ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Методика мультиголосового поділу на основі нейронних мереж»

на здобуття освітнього ступеня магістра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Максим ГАПЕЙ
(підпис)

Виконав: здобувач вищої освіти групи ПДМ-61
Максим ГАПЕЙ

Керівник: Максим ФЕСЕНКО
к.т.н., доцент

Рецензент: Наталія ТРІНТІНА
к.т.н., доцент

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Магістр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
«_____» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Гапею Максиму Юрійовичу

1. Тема кваліфікаційної роботи: «Методика мультиголосового поділу на основі нейронних мереж»

керівник кваліфікаційної роботи Максим ФЕСЕНКО, к.т.н., доцент,

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «15» жовтня 2024 р. № 320.

2. Строк подання кваліфікаційної роботи «17» грудня 2024 р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література, методики розділення джерел звуку.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):

1. Дослідження методик розділення джерел звуку на основі нейронних мереж.
2. Аналіз архітектур нейронних мереж для мультиголосового поділу.

3. Розробка та валідація методу мультиголосового поділу на основі нейронних мереж.

5. Перелік ілюстративного матеріалу: презентація

1. Мета, об’єкта та предмет дослідження.
2. Порівняльна характеристика.
3. Модифікована архітектура трансформерів.
4. Використані програмні засоби.
5. Екранні форми.
6. Результати моделювання.
7. Висновки.
8. Публікації та апробація роботи.

6. Дата видачі завдання «16» жовтня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	16.10.2024-19.10.2024	Виконано
2	Вивчення матеріалів для аналізу розділення джерел звуку на основі нейронних мереж	20.10.2024-25.10.2024	Виконано
3	Дослідження методик розділення джерел звуку на основі нейронних мереж	26.10.2024-28.10.2024	Виконано
4	Аналіз особливостей впливу кількості мовців на якість мультиголосового поділу	29.10.2024-01.11.2024	Виконано
5	Дослідження архітектур нейронний мереж	02.11.2024-09.11.2024	Виконано
6	Застосування нейронних мереж в мультиголосовому поділі	10.11.2024-21.11.2024	Виконано
7	Оформлення роботи: вступ, висновки, реферат	22.11.2024-24.11.2024	Виконано
8	Розробка демонстраційних матеріалів	25.11.2024-27.11.2024	Виконано
9	Попередній захист роботи	28.11.2024	Виконано

Здобувач вищої освіти _____ Максим ГАПЕЙ
(підпис)

Керівник
кваліфікаційної роботи _____ Максим ФЕСЕНКО
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 70 стор., 14 табл., 22 рис., 30 джерел.

Мета роботи – підвищення точності мультиголосового поділу для схожих голосів за рахунок використання нейронних мереж.

Об'єкт дослідження – процес мультиголосового поділу.

Предмет дослідження – методика мультиголосового поділу на основі нейронних мереж.

У роботі використано різноманітні методи, такі як віконне перетворення Фур'є, фільтр Вінера, алгоритм Гріфіна-Ліма, згорткові, рекурентні, повнозв'язні нейронні мережі, трансформери.

Проведено аналіз сучасних методик: DNN-based HOA Source Separation, Wave-U-Net, DPRNN-based Speaker Separator та SepTDA.

Розроблено методику мультиголосового поділу на основі нейронних мереж та підвищено точність розділення схожих голосів.

Проведено моделювання власної та сучасних методик на синтезованому наборі даних.

Результати дослідження показують приріст точності поділу мовлення, завдяки впровадженню модифікованої архітектури трансформерів (SepTDA).

КЛЮЧОВІ СЛОВА: МУЛЬТИГОЛОСОВИЙ, ЧАСОЧАСТОТНИЙ, ЧАСОАМПЛІТУДНИЙ, СИГНАЛ, АУДІО, ТРАНСФОРМЕР, STFT, LSTM

ABSTRACT

Text part of the master's qualification work: 70 pages, 22 pictures, 14 table, 30 sources.

The purpose of the work – accuracy increasing of multi-voice separation for a similar voices using neural networks.

Object of research – the process of multi-voice separation.

Subject of research – method of multi-voice separation based on neural networks.

In the study, various methods were used, such as short-time Fourier transform, Wiener filter, Griffin-Lima algorithm, convolutional, recurrent, fully connected neural networks, transformers.

An analysis of modern techniques was conducted: DNN-based HOA Source Separation, Wave-U-Net, DPRNN-based Speaker Separator and SepTDA.

A methodology for multivoice separation based on neural networks was developed, and the accuracy of separating similar voices was improved.

Simulations of both our own and modern methodologies were conducted on a synthesized dataset.

The research results show an increase in the accuracy of speech separation, thanks to the implementation of a modified transformer architecture (SepTDA).

KEYWORDS: MULTIVOICE, TIME-FREQUENCY, TIME-AMPLITUDE, SIGNAL, AUDIO, TRANSFORMER, STFT, LSTM.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

FT — Fourier Transform

STFT — Short-Time Fourier Transform

FFN — Feedforward Neural Network

RNN — Recurrent neural network

CNN — Convolutional neural network

LSTM — Long Short-Term Memory

GRU — Gated Recurrent Unit

HOA — High-Order Ambisonic

SDR — Signal-to-Distortion Ratio

SI-SDR — Scale-Invariant Signal-to-Distortion Ratio

ВСТУП

Мультиголосовий поділ — це процес розділення джерел мовлення звуку на окремі голоси. Ця проблема доволі актуальна в сучасному світі через зростання кількості технологій, з якими безпосередньо взаємодіє людина.

Оскільки голосова взаємодія є основним засобом комунікації, багатоголосове середовище та шум є перешкодами до систем, які базуються на розпізнаванні мови. Здатність точно розділяти голоси та запам’ятовувати інформацію від кількох мовців одночасно є ключем до створення багатофункціональних та гнучких систем, особливо у сферах робототехніки та Інтернет речей. Система, яка не здатна точно відокремити голоси, може сплутати (змішати) інформацію, що особливо критично в сферах, де важлива точність. Зі збільшенням обсягу аудіоінформації збільшується потреба в інструментах мовленнєвого аналізу, і відповідно методиках вирішення проблем мультиголосового поділу.

Сучасні алгоритми та методики дозволяють досягати високої точності навіть у складних умовах, однак в більшості випадків вони вирішують конкретні проблеми (вилучення шуму, поділ музики від вокалу, корекція мовлення). Все це потребує критичної оцінки послідовності та доцільності їх використання, і не завжди спеціалізований інструмент може якісно впоратися із завданням.

Мета роботи – підвищення точності мультиголосового поділу для схожих голосів за рахунок використання нейронних мереж.

Об’єкт дослідження – процес мультиголосового поділу.

Предмет дослідження – методика мультиголосового поділу на основі нейронних мереж.

В процесі дослідження вирішувалися наступні завдання:

1. Літературний огляд. Досліджено та проаналізовано сучасні наукові джерела, отримано уявлення про різноманітні методики та алгоритм, що використовуються для розділення джерел звуку.

2. Експериментальні дослідження. За результатами літературного огляду застосовано існуючі алгоритми та методи розділення джерел звуку, проведено аналіз їхніх особливостей та обмежень.

3. Моделювання та розробка. Розроблено, навчено та протестовано модель нейронної мережі для мультиголосового поділу.

4. Порівняльна характеристика. Використано різноманітні метрики оцінювання для визначення точності та швидкості розробленої методики з існуючими.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Основи звуку як фізичного явища

Розуміння відмінностей між аналоговим та цифровим звуком є першим кроком до розробки та використання сучасних технологій обробки аудіо даних. Аналоговий звук надає більш природне звучання, але в контексті його збереження, схильний до спотворень і шуму, тоді як цифровий звук забезпечує високу якість, позбавленого дефектів (під цим слід розуміти, що цифровий звук не може повністю відобразити особливості аналогового звуку).

Аналоговий звук — це безперервний сигнал, який відображає коливання тиску повітря, створюваного звуковими хвилями. Цей тип звуку можна представити як синусоїду, яка безперервно та плавно змінюється з часом.

Цифровий звук — це дискретний сигнал, отриманий шляхом перетворення аналогового сигналу в цифровий. Цей процес включає квантування вибірки амплітуд аналогового сигналу через рівні інтервали часу. Результатом є послідовність чисел, які мають значення амплітуди звукового сигналу в певні моменти часу (див. рис. 1.1).

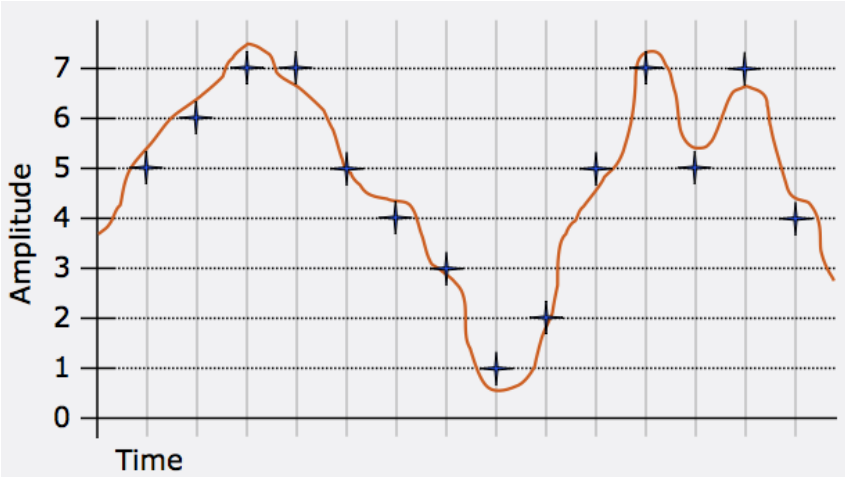


Рис. 1.1 Кожен хрестик відповідає значенню амплітуди дискретного сигналу через рівні інтервали часу

Аналоговий та цифровий звук є хвилями, амплітуда якої визначає інтенсивність або гучність звуку. Амплітуда вимірюється у децибелах (дБ), що є логарифмічною одиницею виміру (використовується десятковий логарифм). Збільшення на 10 дБ сприймається як подвоєння гучності звуку (це слідує з визначення суми логарифмів). Людина може сприймати звуки гучність яких лежить в діапазоні від близько 0 дБ до 120-130 дБ.

