skupu podataka metodama grupiranja, otkrivanja stršećih vrijednosti, smanjenja dimenzionalnosti.
- Podržano učenje — učenje optimalne strategije na temelju pokušaja s odgođenom nagradom.

Duboko učenje podskup je strojnog učenja koje koristi duboke neuronske mreže, mreže s više slojeva umjetnih neurona između ulaznog i izlaznog sloja. Umjetni neuron je linear funkcija koja prima jedan ili više ulaza, primjenjuje težine ( $w_i$ ) na te ulaze ( $x_i$ ) i zbraja ih kako bi proizvela izlaz.

$$y = \varphi\left(\sum_{i=0}^m w_i x_i\right) \quad (2.6)$$

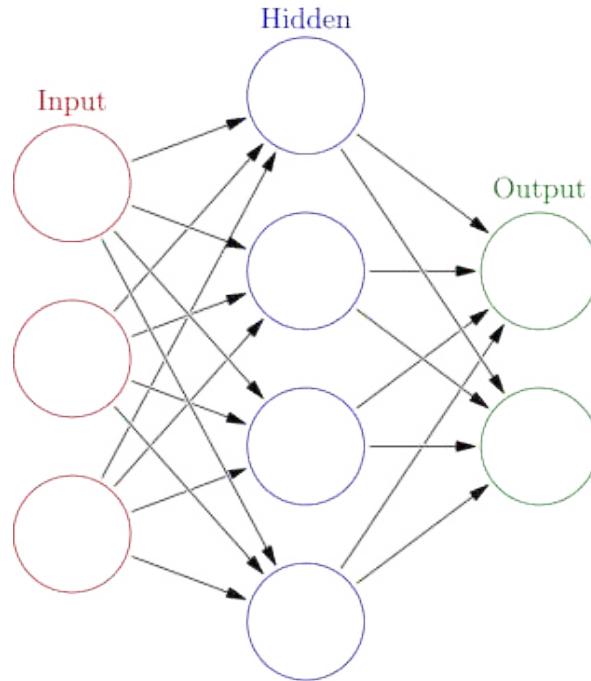
Funkcija  $\varphi$  je funkcija prijenosa. Da bi jednadžba 2.6 vrijedila, u ulazni vektor  $\mathbf{x}$  se mora dodati  $x_0 = 1$ , gdje je  $w_0$  pristranost neurona. Izlaz je analogan aksonu biološkog neurona, a njegova se vrijednost širi do ulaza sljedećeg sloja, kroz sinapsu.



Slika 2.2. Ilustracija građe umjetnog neurona [4]

Slaganjem umjetnih neurona u slojeve i njihovim međusobnim povezivanjem nas-

taju umjetne neuronske mreže. Način na koji su neuroni posloženi i povezani naziva se arhitektura neuronske mreže. Pošto je umjetni neuron linear funkcija, između slojeva mora postojati prijenosna funkcija koja unosi nelinearnost u neuronsku mrežu i omogućuje opisivanje nelinearnih odnosa.



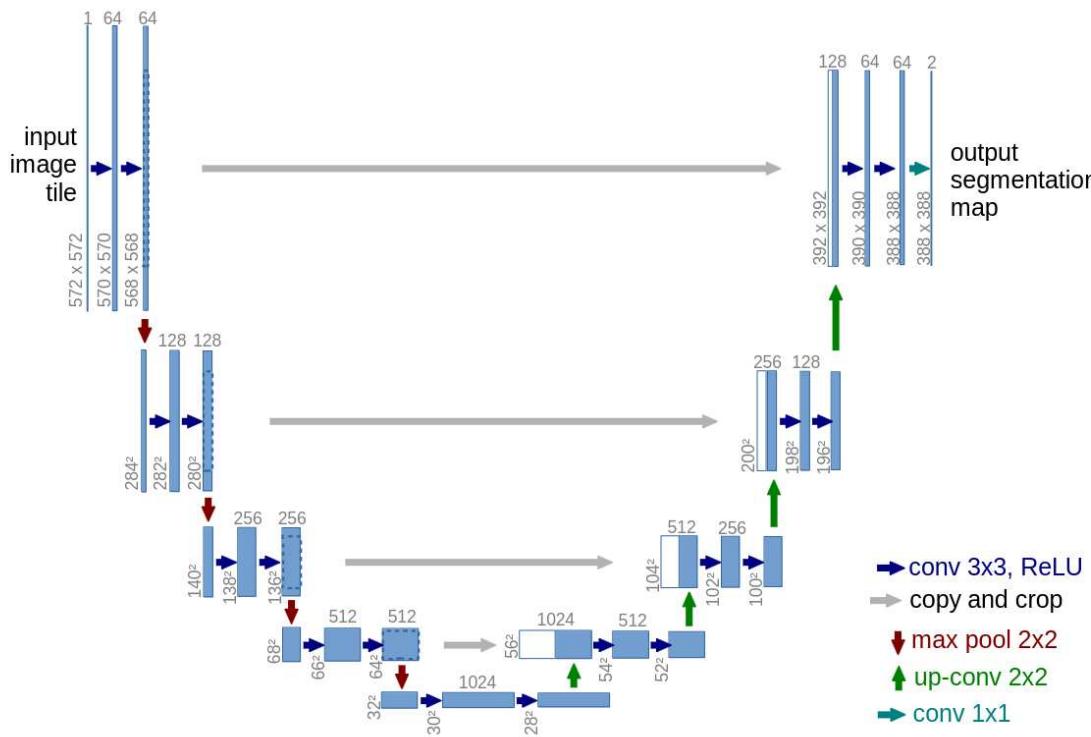
**Slika 2.3.** Ilustracija neuronske mreže i njenih slojeva: ulazni, skriveni i izlazni [5]

Modeli strojnog učenja zasnovani na neuronskim mrežama uče iterativnim ažuriranjem težina s ciljem smanjivanja funkcije gubitka. Težine se ažuriraju postupkom propagacije greške unatrag koji se temelji na izračunu gradijenata (parcijalnih derivacija funkcije pogreške s obzirom na svaku težinu i prag).

Konvolucijske neuronske mreže razlikuju se od ostalih neuronskih mreža svojom sposobnošću samostalnog izdvajanja značajki ulaza primjenom konvolucije. Konvolucija je proces pomicanja filtra po tenzoru ulaza i primjena skalarnog produkta između filtra i područja tensora koje filter prekriva. Tenzor sačinjen od tih skalarnih produkata poznat je kao mapa značajki, aktivacijska mapa ili konvoluirana značajka. Zbog toga se konvolucijske neuronske mreže koriste pri rješavanju zadataka segmentacije i klasifikacije slike. Pretvaranjem zvukovnog zapisa u spektrogram, problem segmentacije zvuka postaje problem segmentacije slike.