Кодування цифрового звуку — це перетворення звукового сигналу у форму, зручну для зберігання, передачі та обробки. Основними етапами цього процесу є дискретизація, квантування, а також різні методи стиснення та перетворення сигналу (стиснення допускає видалення різних частот, які не входять в діапазон сприйняття людини). Процес вимірювання значення амплітуди звуку з однаковим інтервалом у часі - називається дискретизацією. Оскільки пам’ять та швидкість пристроїв записування звуку є обмеженими, вводиться термін частоти дискретизації (Гц) — тобто кількість вимірювань в секунду. На рис. 1.2 зображено різницю між різними частотами дискретизації.

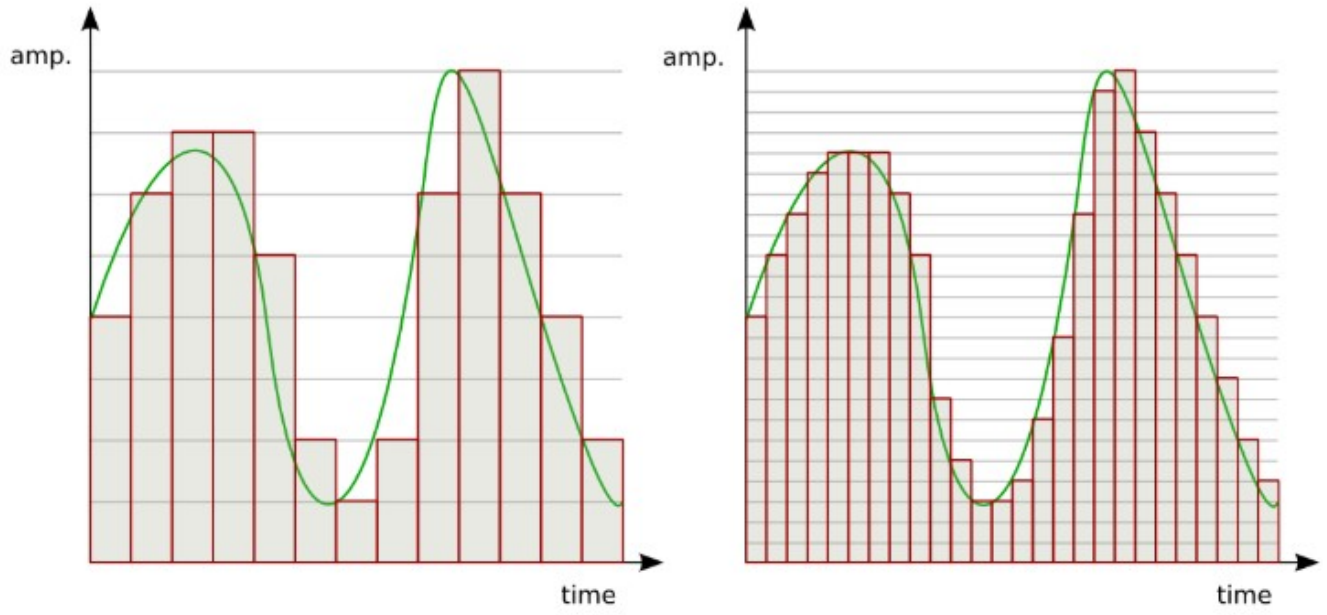


Рис. 1.2 Менша частота дискретизації (зліва) та більша частота дискретизації (справа)

Не варто плутати частоту дискретизації із частотою звуку, оскільки друге визначає кількість коливань звукової хвилі за секунду. Частота звуку визначає його висоту. Високі частоти відповідають високим звукам, а низькі частоти – низьким звукам.

Діапазон частот чутних для людини зазвичай становить від 16 Гц до 20 кГц. Помилково припускати, що висота звуку впливає на її гучність, насправді цей параметр звуку впливає лише на людське сприйняття звуку. Наприклад, спів пташок або гра на скрипці породжують високий звук, тоді як гуркіт грому чи удар барабанів властиво низькому звуку.

Квантування виконується для того, щоб округлити та перевести значення амплітуд в двійковий код, саме бітова глибина (рівень квантування) дозволяє більш точно відтворити аналоговий звук.

Бітова глибина — кількість бітів, що використовуються для представлення амплітуди сигналу в кожен момент вимірювання. Зі збільшенням бітової глибини, збільшується не лише можливість до відтворення звуків високої інтенсивності, але й покращується точність квантування. Гірша точність квантування зумовлена тим, що при меншій глибині необхідно кодувати відносно близькі значення амплітуд одним і тим самим значенням біт (див. рис 1.3).

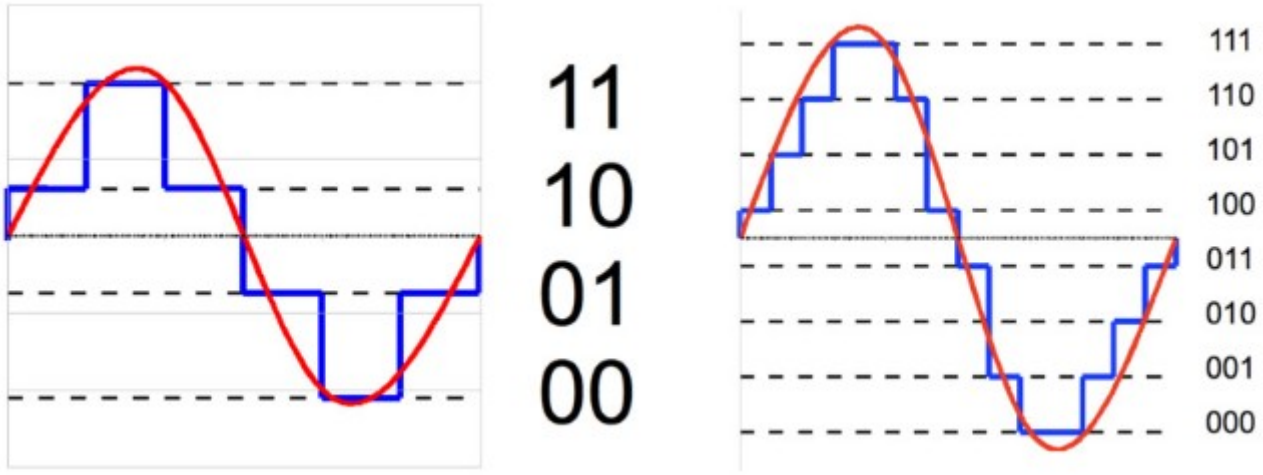


Рис. 1.3 Менша бітова глибина (зліва) та більша бітова глибина (справа)

На практиці застосовують відносно невелику глибину: 16 біт (стандарт для CD дисків), 24 біти (використовується в професійних аудіозаписах), 32 біти (використовується для надзвичайно високоякісного аудіо).

Існує два найпопулярніші методи квантування — лінійне та логарифмічне. Лінійне квантування використовує рівномірні інтервали (застосовуючи математичні функції округлення). Перевагою є простота реалізації, проте вона може бути менш ефективною при обробці сигналів зі змінною в часі амплітудою коливань. Логарифмічне квантування використовує нерівномірні інтервали, що дозволяє краще враховувати особливості людського слуху, який більш чутливий до зміни амплітуди на низьких рівнях гучності. В професійному оцифровуванні звуку частіше використовується логарифмічне квантування та його модифікації.

При оцифруванні звуку слід пам’ятати про теорему Найквіста, суть якої полягає в тому, що для точного відтворення аналогового сигналу в його цифровій формі частота дискретизації повинна бути не менш ніж в двічі більшою більшою за максимальну частоту цього сигналу. Наприклад, для аудіосигналу з максимальною частотою 20 кГц - частота дискретизації повинна бути не менше 40 кГц. Це пов’язано з природою аналогового звуку, як синусоїди, тому для її відтворення необхідно вимірювати двічі за проміжок часу, розрахований для частоти дискретизації.

Антиалайзинг та дитеринг — використовуються для покращення якості цифрового звуку при дискретизації та квантуванні. Антиалайзинг — процес фільтрації сигналу для видалення частот, що перевищують половину частоти дискретизації (за теоремою Найквіста), щоб запобігти аліасингу (спотворення сигналу, що виникає при недостатній частоті дискретизації). Дитеринг — процес додавання невеличкого випадкового шуму до сигналу перед квантуванням, з метою згладжування квантувальних помилок та зменшення спотворень. Дитерінг робить квантувальний шум менш помітним для людського вуха.

Існує кілька способів запису звуку, які можна поділити на аналогові та цифрові методи. Аналоговий запис здійснюється на магнітофонних стрічках та вінілових пластинах. На магнітофонній стрічці звук записується у вигляді

магнітного сліду, який відповідає частоті та амплітуді сигналу. При відтворенні магнітний слід перетворюється назад у звук. На вініловій пластині звук записується у вигляді аналогових рельєфів. При відтворенні голка фонографа рухається відповідно до рельєфу, створюючи вібрації, які перетворюються в звук.

Цифровий запис супроводжується створенням аудіофайлу (WAV, OGG, MP3 та інших форматів), який може зберігатися на різних носіях (HDD та SSD диски, CD та DVD диски, флеш-накопичувачі) та передаватися через інтернет.

Порівняльний аналіз ключових характеристик між аналоговим та цифровим звуком наведено в таблиці 1.1.

Таблиця 1.1

Основні відмінності між аналоговим та цифровим звуком

	Аналоговий	Цифровий
Якість	Залежить від якості запису та відтворення. Безперервний сигнал - природне звучання, може краще передавати нюанси живих виконань. Схильний до шуму й спотворень.	Дискретний сигнал - залежить від частоти дискретизації та бітової глибини, при правильному виборі цих параметрів може досягти дуже високої якості.
Збереження	Зберігається на фізичних носіях, де кожен запис є унікальним і може мати фізичні дефекти.	Зберігається у вигляді цифрових файлів різних форматів.
Обробка	Важче піддається обробці через необхідність маніпуляцій з фізичними носіями. Обмежені можливості редагування та копіювання без втрати якості.	Легко піддається копіюванню, обробці та редагуванню за допомогою програмного забезпечення. Вимагає складного обладнання та програмного забезпечення для обробки.

При розділенні мовці, слід враховувати частотний діапазон людського голосу, який утворюється під час мовлення. Цей діапазон залежить від статі, віку та фізіологічних особливостей людини [12]. Нижче наведено детальний розгляд характеристик частотного діапазону людського голосу. Основний тон голосу визначає висоту звуку і залежність від вібрації голосових зв'язок. У чоловічому голосі домінують низькі частоти через довші та товстіші голосові зв'язки, які створюють повільніші коливання. Жіночий голос має вищу частоту через коротші та тонші голосові зв'язки. Дитячі голоси мають високий тон через коротші голосові зв'язки, які створюють частіші коливання. У підлітків (особливо хлопців) під час зрілості голос знижується через збільшення довжини зв'язок. У людей похилого віку діапазон може звужуватися, а частота зростати через старіння голосових зв'язок. У таблиці 1.2 наведено підсумкове співвідношення між типом голосу та його частотним діапазоном.

Таблиця 1.2

Частотний діапазон різних голосів

Тип голосу	Частотний діапазон (Гц)		
	нижній	середній	верхній
Чоловічий	85-110	110-150	150-225
Жіночий	165-180	180-230	230-255
Дитячий	250-400		

Це основні частоти, які генеруються голосовими зв'язками. Висота звуку (частота) змінюється від напруження зв'язок, їхньої довжини та товщини. Окрім основного тону, людський голос утворює гармоніки, які створюють тембр голосу і відіграють ключову роль у визначенні індивідуальності голосу [12]. Гармоніки можуть досягати частоти у діапазоні до 8-10 кГц. У нижньому діапазоні (до 1 кГц) формується основний тембр. Вищі гармоніки (2–4 кГц) додають ясності голосу, що важливо для розбірливості мови.

Форманти — це частоти резонансу (різкої зміни частоти), які утворюються у вокальному тракті (глотка, ротова та носова порожнини). Вони надають голосу унікальне звучання. Вважається, що для характеристики людського мовлення достатньо чотирьох формант, у таблиці 1.3 наведено структуровану інформацію про їхні особливості та частотний діапазон [12].

Таблиця 1.3

Основні характеристики формант

Форманта (F)	Особливості	Частотний діапазон (Гц)
FI	Залежить від положення щелепи.	250-1000
FII	Пов’язана з положенням язика.	900-2500
FIII	В більшості пов’язана з округленням губ.	2000-3500
FIV	Залежить від тонких артикуляційних деталей.	3000-4000

Також, у будь-якому голосі присутній шум (наприклад, при вимові свистячих та шиплячих приголосних звуків), який знаходиться в діапазоні від 2 кГц до 10 кГц. Варто зазначити, що високочастотний шум підвищує чіткість та впізнаваність звуків, а отже його присутність у мовленні є невід’ємною складовою.

1.2 Віконне перетворення Фур'є

Віконне перетворення Фур'є (STFT) — метод для аналізу звуку, який дозволяє досліджувати частотні характеристики сигналу в часі, оскільки дозволяє представити аудіосигнал часоамплітудного формату в часочастотний. Звичайне перетворення Фур'є (FT) дає частотну характеристику сигналу, але не дає

інформації про те, коли ці частоти з'являються в часі. Відповідно STFT, поєднує в собі FT з віконною функцією для аналізу сигналів у часі та частоті одночасно.

STFT застосовується у багатьох галузях, таких як:

— аналіз звуку (виявлення частотних компонент у музичних творах чи мовленні);

— обробка зображень (аналізу просторових частот в зображеннях);

— біомедична обробка сигналів, (аналізу електроенцефалограм, електрокардіограм тощо).

Результатом перетворення є часочастотне представлення (задане комплексним числом), яке дозволяє отримати спектрограму. Сигнал розбивається на короткі частини (вікна), які не обов’язково повинні мати прямокутну чи квадратну форму. Кожне вікно перетворюється за допомогою класичного FT. Це дозволяє виявити частотний склад сигналу в межах цього вікна на кожному часовому кроці. STFT визначається за формулою нижче (1.1):

$$STFT\{x(t)\}(\tau,\omega) = X(\tau,\omega) = \int_{-\infty}^{\infty} x(t)w(t-\tau)e^{-2i\pi\omega t}dt, \qquad (1.1)$$

де τ — час сигналу $x(t)$, що аналізується;

ω — частота сигналу $x(t)$, що аналізується;

$x(t)$ — сигнал, який аналізується в часі t;

$X(\tau,\omega)$ — результат STFT до сигналу $x(t)$ у часі τ та частоті ω;

$w(t-\tau)$ — віконна функція, яка зміщена на час τ;

$e^{-2i\pi\omega t}$ — стандартне ядро (комплексна експонента) для FT;

t — змінна інтегрування, що представляє час в межах вікна $w(t-\tau)$.

Очевидно, що ST еквівалент STFT з вікном довжиною в нескінченність. На практиці застосовують дискретну форму SFTF через властивості характерні цифровому сигналу (дискретні вимірювання, замість неперервних), яка визначається за наступною формулою (1.2):

$$STFT\{x[n]\}(m,k) = X[m,k] = \sum_{n=0}^{N-1} x[n+mH]w[n]e^{-2i\pi\frac{kn}{N}}, \quad (1.2)$$

де m — часове положення вікна дискретного сигналу $x[n]$, що аналізується;

k — індекс частоти дискретного сигналу $x[n]$, що аналізується (індекс лежить в межах від 0 до $N-1$);

H — крок перекриття між вікнами $w[n]$;

$x[n]$ — дискретний сигнал;

$x[n+mH]$ — сигнал, зміщений на mH;

$X[m,k]$ — результат STFT до дискретного сигналу $x[n]$ з часовим положенням m та частотним індексом k;

$w[n]$ — віконна функція;

$e^{-2i\pi\frac{kn}{N}}$ — дискретне ядро (комплексна експонента) для FT (із частотою k/N);

N — розмір вікна (кількість точок для обчислення FT);

n — це індекс, що пробігає всі точки в межах вікна (від 0 до $N-1$).

Результатом STFT є матриця комплексних чисел (часочастотне представлення сигналу), яка утворює часочастотну область, значення яких зберігають інформацію про амплітуду та фазу сигналу.

Спектральна потужність STFT (в момент часу τ та частоти ω) описує, скільки потужності міститься в кожній частоті сигналу, обчислюється за формулою (1.3):

$$X_{PS}(\tau,\omega) = |X(\tau,\omega)|^2 = X_{\Re}(\tau,\omega)^2 + X_{\Im}(\tau,\omega)^2, \quad (1.3)$$

де $|X(\tau,\omega)|$ — модуль комплексного числа (абсолютне значення);

$X_{\Re}(\tau,\omega)$ — дійсна частина комплексного числа $X(\tau,\omega)$;

$X_{\Im}(\tau,\omega)$ — уявна частина комплексного числа $X(\tau,\omega)$.

Для представлення потужності в дБ використовують формулу (1.4):

$$X_{PSdB}(\tau,\omega)=10\log_{10}(X_{PS}(\tau,\omega)). \quad (1.4)$$

Амплітуда STFT (в момент часу τ та частоти ω) визначається за формулою (1.5):

$$X_{mag}(\tau,\omega) = |X(\tau,\omega)| = \sqrt{X_{PS}(\tau,\omega)}. \quad (1.5)$$

Фазовий кут STFT (в момент часу τ та частоти ω) відображає, як кожна частота в сигналі зміщена відносно інших (відповідно без фази неможливо відновити сигнал), визначається за формулою (1.6):

$$X_{\phi}(\tau,\omega) = \arctan\frac{X_{\Im}(\tau,\omega)}{X_{\Re}(\tau,\omega)}. \quad (1.6)$$

Застосування віконних функцій STFT дозволяє зменшити ефект країв (який може з'являтися, коли сигнал обрізається або розбивається на частини), а також дає можливість впливати на характеристики сигналу, контролюючи баланс між часовою та частотною роздільною здатністю. Роздільна здатність означає наскільки деталізованим буде той чи інший параметр. Наприклад, висока роздільна здатність частоти дає низьку роздільну здатність часу і навпаки — це випливає із принцип невизначеності FT (див. рис. 1.4).

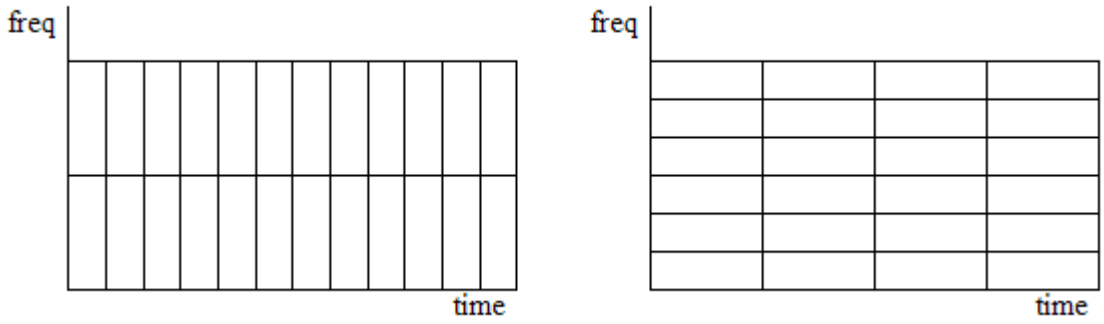


Рис. 1.4 Висока роздільна здатність в часі (зліва), висока роздільна здатність в частоті (справа)

Існують різні типи віконних функцій, використання яких, залежно від того, які характеристики сигналу важливі.