### 3. U-Net arhitektura

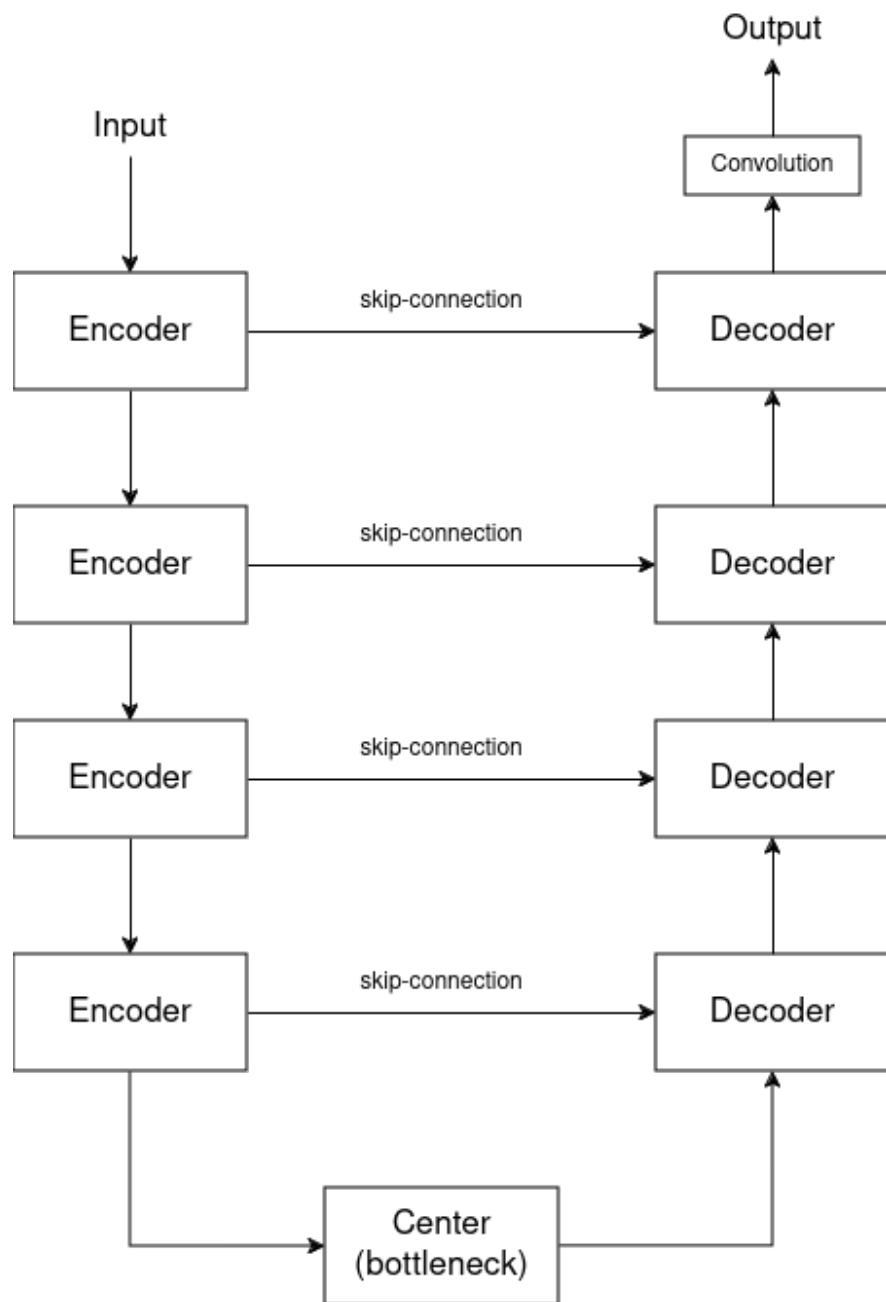
U-Net je konvolucijska neuronska mreža koja je razvijena za segmentaciju biomedicinske slike. [6] Mreža se zasniva na modificiranoj potpuno konvolucijskoj mreži [7] koja je proširena za rad s manje slike za obuku i za postizanje preciznije segmentacije.



Slika 3.1. Prikaz U-Net arhitekture[6]

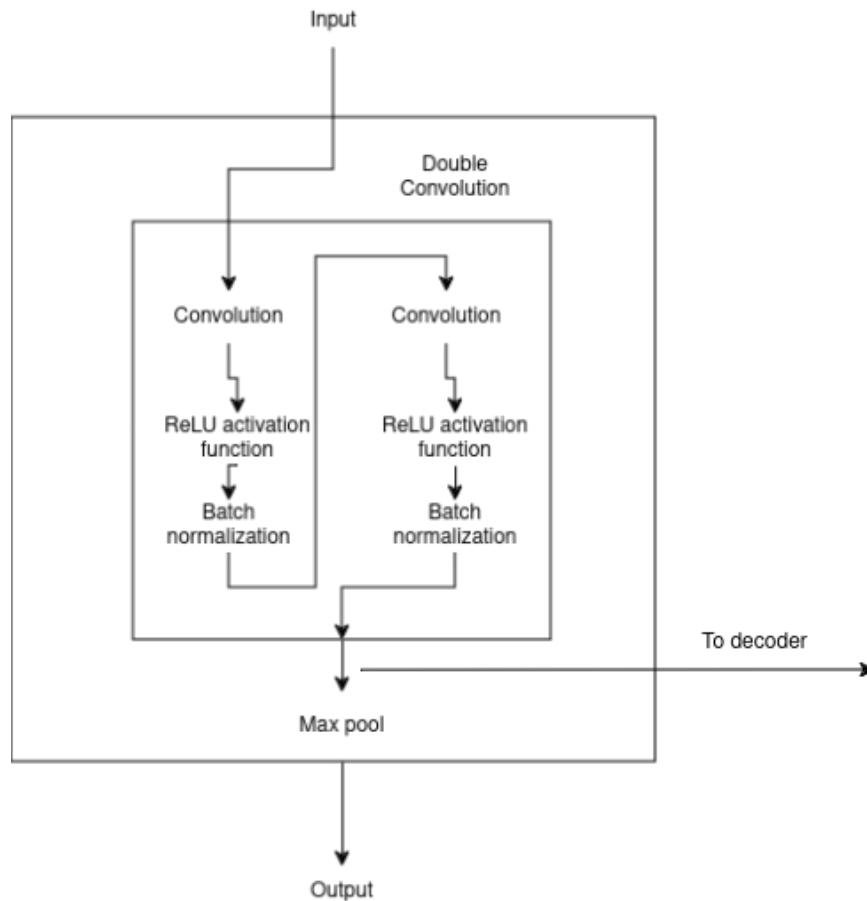
Glavna značajka U-Net arhitekture je njegova sužavajući put i ekspanzijski put, što mu daje oblik slova U. Prolaskom kroz sužavajući put, mreža izdvaja značajke ulaza, informacija o veličini slike se smanjuje, dok se broj značajki (kanala) povećava. Ekspanzijski put radi suprotno, povećava veličinu slike, a smanjuje broj značajki, što rezultira otkrivanjem gdje se značajke nalaze. Konačni izlazni sloj mreže procesom konvolucije prilagođava izlaz na broj značajki koje treba predvidjeti.