Прямокутне вікно — забезпечує обмеження сигналу в часовій області. До того ж сигнал залишається незмінним, що призводить до різких переходів на межах вікна. Обчислюється за формулою (1.7):

$$w[n] = 1. \tag{1.7}$$

Вікно Хеммінга — згладжує сигнали на краях вікна, роблячи їх поступово зменшеними до нуля. Часто використовується для більш гладких перетворень. Обчислюється за формулою (1.8):

$$w[n] = 0.54 - 0.46\cos\frac{2\pi n}{N-1}. \tag{1.8}$$

Гаусове вікно — містить хороший баланс між часовою і частотною роздільною здатністю, є найкращим вікном для аналізу коротких сигналів. Містить параметр налаштування, обчислюється за формулою (1.9):

$$w[n] = \exp\left(-\frac{1}{2}\left(\frac{n-0.5N}{0.5N\sigma}\right)^2\right), \tag{1.9}$$

де: σ — ширини вікна, впливає на частотну роздільну здатність.

Вікно Блекмана — забезпечує краще згладження на краях сигналу, ніж вікно Хеммінга, однак потребує більших обчислювальних ресурсів. Обчислюється за формулою (1.10):

$$w[n] = 0.42 - 0.5\cos\frac{2\pi n}{N-1} + 0.08\cos\frac{4\pi n}{N-1}. \tag{1.10}$$

Вікно Кайзера — забезпечує плавний перехід сигналу поза межами вікна (особливо корисно при різких переходах), має хорошу часову та частотну роздільну здатність. Містить безліч параметрів налаштування, в тому числі ширину вікна, обчислюється за формулою (1.11):

$$w[n] = \frac{I_0\left(\pi\alpha\sqrt{1-\left(\frac{2n}{N-1}-1\right)^2}\right)}{I_0(\pi\alpha)}, \tag{1.11}$$

де: α — визначає, як форма вікна впливає на розподіл енергії в частотній області (див. рис. 1.5);

$I_0(x)$ — модифікована функція Бесселя нульового порядку.

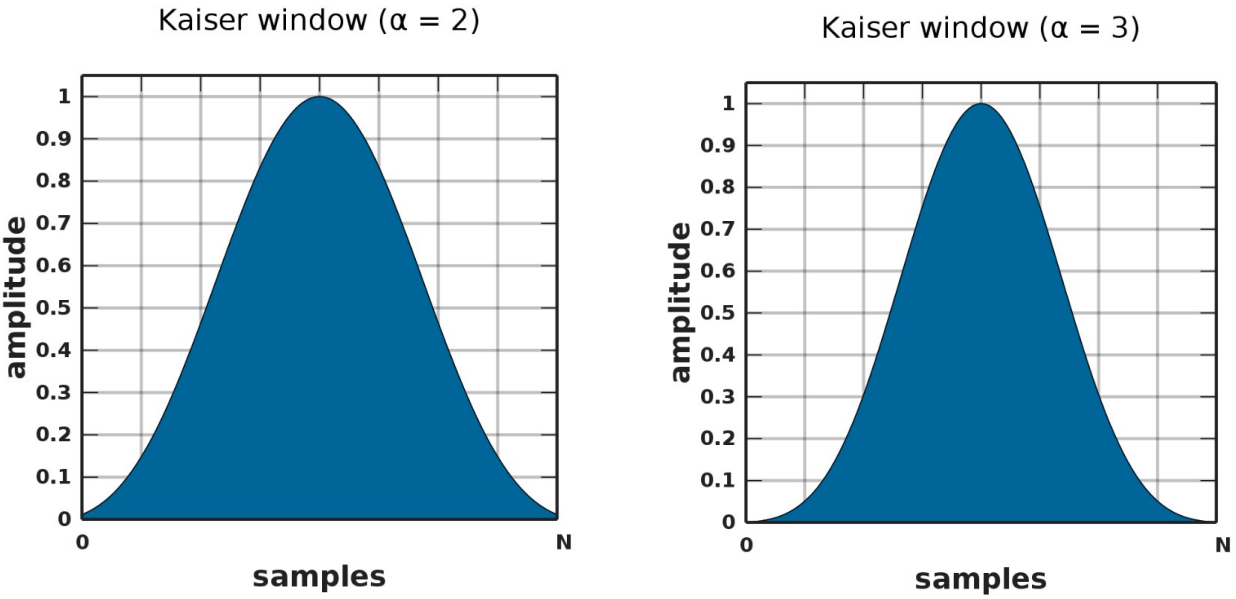


Рис. 1.5 Візуалізація значень вікна Кайзера при різних значеннях α (зліва й справа)

Вибір віконної функції залежить в першу чергу від того, які характеристики сигналу є важливішими. Якщо важлива висока частотна роздільна здатність — можна обрати прямокутне вікна. Якщо потрібен баланс між роздільною здатністю в часі та роздільною здатністю в частоті — можна використовувати вікно

Хеммінга або Блекмана. Найуніверсальнішими є — вікно Кайзера та Гаусове вікно, оскільки вони забезпечують мінімальні спотворення сигналу та утримують баланс між значущими характеристиками сигналу.

В таблиці 1.4 наведено порівняльну характеристику сильних та слабких сторін представлених віконних функцій.

Таблиця 1.4

Переваги та недоліки віконних функції STFT

Назва вікна	Переваги	Недоліки
Прямокутне	Висока роздільна здатність у частотній області.	Погана часова роздільна здатність. Відсутність згладжування породжує спектральні артефакти.
Хеммінга	Баланс між часовою та частотною роздільною здатністю.	Знижує роздільну здатність у частотній області.
Гауссове	Мала кількість спектральних артефактів.	Висока обчислювальна складність. Втрата частотної роздільної здатності для коротких вікон.
Блекмана	Хороший вибір для точних вимірювань.	Висока обчислювальна складність. Менша часова роздільна здатність через ширше вікно.
Кайзера	Дозволяє ефективно зменшити вплив інших сигналів.	Висока обчислювальна складність. Висока складність у правильному налаштуванні параметрів.

STFT має кілька недоліків у контексті розділення аудіо. По-перше, він не здатний одночасно точно відображати час і частоту, оскільки використання фіксованих вікон призводить вибору точності в часі або точності в частоті, що впливає на точність розділення джерел. По-друге, перетворення може погано працювати з сигналами, що містять як низькі, так і високі частоти, оскільки

однакові вікна зазвичай не враховують різні характеристики частот. По-третє, для складних ситуацій з кількома джерелами, метод може бути недостатньо точним, обмежуючи таким чином ефективність розділення.

1.3 Основи нейронних мереж

Нейронні мережі є основним інструментом у задачах розділенні джерел звуку, оскільки вони здатні ефективно виявляти закономірності в часових, частотних і просторових даних. Основні типи нейронних мереж, які використовуються для роботи зі звуком, включають рекурентні нейронні мережі (RNN), згорткові нейронні мережі (CNN) та трансформери.

RNN — розроблені для роботи з послідовними даними. На відміну від звичайних нейронних мереж, вони мають зворотні та латеральні зв'язки, що дозволяє зберігати інформацію про попередні кроки обробки. Завдяки цьому мережі можуть враховувати контекст у послідовності даних. Ця здатність робить їх ефективними для роботи з тимчасовими рядами, текстами, аудіо та іншими послідовностями. Через труднощі з обробкою довгих послідовностей й проблеми зникання (затухання) градієнтів були розроблені вдосконалені версії мереж — довгої короткочасної пам’яті (LSTM) і керованого рекурентного блоку (GRU), які краще працюють з довготривалими залежностями.

LSTM — ця архітектура включає блоки пам’яті, що дозволяють утримувати та передавати інформацію протягом тривалого часу. LSTM широко використовується в задачах поділу голосів у звукових потоках, оскільки дозволяє точно моделювати довготривалі залежності між аудіо компонентами.

LSTM використовує спеціальні блоки пам'яті, які мають три основні ворота. Вони забезпечують адаптивне управління потоком інформації, що дозволяє LSTM утримувати критичні залежності в довгих часових проміжках:

— вхідні ворота визначають, яка інформація з вхідних даних буде збережена в пам'яті;

— ворота забування визначають, яка збережена інформація буде видалена з пам'яті;

— вихідні ворота визначають, яка збережена інформація буде передана на вихід.

На рис. 1.6 схематично зображено архітектуру шару LSTM.

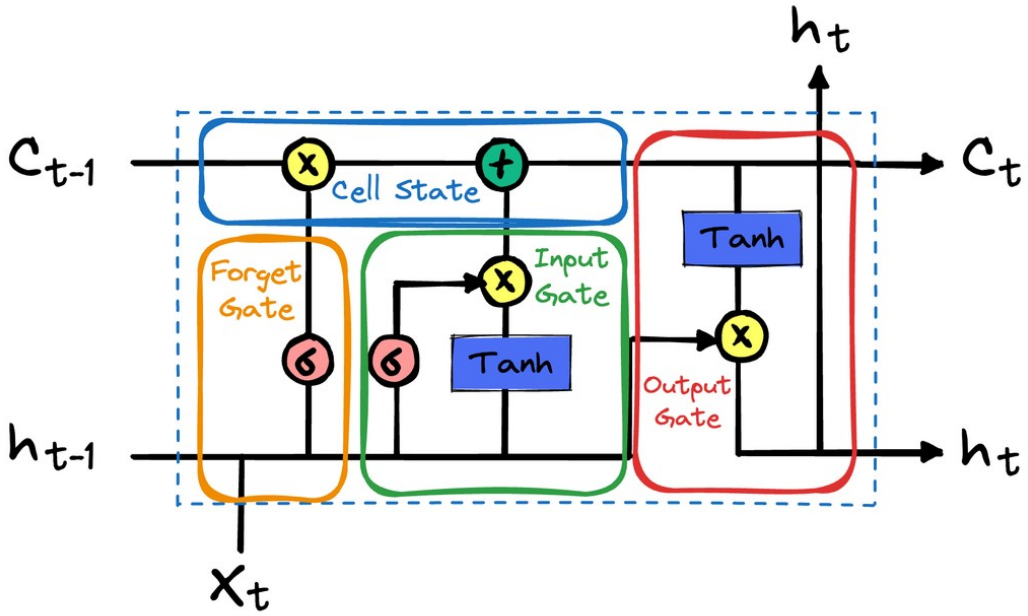


Рис. 1.6 Архітектура шару LSTM

Де: c_{t-1} та c_t — використовуються для збереження інформації про вхідні дані;

h_{t-1} та h_t — попередній і поточний приховані стани, використовуються для передачі інформації на вихід (як кінцевий результат LSTM);

x_t — вхідні дані, що подаються в LSTM (зазвичай це h_t з попереднього шару).

GRU — це спрощена версія LSTM, яка забезпечує швидку та ефективну обробку звуку. Має меншу кількість параметрів, що в свою чергу робить її менш ресурсоємною. На відміну від LSTM, об’єднує вхідні та ворота забування в ворота оновлення. Додатково, використовуються ворота скидання (які дозволяють відфільтровувати збережену інформацію). На рис. 1.7 схематично зображено архітектуру шару GRU.

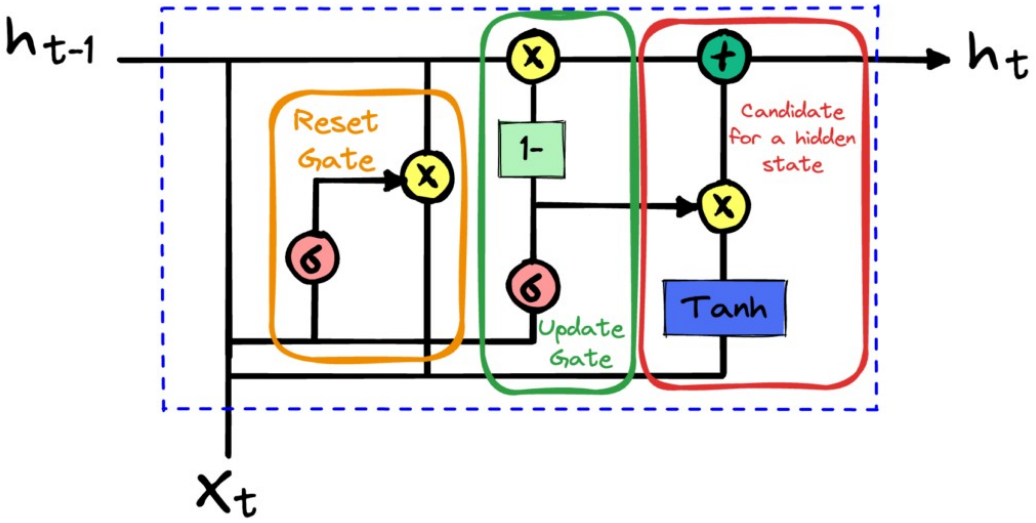


Рис. 1.7 Архітектура шару GRU

Де: 1− (у зеленому квадратику) — позначає операцію віднімання з одиниці значення воріт оновлення.

В таблиці 1.5 наведено підсумковий огляд характеристик LSTM та GRU.

Таблиця 1.5

Порівняльна характеристика LSTM та GRU

Характеристика	LSTM	GRU
Кількість параметрів	Більша	Менша
Обчислювальна складність	Вища	Нижча
Швидкість навчання	Повільніша	Швидша
Гнучкість	Вища	Менша
Застосування	Розпізнавання тексту та мови. Генерація тексту або мелодій. Аналіз послідовностей у задачах з більшим обсягом даних.	Моделі реального часу, де швидкість важливіша за складність. Аналіз послідовностей у задачах з меншим обсягом даних.

Трансформери є сучасним підходом для роботи з послідовними даними. Вони використовують механізм самоуваги, який дозволяє моделювати короткострокові та довгострокові залежності. Основні компоненти архітектури (див. рис. 1.8) трансформера включають: механізм уваги (Self-Attention), мультиголовий механізм уваги (Multi-head Attention), позиційне кодування (Positional Encoding), FFN, замаскована увага (Masked Attention). Також не менш важливими є вхідний ембеддинг та вихідний ембеддинг, завдяки яким вхідні та вихідні дані вбудовуються відповідно до вирішуваної проблеми.

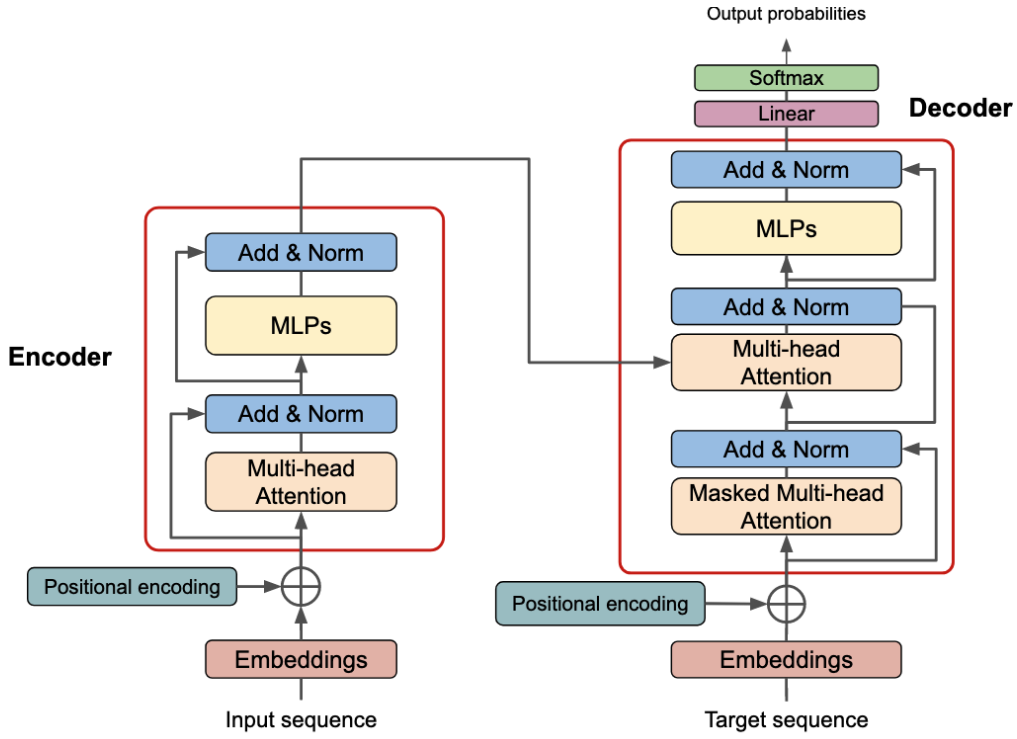


Рис. 1.8 Архітектура трансформерів

На відміну від RNN та CNN, у контексті розділення джерел звуку трансформери мають деякі переваги, представлені нижче:

— підтримка паралельної обробки, що значно прискорює процес навчання;

— гнучкість у роботі з різними типами даних (спектрограма, сирі аудіодані);

— висока ефективність у задачах мультиголосового поділу, навіть при значному перекритті частотних компонентів.

CNN застосовуються для аналізу часочастотного представлення звукових сигналів. Ці мережі дозволяють ефективно обробляти аудіодані, ідентифікуючи елементи сигналу, які відповідають окремим джерелам звуку (наприклад, вокал або музичні інструменти, що мають унікальні спектральні характеристики).

Схожі голоси можуть мати мінімальні відмінності в тоні, тембрі, формантах та гармоніках. Згорткові шари здатні вловлювати ці тонкі відмінності завдяки своїй здатності аналізувати локальні ознаки (патерни) в спектрограмі. Завдяки наявності згорткових фільтрів, модель стає стійкішою до варіативності вхідного сигналу (шум, накладення голосів). Згорткові мережі відомі своєю ефективністю в навчанні на великих наборах даних, оскільки вони використовують спільні ваги (в фільтрах) для локальних ознак, де розмір вхідних спектрограм може бути різним.

Послідовність згорткових шарів дозволяє моделі вивчити різні рівні абстракції. Перші шари можуть виявляти базові частотні компоненти, тоді як наступні шари комбінують ці базові ознаки для розпізнавання більш складних характеристик для кожного голосу. Завдяки операціям згортки з підвибіркою, модель зменшує не тільки обчислювальну складність, але і розмірність вхідних даних, концентруючись на найбільш значущих ознаках.

У сучасних системах часто комбінують різні типи нейронних мереж для досягнення максимальної ефективності, оскільки такий підхід забезпечує високу точність навіть у складних умовах (голоси, що перекриваються). RNN або трансформери можуть бути використані для виявлення часових залежностей, а CNN — для аналізу спектрограм і виділення просторових ознак.