### 3.1. Dijelovi U-Net arhitekture



Slika 3.2. Povezanost blokova kodera i blokova dekodera

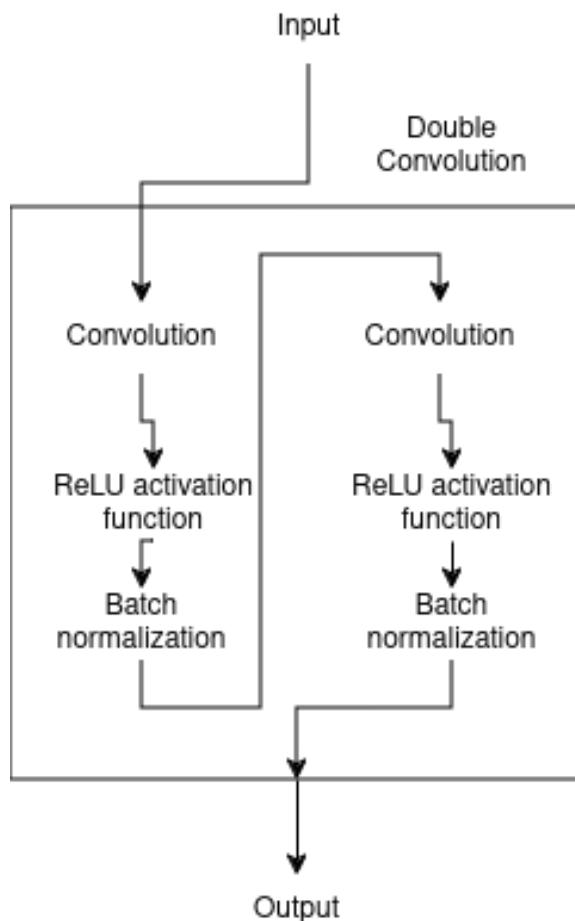
### 3.2. Koder



Slika 3.3. Ilustracija U-Net kodera

Koder se sastoji od dvostrukе konvolucije, čiji se izlaz kasnije koristi kao jedan od ulaza u dekoder, i sloja za maksimalno udruživanje (engl. max pooling layer). Nakon svakog konvolucijskog sloja slijedi prijenosna funkcija zglobnice (engl. Rectified Linear Unit, ReLU) te normalizacijski sloj. Cilj konvolucijskih slojeva je izdvajanje značajki, dok sloj maksimalnog udruživanja smanjuje veličinu ulazne matrice i izabire najbitnije značajke. Prijenosna funkcija zglobnice unosi nelinearnost u neuronsku mrežu te za razliku od sigmoide i tangensa hiperbolnog, nema problem gradijenta koji nestaje. Normalizacijski sloj prilagođava ulazne podatke normalnoj distribuciji što sljedećem sloju omogućuje učinkovitiju analizu podataka i smanjuje mogućnost prenaučenosti.

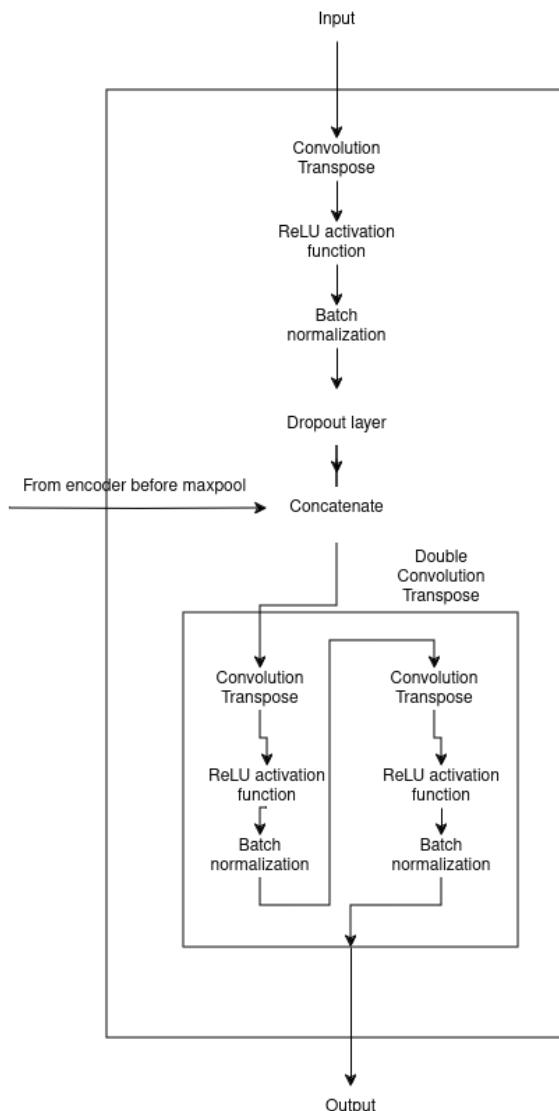
### 3.3. Usko grlo



Slika 3.4. Ilustracija uskog grla U-Net arhitekture

Usko grlo (engl. bottleneck) čini vezu između sužavajućeg puta kodera i ekspanziskog puta dekodera. Njegova je uloga naučiti na koji način se ulaz prolazeći kroz put kodera neuronske mreže sužava te koje su informacije potrebne za vraćanje na početnu veličinu.

### 3.4. Dekoder



**Slika 3.5.** Ilustracija U-Net dekodera

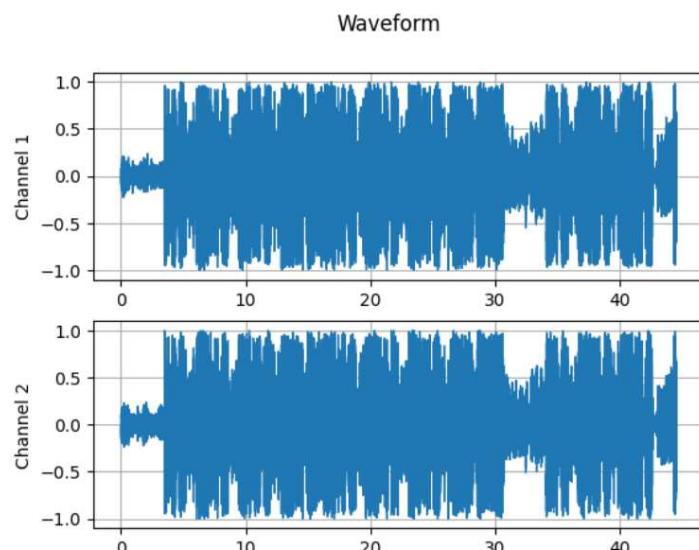
Prolaskom kroz sužavajući put kodera nastaje mapa značajki. Cilj dekodera je mapu značajki ulančati sa zapamćenim tenzorom na paralelnom koderu. Tim postupkom dekoder saznaće gdje se točno značajke nalaze. Rezultat ulančavanja ima onoliko značajki koliko imaju mapa značajki i spremljeni tenzor zajedno. Zbog toga nakon ulančavanja slijedi dvostruka transponirana konvolucija koja prilagođava broj značajki.

## 4. Zvuk kao ulaz u U-Net

Zvuk je oscilacija u tlaku koja se širi u nekom mediju. Mikrofon mjeri promjene tlaka te ih pretvara u električni signal koji se uzorkuje u pravilnim vremenskim intervalima, kvantizira i spremi kao polje (engl. array) u računalu. Stopa uzorkovanja (engl. sample rate) zvuka je bitna mjera koja određuje kvalitetu i raspon frekvencija koje se mogu prikazati. Nyquistova frekvencija je najveća frekvencija koja se može prikazati ovisno o stopi uzorkovanja .

$$f_N = sr/2 \quad (4.1)$$

Smanjenje stope uzorkovanja smanjuje razlučivost zvuka, ali smanjuje vrijeme učitavanja i obrade zvuka.



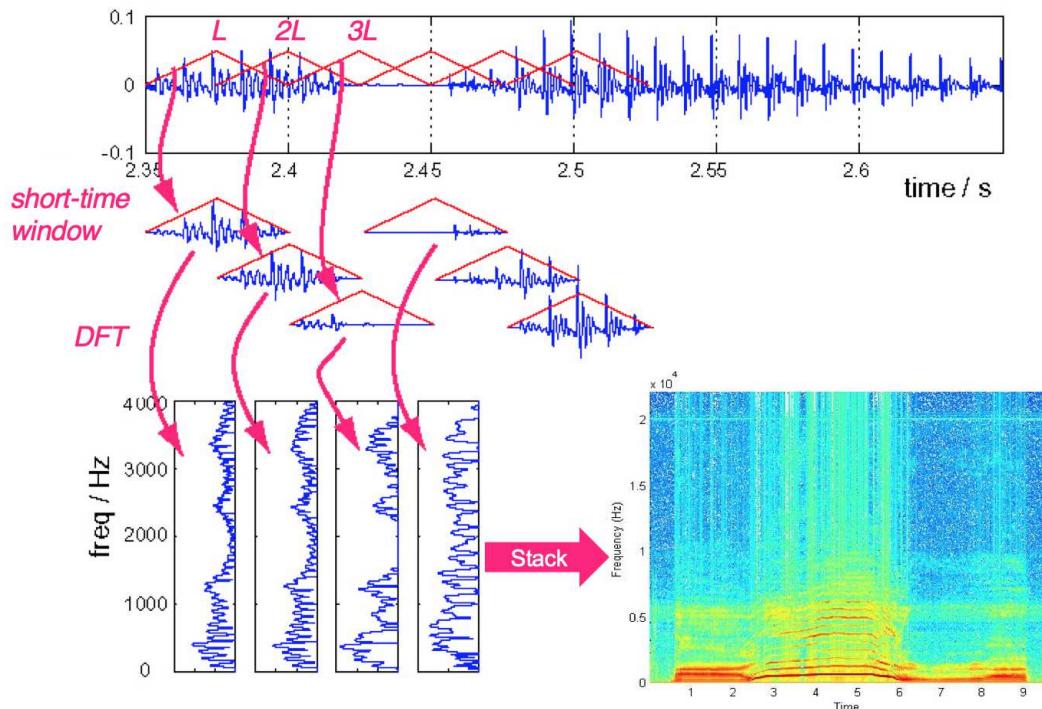
**Slika 4.1.** Prikaz valnog oblika zapisa zvuka