1.4 Огляд актуальних наукових робіт у галузі розділення джерел звуку

Кожен підхід запропонований у багатьох дослідженнях вивчає поділ різних звуків та покращення якості мовлення в одноканальних та багатоканальних записах. Багато досліджень зумовлені на поділ та розпізнавання мовлення в умовах різного рівня зашумленості та кількості мовців. У роботах [1, 16] розглянуто адаптивний підхід для поділу голосів при невідомій кількості спікерів,

при чому методика добре адаптована до великої кількості мовців. В джерелі [16] використовується найсучасніша технологія поділу джерел, яка базується на TDA (Transformer Decoder-based Attractor). Дослідження [6, 11] зосереджені на поділі мовлення з високим ступенем зашумленості. Використання стратегії подвійного виходу ознак дозволяє відокремлювати як цільові, так і сторонні голоси. В дослідженні [9] описаний підхід з використанням фазочутливих мереж для підвищення точності поділу. У дослідженні [13] запропоновано систему поліпшення та розділення мови для цільового мовця з використанням глибоких нейронних мереж. Запропонована методика [20] використовує асинхронні нейронні мережі для поділу мовлення, яка досягає хороших результатів в реальних умовах.

Дослідження поділу співочих голосів націлені на збереженні специфічних ознак в музичному сигналі. Роботи [3, 17] пропонують методи, орієнтовані на поділ з використанням рекурентних нейронних мереж, які також підтримують одноканальний поділ. Методика [23] пропонує комбіновану архітектуру U-Net для паралельного поділу голосу та оцінки частоти. В дослідження [26] розглянуто підхід, що дозволяє користувачам втручатися в процес навчання, що дає можливість покращення результатів поділу.

Частина досліджень сфокусовані на багатоканальному поділі, використовуючи додаткову інформацію для досягнення високої точності. Дослідження [10, 22] більше спрямовані на багатоканальні записи для поділу музики і в деякій мірі мовлення. Багатоканальність запису дозволяє зменшувати чутливість до джерел шуму. У роботі [15] запропоновано SepNet, що включає прогнозування матриці поділу. End-to-end архітектури, такі як Wave-U-Net, дозволяють здійснювати поділ у часовій області, враховуючи при цьому контекст. Модель, що представлена в роботі [4], також забезпечує збереження часових особливостей сигналу, з наданням моделі додаткового контексту. Цей підхід є перспективним завдяки своїй здатності виявляти як короткострокові, так і довгострокові залежності, дещо схоже запропоновано в дослідженні [27].

Деякі роботи зосереджені на зниженні рівня шуму в мовленні з використанням рекурентних нейронних мереж. Описані [21, 29] моделі з покращеною пам'яттю, використовуються для забезпечення стійкості до шуму. Це особливо корисно в системах автоматичного розпізнавання спікерів. Дослідження [30] базується на споріднених підходах, використовуючи коротку довгострокову пам’ять, та маючи підвищену стійкості до шуму. Методики запропоновані в [18, 28] зосередженні на підвищенні чистоти та чіткості мовного сигналу.

Неконтрольовані методи поділу мовлення освітлено в роботі [25], в якій запропоновано підхід з використанням моделі й надлишкових сумішей під час тренування. Це забезпечує модель здатністю ефективно розділяти мовлення навіть за відсутності зразків для порівняння.

Незначна кількість дослідження націлена на поділі голосів у музичних даних (символічна музика). В роботах [14, 19] використовують евристичні підходи (жадібний алгоритм, оцінка входу голосів) для поділу в складних музичних композиціях У дослідженні [26] представлено підхід, що передбачає корегування моделі на основі зворотного зв'язку від користувача. Мультимодальний підхід [24] двоетапного поділу з використанням аудіо-відео даних, надає можливість для поділ аудіоджерел з візуальною підтримкою.

Решата досліджень охоплюють ключові напрямки аудіообробки, такі як: аналіз музики, виділення мелодії, розділення джерел, підвищення стійкості сигналів мовлення до шуму та їх діагностику. Дослідження [2, 5] спрямовані на аналіз музичних даних та покращення сегментації мелодій. Роботи [7, 8, 20] досліджують методи розділення аудіоджерел, що допомагають відокремлювати мовлення від шуму.

2 АНАЛІЗ МЕТОДИК РОЗДІЛЕННЯ ДЖЕРЕЛ ЗВУКУ НА ОСНОВІ НЕЙРОННИХ МЕРЕЖ

2.1 Огляд фундаментальних методів та алгоритмів

Моделювання спектральної інформації — дозволяє аналізувати і моделювати сигнали більш ефективно, і дає можливість працювати з окремими компонентами сигналу, які в доволі часто перекриваються. Аналіз дає можливість працювати з окремими компонентами сигналу, які доволі часто перекриваються.

На рис. 2.1 зображено приклад моделювання спектральної інформації для аудіо сигналу.

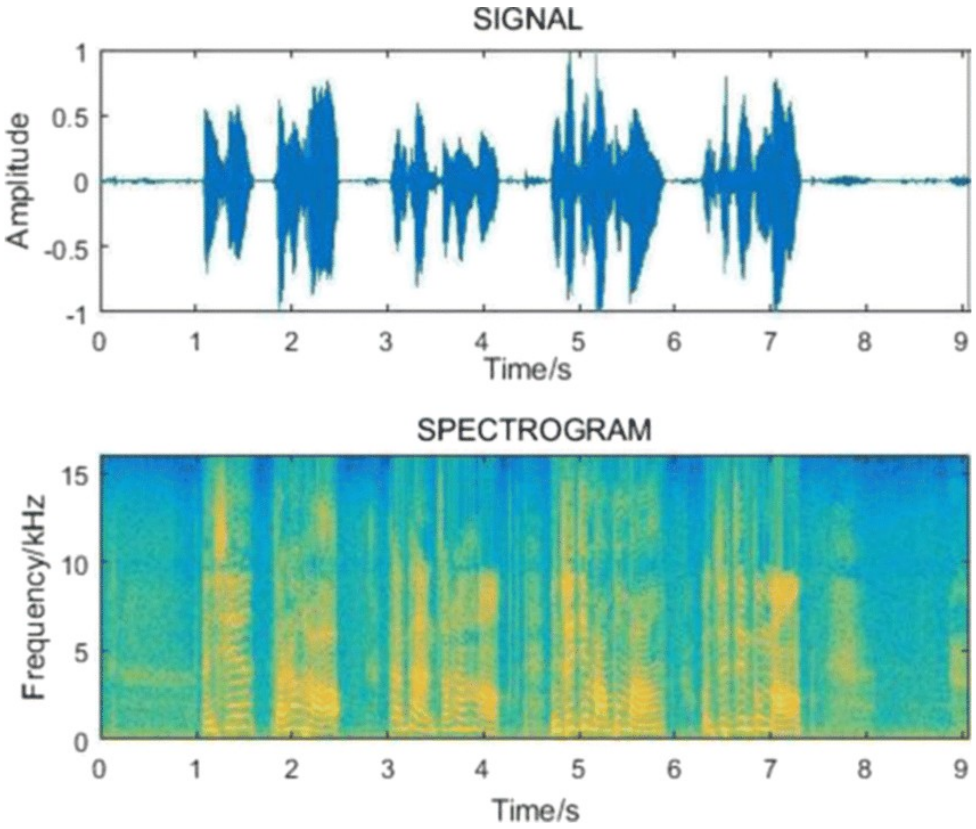


Рис 2.1 Візуалізація сигналу в часоамплітудній області (вгорі) та в часочастотній (знизу)

Але моделювання ігнорує фазову інформацію і робить продуктивність розділення залежною від параметрів для попередньої спектральної обробки. Глибокі нейронні мережі, які працюють зі спектральною інформацією, можуть використовувати спектр амплітуд для визначення основних елементів сигналу, виділяючи частотні та часові особливості.

Багатоканальний фільтр Вінера — метод оптимальної фільтрації для зниження шуму та покращення якості сигналу, використовується для обробки багатоканальних сигналів (в основному стерео звук). Фільтр застосовує статистичні характеристики сигналу й шуму, щоб мінімізувати середньоквадратичну помилку між справжнім та відфільтрованим сигналом. Для великих даних обчислення фільтра може бути ресурсоємним і повільним в реальному часі. Метод потребує точних статистичних характеристик, але якщо ці дані неправильні - ефективність знижується. Фільтр може не відокремлювати джерела звуку належним чином зі складними звуковими сигналами (наприклад, ті які переплітаються). Фільтр не завжди добре працює із просторовими взаємозв'язками, й може бути чутливим до шуму, при низькому співвідношенні сигнал-шум.

На рис. 2.2 продемонстровано застосування фільтру Вінера до зашумленого сигналу.

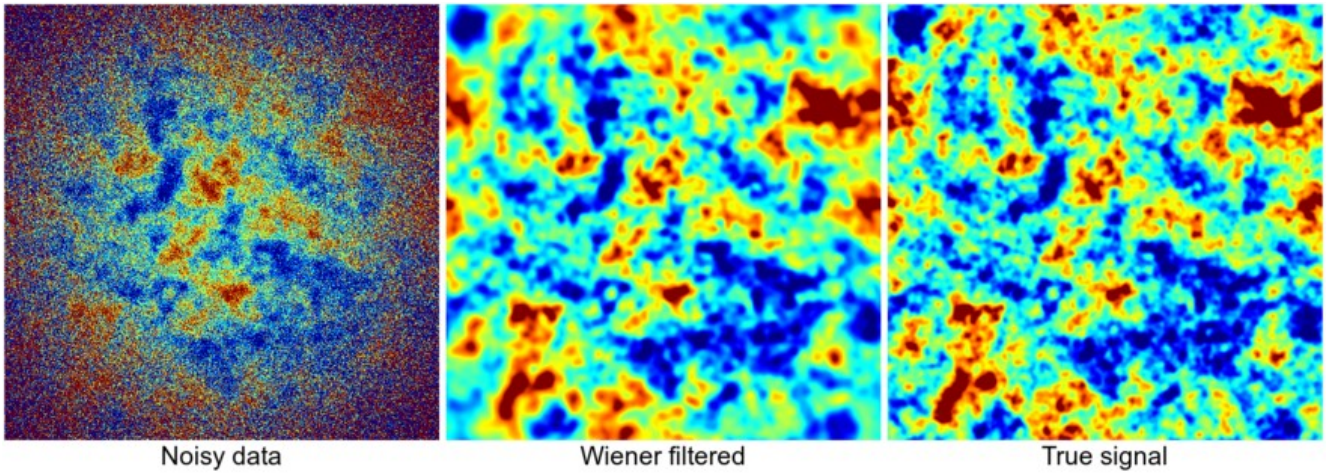


Рис. 2.2 Зашумлений сигнал (зліва), відфільтрований вхідний зашумлений сигналу (посередні) та оригінальний сигнал (справа)

Результат застосування STFT до сигналу із шумом можна описати наступною формулою (2.1):

$$Y(\tau,\omega) = X(\tau,\omega)+N(\tau,\omega), \tag{2.1}$$

де: $Y(\tau,\omega)$ — часочастотне представлення спостережуваного (вхідного) сигналу із шумом;

$X(\tau,\omega)$ — часочастотне представлення сигналу без шуму;

$N(\tau,\omega)$ — часочастотне представлення шуму, однак щоб його знайти (для застосування фільтра Вінера) необхідно здійснити оцінку шуму, оскільки ми не можемо просто взяти його із спостережуваного сигналу.