Odvajanje izvora zvuka pomoću U-Net arhitekture je proces u tri koraka:

1. Pretvaranje valnog zapisa (.wav) u spektrogram (vremensko-frekvencijska domena),
2. prolazak spektrograma kroz neuronsku mrežu,
3. pretvaranje svakog spektrograma za svaki predviđeni izvor u valni zapis.

## 4.1. Kratkotrajna Fourierova transformacija

Vremenski kratkotrajna Fourierova transformacija je proces primjena diskretnog Fourierovog transformatora na male isječke signala prolaskom pomičnog prozora. Svaki izračunati frekvencijski bin je u kompleksnom obliku jer zadrži komponentu magnitude i komponentu faze. Obje komponente su potrebne za ponovno pretvaranje u valni oblik.



**Slika 4.2.** Prikaz procesa kratkotrajne Fourierove transformacije i stvaranje spektrograma[8]

Rezultat kratkotrajne Fourierove transformacije je spektrogram kojeg prikazujemo u matričnom obliku. Matrica je oblika *broj frekvencijskih binova* x *broj okvira*

$$\text{broj\_frekvencijskih\_binova} = \lfloor \frac{\text{velicina\_okvira}}{2} \rfloor + 1 \quad (4.2)$$

$$\text{broj\_okvira} = \lceil \frac{\text{broj\_uzoraka}}{\text{velicina\_skoka}} \rceil \quad (4.3)$$

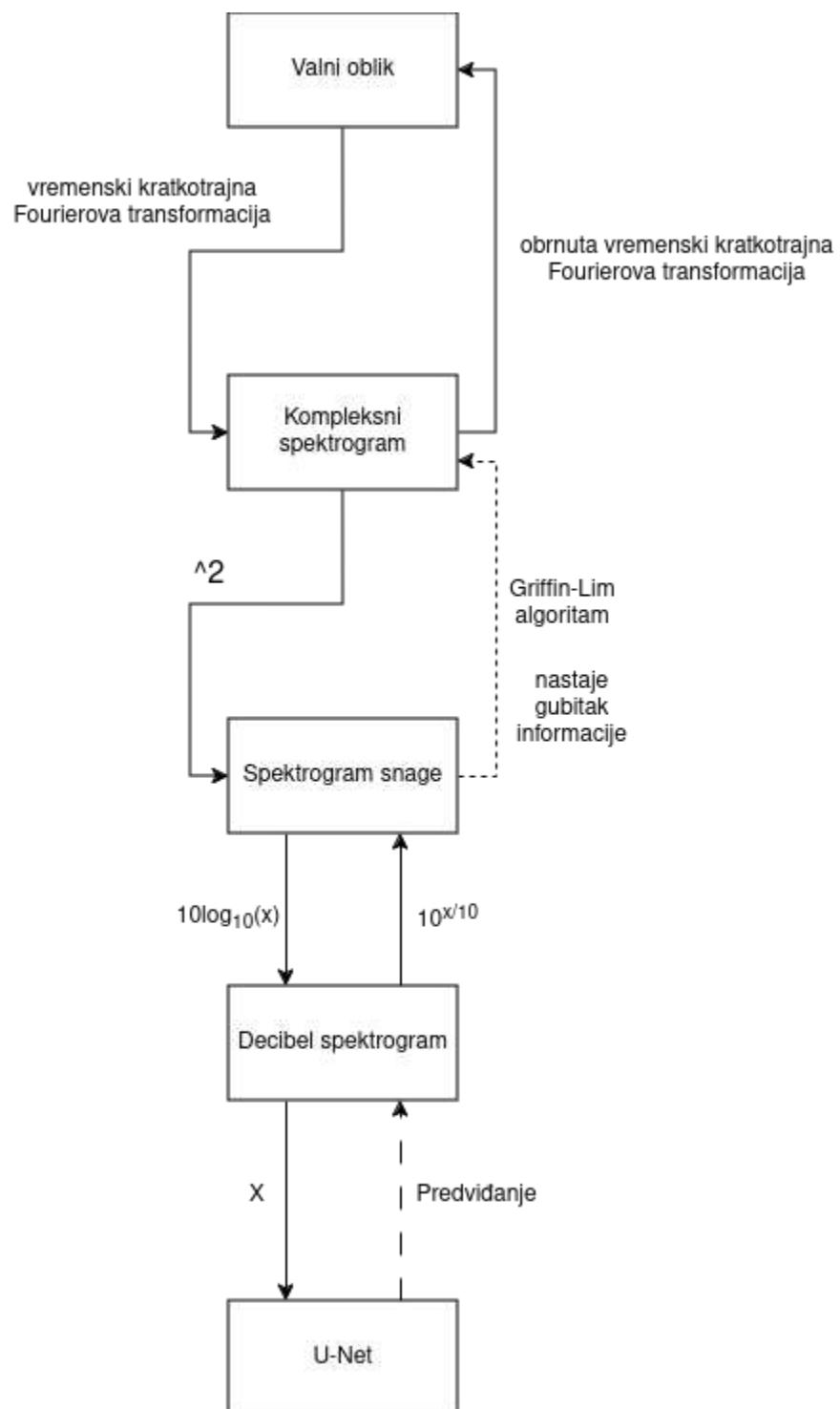
Pretvaranjem spektrograma u spektrogram snage izražen u decibelima (dB), bolje se prikazuje kako ljudi stvarno percipiraju zvuk. [9]

$$S_N(k) = 10\log_{10}(|X_n(k)|^2) = 20\log_{10}(|X_n(k)|), [\text{dB}] \quad (4.4)$$

$X_n(k)$  je isječak dobiven primjenom diskretnog Fourierovog transformatora, a  $S_N(k)$  je isječak spektrograma snage izražen u decibelima.

## 4.2. Griffin-Lim algoritam

Kako bi predviđanje neuronske mreže bilo iskoristivo, predviđeni spektrogram se mora moći pretvoriti u valni oblik. Griffin-Lim algoritam pokušava obnoviti faznu komponentu kompleksnog spektrograma izgubljenu pri pretvorbi u spektrogram snage. Algoritam koristi činjenicu da mjesta u kojima se izračunati frekvencijski binovi preklapaju sadrže informaciju o fazi. Algoritam iterativno namješta procijenjenu fazu pretvorbom spektrograma u valni oblik te nazad u spektrogram sve dok se procijenjena faza ne ustabilili.



**Slika 4.3.** Prikaz obrade zvuka za ulaz u U-Net

## 5. Programsко ostvarenje

Sustav za razdvajanje izvora zvuka je ostvaren u programskom jeziku Python u PyCharm radnom okruženju. PyTorch radni okvir korišten je za dizajn, treniranje i testiranje modela baziranog na U-Net arhitekturi. MUSDB18-HQ skup podataka je pružio potrebne zvukovne zapise za treniranje neuronske mreže. Učenje duboke neuronske mreže poput U-Net zahtijeva mnogo računalnih resursa te je zbog toga bio potreban Google Colab servis.

### 5.1. Programski jezik Python

Python [10] je programski jezik koji omogućuje brži razvoj sustava, laku izradu prototipova i brzo testiranje izrađenog sustava. Široka baza korisnika potiče aktivan razvoj raznih knjižnica i radnih okvira, ali i pip upravitelj paketa omogućuje jednostavnu instalaciju. Python olakšava rad s virtualnim okruženjima u kojima se s pomoću pip-a može lokalno instalirati paketi u projektu umjesto globalno u operacijskom sustavu. Virtuelna okruženja smanjuju mogućnost konflikata s paketima ali i pokazuju koje su verzije paketa potrebne za pokretanje projekta.

Paketi poput NumPy, TensorFlow, PyTorch čine strojno učenje u Pythonu lakšim, a paketi poput Librosa i OpenCV služe za obradu zvuka i slike koje koriste modeli strojnog učenja za treniranje.

### 5.2. PyTorch radni okvir

PyTorch [11] je radni okvir za duboko strojno učenje najčešće korištena za računalni vid i obradu prirodnog jezika. Radni okvir prati objektno orijentiranu paradigmu programiranja te traži od korisnika da svaka komponenta neuronske mreže nasljeđuje razred

`torch.nn.Module` i definira svoje operacije u funkciji `forward()`. Podatci se spremaju u višedimenzionalna polja definirana razredom `torch.Tensor`.

```
import torch
from torch import nn

class NeuralNet(nn.Module):
    # konstruktor
    __init__(self):
        super().__init__()
        # definirati dijelove neuronske mreze

    # funkcija uzima tenzor s podacima
    # vraca obradjeni tenzor
    def forward(self, x: torch.Tensor) -> torch.Tensor:
        # napraviti potrebne operacije na tensoru
        return x
```

**Isječak koda 5..1.** Ilustracija korištenja PyTorch radnog okvira

Radni okvir sadrži knjižnicu `torchaudio` koja pruža funkcije za obradu zvuka ali i omogućuju jednostavno dohvaćanje najpopularnijih skupova podataka.

```
import torchaudio
from torchaudio import transforms as T

spectrogram = T.Spectrogram(n_fft=2048, hop_length=512, power=2.0)

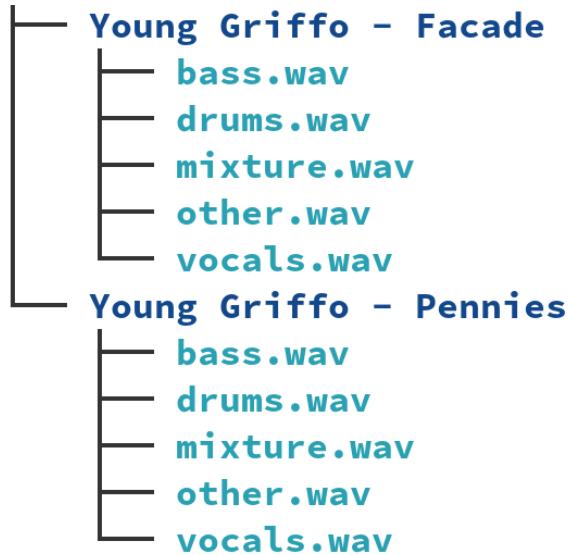
wave, sample_rate = torchaudio.load(path)
power_spec = spectrogram(wave)
```

**Isječak koda 5..2.** Korištenje `torchaudio` knjižnice

Isječak koda prikazuje kako uz `torchaudio` knjižnicu transformirati valni oblik zapisa zvuka u spektrogram. Na jednak način se definiraju inverzne transformacije.

### 5.3. MUSDB18-HQ skup podataka

MUSDB18-HQ [12] je skup 150 glazbenih zapisa koji ukupno traju približno 10 sati. Za svaku pjesmu su izdvojeni vokali (vocals.wav), bubnjevi (drums.wav), bas (bass.wav), ostalo (other.wav) i svi navedeni dijelovi pomiješani zajedno (mixture.wav).



Slika 5.1. Zvukovni zapisi za svaku pjesmu u skupu podataka

Svaki zapis je u obliku wave stereo datoteke sa stopom uzorkovanja 44100 Hz. Skup je podijeljen na skup za testiranje od 50 pjesama, skup za treniranje od 85 pjesama i skup za validaciju od 15 pjesama.

```
import torchaudio.datasets as ds
```

```
data_set = ds.MUSDB_HQ(  
    root=". / data", subset='test', download=True)  
for sample in data_set:  
    # obradi podatke
```

Torchaudio knjižnica podržava rad s MUSDB18-HQ skupom podataka. Isječak koda prikazuje jednostavno skidanje skupa na računalo i iteriranje po skupu za testiranje. Ako je zastavica `subset = 'train'`, onda se može definirati `split='train'` ili `split='validation'` što će omogućiti iteriranje po skupu za učenje ili validaciju. Zastavicom `sources` označavamo od kojih će se izvora sastojati skup podataka. Iteriranjem po skupu podataka dobiva se n-torka koja sadrži tenzor sa svim traženim izvorima, stopu uzorkovanja, broj okvira i

naziv pjesme.

Uz odabran *sources = ["mixture", "vocals"]* vraćeni tenzor je oblika  $2, 2, \text{broj\_okvira}$ . Prva dimenzija označava broj odabralih izvora, a druga broj kanala (stereo signal = 2). Broj okvira se računa kao umnožak trajanja u sekundama i stope uzorkovanja.

$$\text{broj\_okvira} = \text{stopa\_uzorkovanja} * \text{trajanje} \quad (5.1)$$

## 5.4. Google Colab

Google Colab [13] je usluga u oblaku koja omogućuje povezivanje s udaljenim Nvidia GPU-om kako bi se operacije nad tensorima obavile brže. PyTorch ima ugrađenu podršku za CUDA - u (Compute Unified Device Architecture [14]), platformu za paralelno programiranje.

```
device = "cuda" if torch.cuda.is_available() else "cpu"
```

**Isječak koda 5..3.** Ispitivanje koji su uređaju slobodni

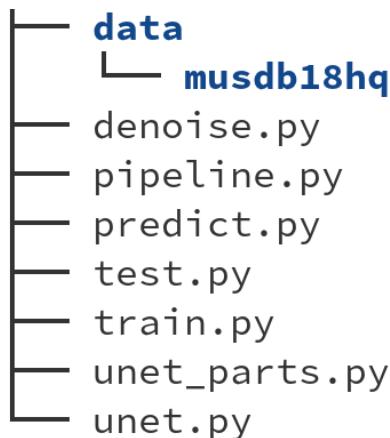
Isječak koda 5.3 se obično nalazi na početku svakog PyTorch programa. Varijabla device sprema koji je uređaj dostupan te je bitno kasnije PyTorch operacijama reći na točno kjom se uređaju trebaju obaviti. Google Colab dopušta spajanje na A100, L4 i T4 grafičku karticu koje podržavaju CUDA tehnologiju .

Colab ima mogućnost integracije osobnog Google Diska u memoriju sjednice Colaba. To je korisno za spremanje i učitavanje naučenog modela.

## 5.5. Ostvareno rješenje

### 5.5.1. Struktura projekta

Projekt se sastoji od nekoliko dijelova: implementacije neuronske mreže, struktura podataka za obradu zvuka i skripti za učenje, testiranje i predviđanje.



**Slika 5.2.** Prikaz strukture projekta

Datoteke *pipeline.py* i *denoise.py* sadrže strukture potrebne za pretvaranje zvukovnog zapisa u spektrogram, spektrogram u valni oblik te strukture potrebne za smanjenje šuma u konačnom predviđenom izvoru zvuka. Neuronska mreža je podijeljena na dvije datoteke: *unet.py* i *unet\_parts.py*. U Python datoteci *unet\_parts.py* su definirani dijelovi U-Net arhitekture koje *unet.py* poziva i povezuje u konačnu strukturu. Skripte *train.py*, *test.py* i *predict.py* služe za učenje, testiranje modela i predviđanje. Skripte se mogu pozvati na lokalnom računalu, ali i zalijepiti u Google Colab Jupyter notebook i tamo pokrenuti.