Фільтр Вінера обчислюється для кожного часочастотного каналу через сподівання, яке можна виразити за формулою (2.2):

$$H(\tau,\omega) = \frac{M[X_{PS}(\tau,\omega)]}{M[X_{PS}(\tau,\omega)]+M[N_{PS}(\tau,\omega)]}, \tag{2.2}$$

де: $H(\tau,\omega)$ — значення фільтру Вінера в часі τ та частоті ω;

$M[X_{PS}(\tau,\omega)]$ — математичне сподівання спектральної потужності сигналу без шуму;

$M[N_{PS}(\tau,\omega)]$ — математичне сподівання спектральної потужності шуму.

Для отримання часочастотного представлення сигналу із послабленим шумом на кожному частотному каналі, необхідно домножити (множення Адамара) фільтр Вінера на часочастотне представлення спостережуваного сигналу (2.3):

$$\hat{X}(\tau,\omega) = H(\tau,\omega)Y(\tau,\omega), \tag{2.3}$$

де: $\hat{X}(\tau,\omega)$ — часочастотне представлення сигналу із послабленим шумом (застосування оберненого STFT до якого дасть сигнал кращої якості).

Метод на основі масок в обробці звукових сигналів часто використовується для розділення звуків (мовлення, музики, інструментів і шуму). Маски визначають, які частини амплітуд та частоти належать до джерела звуку. Звуковий сигнал перетворюється в спектрограму, де кожна частотна компонента аналізується чи пригнічується за допомогою масок, які виділяють важливі елементи для цільового сигналу. Цей метод ефективний у відновленні та покращенні голосу в шумних середовищах. Існують обмеження відповідно до використання цього методу, оскільки маска повинна витягувати та пригнічувати все більше й більше інформації зі зростанням кількості джерел звуку.

Маска — це матриця, яка множиться (множення Адамара) на результат часочастотного представлення STFT, кожен елемент якої може мати значення від 0 (приглушення компонента) до 1 (збереження компонента) або навіть перевищувати 1 (підсилення компонента). Відповідно часочастотне представлення сигналу із застосуванням маски має вигляд (2.4):

$$\hat{X}(\tau,\omega) = M(\tau,\omega)Y(\tau,\omega), \tag{2.4}$$

де: $\hat{X}(\tau,\omega)$ — часочастотне представлення виділеного сигналу;

$M(\tau,\omega)$ — значення маски у часі τ та частоті ω;

$Y(\tau,\omega)$ — часочастотне представлення вхідного (спостережуваного) сигналу.

Існує безліч масок, використання який підпорядковане специфіці вирішуваної проблеми. Розглянемо основні із них:

— ідеальна спектральна маска (ideal binary mask), елементи якої дорівнюють 0 або 1;

— ідеальна амплітудна маска (ideal ratio mask), елементи якої дорівнюють зваженими відношенням амплітуд;

— розумна маска (phase-sensitive mask), елементи якої враховують фазову інформацію сигналу.

Ідеальна спектральна маска визначається за формулою (2.5):

$$M(\tau,\omega) = \begin{cases} 1, & N_{mag}(\tau,\omega) < X_{mag}(\tau,\omega) \\ 0, & N_{mag}(\tau,\omega) \geq X_{mag}(\tau,\omega) \end{cases}, \quad (2.5)$$

де: $N_{mag}(\tau,\omega)$ — амплітуда шуму цільового сигналу (джерела);

$X_{mag}(\tau,\omega)$ — амплітуда цільового сигналу.

Ідеальна амплітудна маска виражається формулою (2.6):

$$M(\tau,\omega) = \frac{X_{mag}(\tau,\omega)}{X_{mag}(\tau,\omega)+N_{mag}(\tau,\omega)}. \quad (2.6)$$

Розумна маска обчислюється за формулою (2.7):

$$M(\tau,\omega) = \frac{X_{mag}(\tau,\omega)}{Y_{mag}(\tau,\omega)}\cos(X_{\phi}(\tau,\omega)-Y_{\phi}(\tau,\omega)), \quad (2.7)$$

де: $Y_{mag}(\tau,\omega)$ — амплітуда спостережуваного сигналу;

$X_{\phi}(\tau,\omega)$ — фазовий кут цільового сигналу;

$Y_{\phi}(\tau,\omega)$ — фазовий кут спостережуваного сигналу;

$\cos(X_{\phi}(\tau,\omega)-Y_{\phi}(\tau,\omega))$ — ваговий множник, який враховує розбіжність фаз.

Алгоритм Гріффіна-Ліма — відновлення фазової інформації для звукового сигналу, дозволяє перетворити амплітуду FT в часовий сигнал. Застосування алгоритму допомагає в синтезі мови, аудіообробці чи розділенні джерел звуку. Він особливо корисний, коли є лише амплітудна частина спектру, а фазова інформація втрачена або недостатня. Широко використовується у сценаріях, де важливо відновити природне звучання сигналу, таких як голосовий помічник та система розпізнавання мови.

Алгоритм базується на ітераційному підході: процес починає з випадкового фазового розподілу, а потім покроково оптимізує фазу, мінімізуючи відхилення між отриманою та оригінальною амплітудною спектрограмою. Після кількох ітерацій алгоритм досягає наближеної фази, даючи змогу перетворити

спектрограму в часовий сигнал з наближеним (але якісним звучанням), однак цей процес повільний і часто такий сигнал не існує.

Початковий крок ітерації алгоритму можна записати формулою (2.8):

$$Y^{(k)}(\tau,\omega) = X_{mag}(\tau,\omega)\exp(i X_{\phi}^{(k)}(\tau,\omega)), \tag{2.8}$$

де: $X_{\phi}^{(k)}(\tau,\omega)$ — фазовий кут з попереднього кроку (для $k = 0$ фазовий кут ініціюється випадковим значенням);

$Y^{(k)}(\tau,\omega)$ — оцінений сигнал $X^{(k)}(\tau,\omega)$.

Проміжний крок ітерації алгоритму можна записати формулою (2.9):

$$\hat{X}^{(k)}(\tau,\omega) = STFT\{\ STFT^{-1}\{Y^{(k)}(\tau,\omega)\}(t)\ \}(\tau,\omega), \tag{2.9}$$

де: $\hat{X}^{(k)}(\tau,\omega)$ — скоригований сигнал $X^{(k)}(\tau,\omega)$ в часочастотному представленні (варто зазначити, що застосування оберненого і прямого STFT дасть вхідний результат лише тоді, коли амплітуда та фазовий кут сигналу правильні).

Фінальний крок ітерації алгоритму можна записати формулою (2.10):

$$X_{\phi}^{(k+1)}(\tau,\omega) = \hat{X}_{\phi}^{(k)}(\tau,\omega), \tag{2.10}$$

де: $X_{\phi}^{(k+1)}(\tau,\omega)$ — скоригований фазовий кут для наступної ітерації.

Алгоритм виконується до тих пір, поки не виконається критерії мінімуму (2.11):

$$\sum_{\tau}\sum_{\omega}\left(\hat{X}_{mag}^{(k)}(\tau,\omega) - X_{mag}(\tau,\omega)\right)^2 \le \alpha, \tag{2.11}$$

де: $\hat{X}_{mag}^{(k)}(\tau,\omega)$ — амплітуда скоригованого сигналу k ітерації;

α — критерій мінімуму.

На рис. 2.3 зображено приклад відновлення фазової інформації деякого сигналу 0-ї, 5-ї, 10-ї та 15-ї ітерацій алгоритму, де відновлюваний сигнал поступово наближається до оригінального.