### 5.5.2. Obrada zvuka

Datoteka *pipeline.py* sadrži sljedeće razrede:

- PadAudio
- AudioPipeline
- PredictAudioPipeline

Sva tri razreda nasljeđuju *torch.nn.Module*.

## © pipeline.PadAudio

- (m) `_init_(self, num_frames: int, chunk_size: int)`
- (m) `forward(self, x: torch.Tensor)`

**Slika 5.3.** UML dijagram razreda PadAudio

Zadatak razreda PadAudio je dodati tišinu na kraj audio zapisa da bi postao dijeljiv na N segmenata veličine *chunk\_size*.

## © pipeline.AudioPipeline

- (m) `_init_(self, sample_rate: int, chunk_size: int, hop_length: int, frame_length: int, batch_size: int = 8, num_max_pool_layers: int = 4)`
- (m) `forward(self, x: torch.Tensor, num_frames: int)`

**Slika 5.4.** UML dijagram razreda AudioPipeline

Razred AudioPipeline zahtijeva da se pri inicijalizaciji definiraju parametri za vremenski kratkotrajnu Fourierovu transformaciju te na koliko velike dijelove treba razdijeliti zvuk. Funkcija *forward()* uzima tenzor oblika *broj\_izvora X broj\_kanala X broj\_okvira* te svaki izvor dijeli na jednak broj segmenata veličine *chunk\_size*. Svaki segment pretvara u spektrogram i takve ih grupira po hrpama (engl. batch). Može se dogoditi da broj segmenata nije djeljiv s veličinom hrpe (*batch\_size*) te u tom slučaju će zadnja hrpa biti manja. Funkcija *forward()* vraća za svaki izvor listu hrpa, gdje je svaka hrpa tenzor oblika *velicina\_hrpe X broj\_kanala X broj\_frekvencijskih\_binova X broj\_okvira*.

## © pipeline.PredictAudioPipeline

- (m) `_init_(self, sample_rate: int, chunk_size: int, hop_length: int, frame_length: int, batch_size: int = 8, num_max_pool_layers: int = 4)`
- (m) `forward(self, x: torch.Tensor, num_frames: int)`
- (m) `backward(self, x: torch.Tensor)`

**Slika 5.5.** UML dijagram razreda PredictAudioPipeline

PredictAudioPipeline funkcijom *forward()* dijeli ulazni zvukovni zapis na listu hrpa spektrograma kako bi neuronska mreža mogla napraviti predviđanje. Funkcija *backward()* za svaki ulazni tenzor spektrograma vraća valni oblik zvuka.

### 5.5.3. Implementacija U-Net arhitekture

```
class Down(nn.Module):  
    def __init__(self, in_ch, out_ch):  
        super().__init__()  
        self.conv = DoubleConv(in_ch, out_ch)  
        self.pool = nn.MaxPool2d(kernel_size=2, stride=2)  
  
    def forward(self, x):  
        x = self.conv(x)  
        return x, self.pool(x)
```

**Isječak koda 5.4.** Implementacija kodera U-Net arhitekture

Razred *Down* je PyTorch implementacija kodera (slika 3.3.) U-Net arhitekture. Metoda *forward(x)* vraća *torch.Tensor* prije i nakon maksimalnog udruživanja (engl. max pooling). Tenzor prije maksimalnog udruživanja se čuva za kasniji ulaz u dekoder.

```
class Up(nn.Module):  
    def __init__(self, in_ch, out_ch):  
        super().__init__()  
        self.up = nn.Sequential(  
            nn.ConvTranspose2d(in_ch, out_ch, kernel_size=2, stride=2),  
            nn.ReLU(inplace=True),  
            nn.BatchNorm2d(out_ch),  
            nn.Dropout2d(0.4)  
        )  
        self.convT = DoubleConvT(in_ch, out_ch)  
  
    def forward(self, x1, x2):  
        x1 = self.up(x1)  
        x = torch.cat([x1, x2], dim=1)  
        return self.convT(x)
```

**Isječak koda 5.5.** Implementacija dekodera U-Net arhitekture

Razred *Up* je implementacija dekodera (slika 3.5.). Metoda *forward*(*x1*, *x2*) vraća tenzor dobiven ulančavanjem sačuvanog tenzora prije izlaska iz paralelnog kodera i izlaznog tenzora prijašnjeg sloja.

```
class UNet(nn.Module):

    def __init__(self, in_channels, n_classes):
        super().__init__()

        self.down_conv_1 = Down(in_channels, 32)
        self.down_conv_2 = Down(32, 64)
        self.down_conv_3 = Down(64, 128)
        self.down_conv_4 = Down(128, 256)

        self.bottleneck = DoubleConv(256, 512)

        self.up_conv_1 = Up(512, 256)
        self.up_conv_2 = Up(256, 128)
        self.up_conv_3 = Up(128, 64)
        self.up_conv_4 = Up(64, 32)

        self.out = nn.Conv2d(32, n_classes, kernel_size=1)

    def forward(self, x):
        down_1, pool_1 = self.down_conv_1(x)
        down_2, pool_2 = self.down_conv_2(pool_1)
        down_3, pool_3 = self.down_conv_3(pool_2)
        down_4, pool_4 = self.down_conv_4(pool_3)

        bottleneck = self.bottleneck(pool_4)

        up_1 = self.up_conv_1(bottleneck, down_4)
        up_2 = self.up_conv_2(up_1, down_3)
        up_3 = self.up_conv_3(up_2, down_2)
        up_4 = self.up_conv_4(up_3, down_1)
```

```
    return self.out(up_4)
```

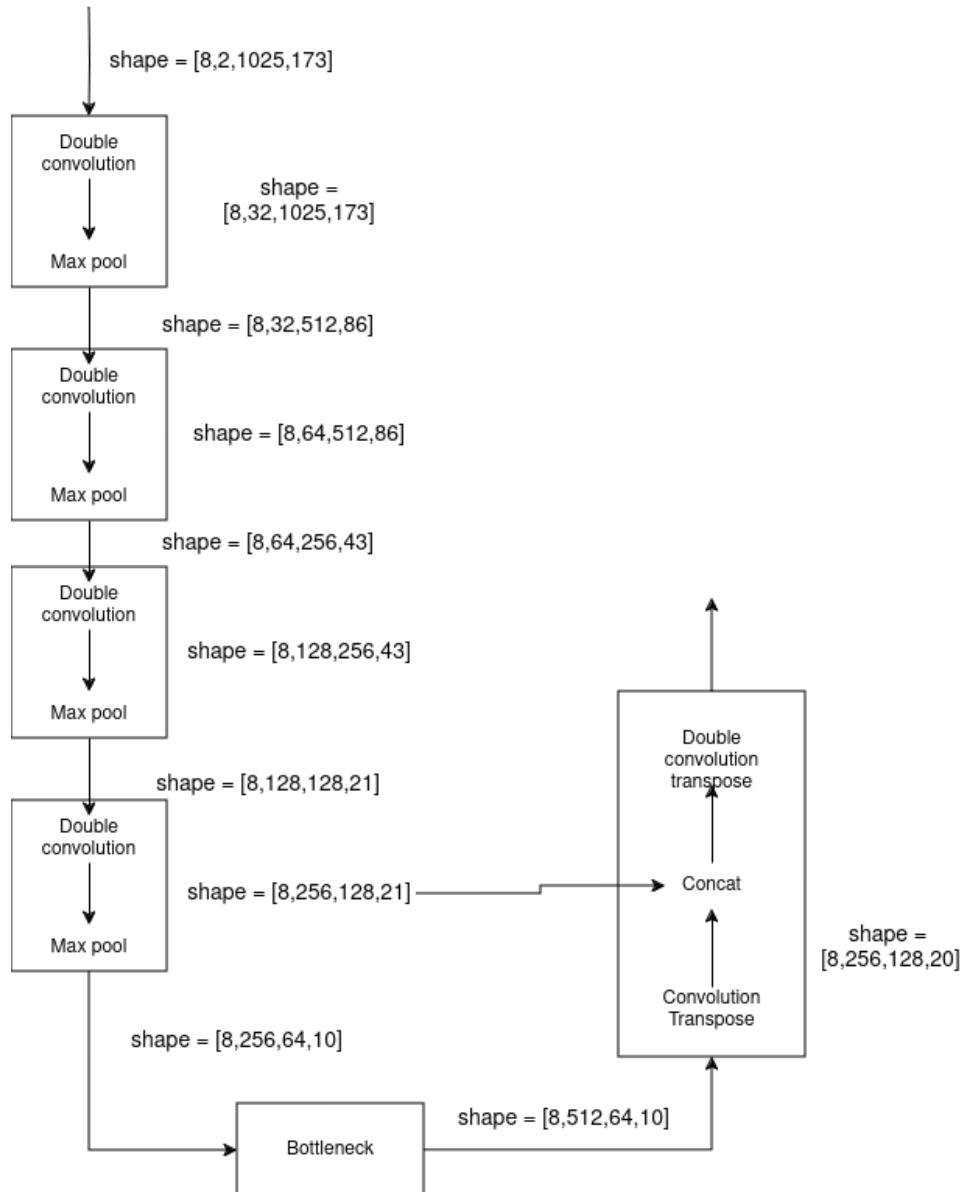
#### Isječak koda 5..6. Implementacija U-Net arhitekture

Isječak koda 5.6 prikazuje konačnu implementaciju U-Net Arhitekture (slike 3.2.3.1.). Metoda *forward(x)* definira čitav prolazak tenzora kroz neuronsku mrežu, od sužavajućeg puta kodera preko uskog grla do ekspanzijskog puta dekodera i konačno izlaznog sloja.

#### 5.5.4. Ulazni podaci

Sustav svaki ulazni zapis zvuka dijeli na segmente trajanja dvije sekunde. Uz stopu uzorkovanja 44.1kHz, zapis će se sastojati od 88200 uzoraka. Veličina okvira Fourierove transformacije je 2048, a veličina skoka 512. Uvrštavanjem u jednadžbe 4.2 i 4.3 dobiva se veličina tenzora spektrograma  $1025 \times 173$ . Da bi tenzor prošao kroz mrežu bez gubljenja informacije i bez potrebnih dopunjavanja nulama, dimenzije trebaju biti djeljive s 2 onoliko puta koliko prolazi kroz sloj maksimalnog udruživanja. U-Net arhitektura sadrži 4 slojeva maksimalnog udruživanja, što znači da dimenzije ulaznog spektrograma moraju biti djeljive s  $2^4 = 16$ . Zbog toga sustav ulazne spektrograme dopunjava nulama do dimenzija  $1040 \times 176$ .

Slika 5.6. prikazuje problem u sloju za ulančavanje na prvom dekoderu kada su ulazne dimenzije  $1025 \times 173$ . Da bi ulančavanje bilo moguće, dva tenzora se moraju imati isti oblik osim u ulančanoj dimenziji. Problem ne bi nastao da su ulazne dimenzije bile djeljive sa 16.



**Slika 5.6.** Prikaz problema s ulaznim dimenzijama

### 5.5.5. Treniranje modela

Učenje neuronske mreže se obavlja u više epoha, a svaka epoha označava jedan prolazak kroz skup podataka. Određuje se veličina skupa uzoraka (engl. batch size) nakon kojih se računa pogreška predviđanja neuronske mreže i poziva optimizator. Optimizator ima određenu stopu učenja (learning rate) koja određuje veličinu koraka u svakoj iteraciji dok se kreće prema minimumu funkcije gubitka.

Ostvareni sustav koristi AdamW optimizator sa stopom učenja 0.001 i propadanjem težina (engl. weight decay)  $10^{-6}$ . L1 je korištena funkcija gubitka.

$$L1 = \sum_{i=1}^N |y_{true} - y_{predicted}| \quad (5.2)$$

```

import torch

model = UNet(2, 4).to(device)
optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)
criterion = torch.nn.L1Loss()
for epoch in range(epochs):
    for sample in data_set:
        for x_batch, y_batch in get_batches(sample):
            y_pred = model(x_batch)
            loss = criterion(y_pred, y_batch)

            loss.backward()
            optimizer.step()

```

**Isječak koda 5..7.** Pojednostavljena petlja za učenje

Skripta *train.py* se pokreće uz sedam argumenta komandne linije:

- *--data\_path*, putanja skupa podataka
- *--model\_save\_path*, putanja do direktorija za spremanje modela
- *--learning\_rate*, stopa učenja
- *--batch\_size*, veličina hrpe
- *--epochs*, koliko epoha traje učenje
- *--checkpoint*, nakon koliko epoha se spremi model
- *--early\_stopping*, koristi li se rano zaustavljanje

Ako je rano zaustavljanje uključeno zastavicom *--early\_stopping*, čim se greška na skupu podataka za provjeru prestane smanjivati, petlja za učenje se prekida.

### 5.5.6. Predviđanje modela

Model predviđa segmentacijske maske za vokale i instrumentaciju. Množenjem segmentacijske maske i ulaznog zvuka nastaje predviđanje vokala i instrumentacije.

Skripta *predict.py* se pokreće uz četiri argumenta komandne linije:

- *--input\_path*, putanja do ulazne wav datoteke
- *--output\_path*, putanja do direktorija za spremanje predviđenih vokala i instrumentacije
- *--model\_path*, putanja do rječnika stanja modela
- *--denoise*, zastavica koja označava uklanjanje li se šum

### 5.5.7. Uklanjanje šuma

Predviđeni izvori zvuka često znaju biti puni šumova i artefakata. Paket noisereduce [15] se koristi za uklanjanje šuma.

```
from noisereduce import reduce_noise as rn  
  
clean_sound = rn(noisy_sound, self.sample_rate, stationary=True)
```

**Isječak koda 5..8.** Primjer korištenja noisereduce

Zastavica *stationary* označava da se procijenjeni prag šuma održava na istoj razini za cijeli signal.

## 6. Rezultati i rasprava

Modeli strojnog učenja za razdvajanje izvora zvuka se vrednuju raznim mjerama:

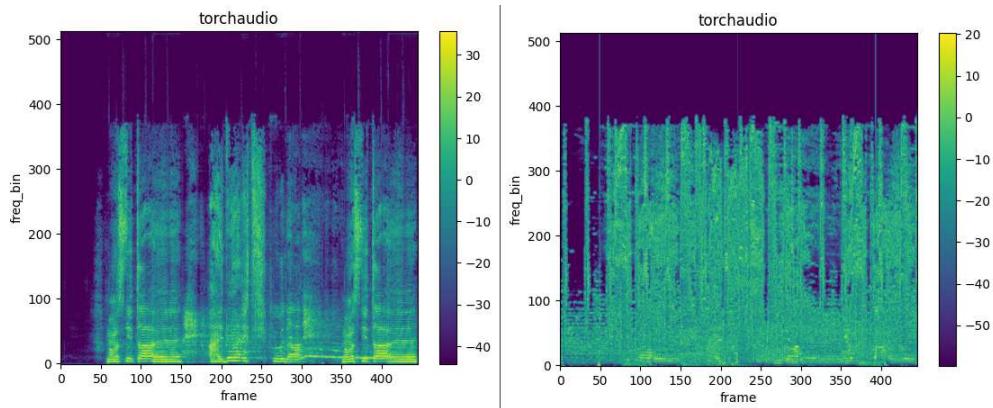
- omjer signala i izobličenja (engl. signal to distortion ratio - SDR)
- omjer signala i artefakata (engl. signal to artifacts ratio - SAR)
- omjer signala i smetnji (engl. signal to interference Ratio - SIR)

Zbog ograničenja torchmetrics knižnjice [16], sustavu je samo izmjerena omjer signala i izobličenja te uspoređen s Deezer Spleeter-om i Open-Unmix.

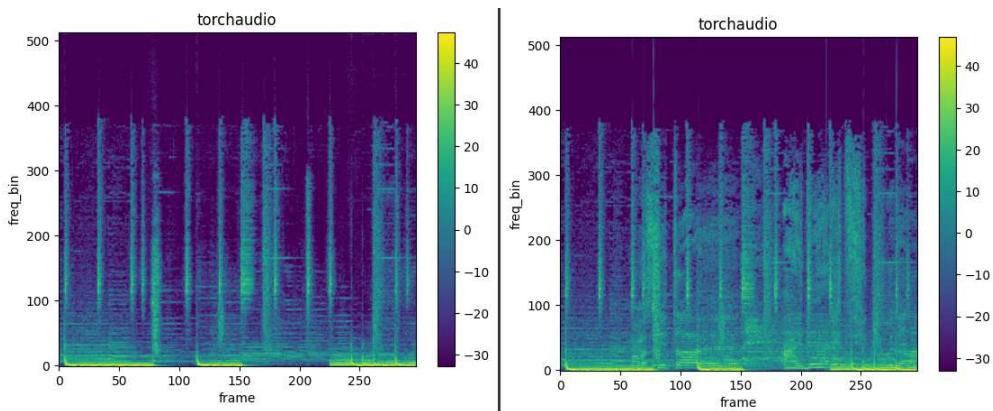
	Spleeter	Open-Unmix	Implemented system (15 epoch)
Vocal SDR	6.55	6.32	-39.22
Avg. L1 loss	-	-	1890.19

Vrednovani model je treniran kroz 35 epoha, sa stopom učenja 0.001 veličinom hrpe uzoraka 8. Pri treniranju, greška na skupu za provjeru pada te nakon 10 epoha počinje opet rasti. Promjena veličine hrpe uzoraka ili smanjenje stope učenja ne mijenja promatrani fenomen. Rezultat tog fenomena je da model koji je učio 15 epoha ima manju grešku na skupu za testiranje od modela koji je učio 30 epoha. Velika greška na skupu za provjeru ukazuje na lošu generalizaciju modela. Negativni SDR označava da ima više izobličenja nego signala.

Predviđena segmentacijska mapa vokala je nešto bolja od segmentacijske mape za instrumentaciju. Predviđeni vokali su tihi i šumoviti, ali ne ostaje mnogo instrumentacije. Instrumentacija dobivena primjenom segmentacijske mape ni malo ne smanjuje prisutnost vokala.



**Slika 6.1.** Razlika u predviđenim vokalima između Spleetera (lijevo) i ostvarenog modela (desno)



**Slika 6.2.** Razlika u predviđenim instrumentalima između Spleetera (lijevo) i ostvarenog modela (desno)

Skripta *test.py* se pokreće uz dva argumenta komandne linije:

- *--data\_path*, putanja do skupa podataka
- *--model\_path*, putanja do rječnika stanja modela

## 7. Zaključak

Tehnologija koja može klasificirati objekt na slici ili razložiti sliku na prepoznate klase postoji već duže vrijeme. Iste te metode mogu se iskoristiti za razdvajanje izvora zvuka. Primjenom vremenski kratkotrajne Fourierove transformacije dobiva spektrogram koji se može smatrati slikovnom reprezentacijom zvuka. Postoje i druge slikovne reprezentacije zvuka, ali pokazalo se da je segmentacijsku mapu najlakše generirati za spektrogram.

Konvolucijske neuronske mreže imaju sposobnost samostalnog prepoznavanja značajki, a U-Net može još i reći gdje su točno nalaze i generirati segmentacijsku mapu. Zbog takvih svojstva U-Net se ne koristi samo za segmentaciju, nego i za generiranje slike i smanjenje šuma.

## Literatura

- [1] R. Hennequin, A. Khelif, F. Voituret, i M. Moussallam, “Spleeter: A fast and state-of-the art music source separation tool with pre-trained models”, Late-Breaking/Demo ISMIR 2019, November 2019., deezer Research.
- [2] N. Mitianoudis i M. Davies, “Audio source separation: Solutions and problems”, *International Journal of Adaptive Control and Signal Processing*, sv. 18, str. 299 – 314, 04 2004. <https://doi.org/10.1002/acs.795>
- [3] Qwertyus, “Nmf”, <https://commons.wikimedia.org/w/index.php?curid=29114677>, 2013.
- [4] Funcs, “Artificial neuron”, <https://commons.wikimedia.org/w/index.php?curid=148910507>, 2024.
- [5] Glosser.ca, “Artificial neuron”, <https://commons.wikimedia.org/w/index.php?curid=24913461>, 2013.
- [6] O. Ronneberger, P. Fischer, i T. Brox, “U-net: Convolutional networks for biomedical image segmentation”, *CoRR*, sv. abs/1505.04597, 2015. [Mrežno]. Adresa: <http://arxiv.org/abs/1505.04597>
- [7] J. Long, E. Shelhamer, i T. Darrell, “Fully convolutional networks for semantic segmentation”, *CoRR*, sv. abs/1411.4038, 2014. [Mrežno]. Adresa: <http://arxiv.org/abs/1411.4038>
- [8] B. Pardo, [https://source-separation.github.io/tutorial/\\_images/stft\\_process.png](https://source-separation.github.io/tutorial/_images/stft_process.png).

- [9] D. Petrinović, "Uvod u digitalnu obradbu govora korištenjem matlaba", [https://www.fer.unizg.hr/\\_download/repository/Uvod\\_u\\_digitalnu\\_obradbu\\_govora\\_koristenjem\\_Matlaba\\_s\\_naslovnicom.pdf](https://www.fer.unizg.hr/_download/repository/Uvod_u_digitalnu_obradbu_govora_koristenjem_Matlaba_s_naslovnicom.pdf), 2010.
- [10] <https://www.python.org/>.
- [11] <https://pytorch.org/>.
- [12] <https://sigsep.github.io/datasets/musdb.html>.
- [13] <https://colab.google/>.
- [14] <https://developer.nvidia.com/cuda-toolkit>.
- [15] timsainb, <https://pypi.org/project/noisereduce/>.
- [16] <https://lightning.ai/docs/torchmetrics/stable/>.

## **Sažetak**

### **Razdvajanje izvora glazbe u zvukovnim zapisima primjenom dubokih neuronskih mreža**

Matej Magat

U završnom radu obrađen je problem razdvajanja izvora zvuka i prikazani su različiti pristupi rješavanja. Naglasak je postavljen na duboko učenje i konvolucijske neuronske mreže. Objasnjene su potrebne transformacije zvučnih zapisa za duboko učenje. Opisan je sustav za razdvajanje izvora zvuka i razvojno okruženje potrebno za njegovo ostvarivanje.

**Ključne riječi:** duboko učenje; konvolucijske neuronske mreže; segmentacija izvora

# **Abstract**

## **Music Source Separation in Sound Records Using Deep Neural Networks**

Matej Magat

In this thesis, the problem of sound source separation is addressed and different approaches to solving it are presented. Emphasis is placed on deep learning and convolutional neural networks. The necessary transformations of sound tracks for deep learning are explained. The system for separation of sound sources and the development environment necessary for its implementation are described.

**Keywords:** deep learning; convolutional neural networks; source segmentation