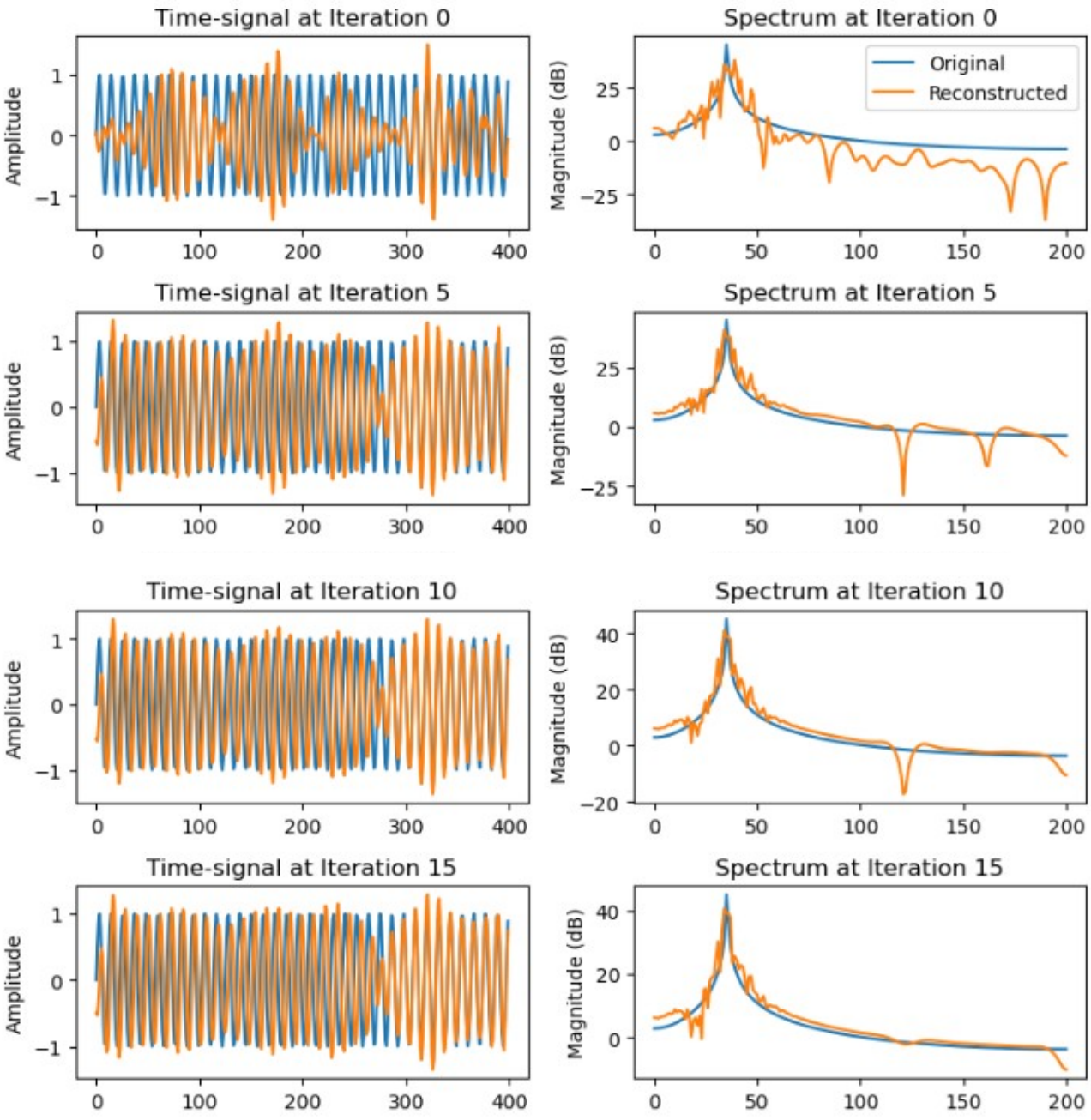


Рис. 2.3 Ітерацій відновлення фазової інформації сигналу, часоамплітудна область (зліва) та часофазова область (справа)

Інваріантна до перестановок функція втрат — забезпечує однакову оцінку помилок незалежно від порядку об’єктів на вході. Такий підхід особливо корисний при розділенні аудіосигналів або при аналізі наборів даних, сформованих у вигляді множин, де потрібно ідентифікувати і порівнювати множини ознак без строгого врахування послідовності. Інваріантність до перестановок може призвести до порушення контексту та втрати важливої часової інформації звуку, ігноруючи взаємозв'язки між елементам (ритм, мелодія, тембр). Це особливо проблематично при обробці складних композицій, де виникнення помилка може значно ускладнити процес навчання.

Інваріантну до перестановок функції втрат, в контексті розділення джерел, можна записати наступною формулою (2.12):

$$LOSS(X,Y)=\min_{\pi\in Perm(N)}\frac{1}{N}\sum_{i=1}^{N}loss(x_{\pi(i)},y_i), \tag{2.12}$$

де: X — вектор передбачуваних джерел звуку;

Y — вектор реальних джерел звуку;

N — кількість джерел в векторі X (вважається, що кількість джерел в векторі Y однакова);

$\pi\in Perm(N)$ — всі можливі перестановки π для N елементів;

$\pi(i)$ — i-й елемент перестановки π;

$x_{\pi(i)}$ — i-е джерело передбачуваного вектора X перестановки π;

y_i — i-е джерело реального (цільового) вектора Y;

$loss(x_{\pi(i)},y_i)$ — деяка функція втрат між $x_{\pi(i)}$ та y_i джерелами;

$\min_{\pi\in Perm(N)}$ — визначає перестановку з мінімальним (оптимальним) значенням.

Очевидно, що зі збільшенням кількості джерел факторіально зростає обчислювальна складність пошуку мінімальної помилки.

Представлені вище алгоритми і методи мають свої сильні та слабкі сторони, і лише в єдності їх правильне використання дасть змогу отримати результати

високого рівня. Слід за значити, що об’єктивна оцінка вирішуваної проблеми є ключовий критерієм вибору тої чи іншої методики. В таблиці 2.1 представлено стислий опис переваг та недолік кожного із описаних методів та алгоритмів.

Таблиця 2.1

Переваги та недоліки фундаментальних методів та алгоритмів

Назва	Переваги	Недоліки
Моделювання спектральної інформації	Ефективний аналіз і моделювання звукового сигналу.	Моделювання ігнорує фазову інформацію; Продуктивність розділення залежить від параметрів для попередньої спектральної обробки.
Багатоканальний фільтр Вінера	Оптимальна фільтрація для зниження шуму та покращення якості звукового сигналу.	Ресурсоємне обчислення фільтра у реальному часі; Ефективність прямопропорційна точності статистичних характеристик; Чутливість до шуму, при низькому співвідношенні сигнал-шум; Не завжди добре працює із просторовими взаємозв'язками.
Метод на основі масок	Визначення частин спектра амплітуд або частотної області.	Маска повинна витягувати та пригнічувати все більше й більше інформації зі зростанням кількості джерел звуку.
Алгоритм Гріффіна-Ліма	Відновлення фазової інформації для звукового сигналу.	Покроково оптимізує фазу, але процес доволі повільний і часто цільовий часовий сигнал не існує.

Продовження таблиці 2.1

Переваги та недоліки фундаментальних методів та алгоритмів

Назва	Переваги	Недоліки
Інваріантна до перестановок функція втрат	Забезпечує однакову оцінку помилок незалежно від порядку об’єктів на вході.	Може призвести до порушення контексту та втрати важливої часової інформації звуку; Ігнорує взаємозв'язки між елементам звуку.

2.2 Аналіз найкращих сучасних методик мультиголосового поділу

2.2.1 DNN-based HOA Source Separation

Стаття пропонує ефективний підхід до розділення мовлення на основі RNN та HOA (High-Order Ambisonics), демонструючи високий потенціал просторової інформації у задачах аудіообробки [22]. Використання HOA та RNN дозволяє досягти суттєвого покращення якості розділення у багатоканальних аудіосценах, роблячи цей підхід перспективним для аудіотехнологій наступного покоління.

Вхідні дані складають аудіозаписи у форматі HOA до третього порядку (Third-Order Ambisonics), що містять до 16 каналів, та просторові ознаки, такі як напрямок приходу сигналу (Direction of Arrival).

Архітектура нейронної мережі містить у собі RNN та часочастотні маски. RNN використовуються для моделювання часових і просторових залежностей у багатоканальних даних. Просторові ознаки інтегруються у RNN як додаткова інформація для покращення розділення. HOA-декодування дозволяє відновити інформацію про напрямок джерел. Використання HOA-сигналів покращує розділення мовлення у порівнянні з нижчими порядками амбісоніки або традиційними багатоканальними методами. У статті застосовано LSTM шари, які забезпечують стабільну роботу з довгими часовими залежностями. Модель

прогнозує маски (для окремих джерел у часочастотному представленні), які використовуються для реконструкції сигналів.

Мережа тренується на синтетичних і реальних аудіозаписах, де змішано кілька джерел мовлення. Функція втрат базується на різниці між відновленими і реальними сигналами в частотно-часовій області.

Запропонований метод демонструє переваги високого порядку амбісоніки у задачах розділення мовлення. Просторова інформація дозволяє враховувати фізичне розташування джерел, зменшуючи перехресні спотворення. Модель адаптується до різних типів записів та кількості джерел.

HOA-сигнали третього порядку генерують велику кількість даних, що потребує значних обчислювальних ресурсів. Модель може потребувати адаптації для роботи з іншими порядками амбісоніки або різними типами шумового середовища. Ефективність перевірена переважно на синтетичних даних. Реальні записи можуть мати більш складні просторові характеристики.

2.2.2 Wave-U-Net

Wave-U-Net є важливим кроком у розвитку моделей для розділення джерел звуку, пропонуючи ефективний та простий підхід до аналізу аудіосигналів у часовій області [4]. Завдяки своїй архітектурі модель забезпечує високу якість розділення джерел без необхідності переходу до часочастотної області, що робить її перспективним інструментом для широкого кола аудіозастосувань.

Основу підходу становить Wave-U-Net, модифікована версія U-Net, яка оперує безпосередньо аудіосигналами та використовує багатомасштабну структуру для захоплення особливостей як на дрібних (локальних), так і великих (глобальних) часових масштабах.

Архітектура Wave-U-Net включає шари згортки (convolution) та апскейлінгу (upscaling) для виділення та реконструкції аудіосигналу. Немає використання пулінгу (pooling), що дозволяє зберігати часову роздільність. Багатомасштабний підхід забезпечує захоплення контексту як локального, так і глобального рівня. В

якості функції втрат використовується L1-норма різниці між оригінальним і реконструйованим сигналом, що забезпечує точність у часовій області.

Модель тренується на датасеті, що містить треки з різними джерелами (вокал, інструменти). Використовуються аудіотреки у форматі моно для спрощення обчислень.

На рис. 2.4 зображено просту архітектуру запропонованої моделі Wave-U-Net із K джерелами та L блоків із згорткових шарів [4], де в кожному блоці присутні шари зниження та збільшення розмірності.

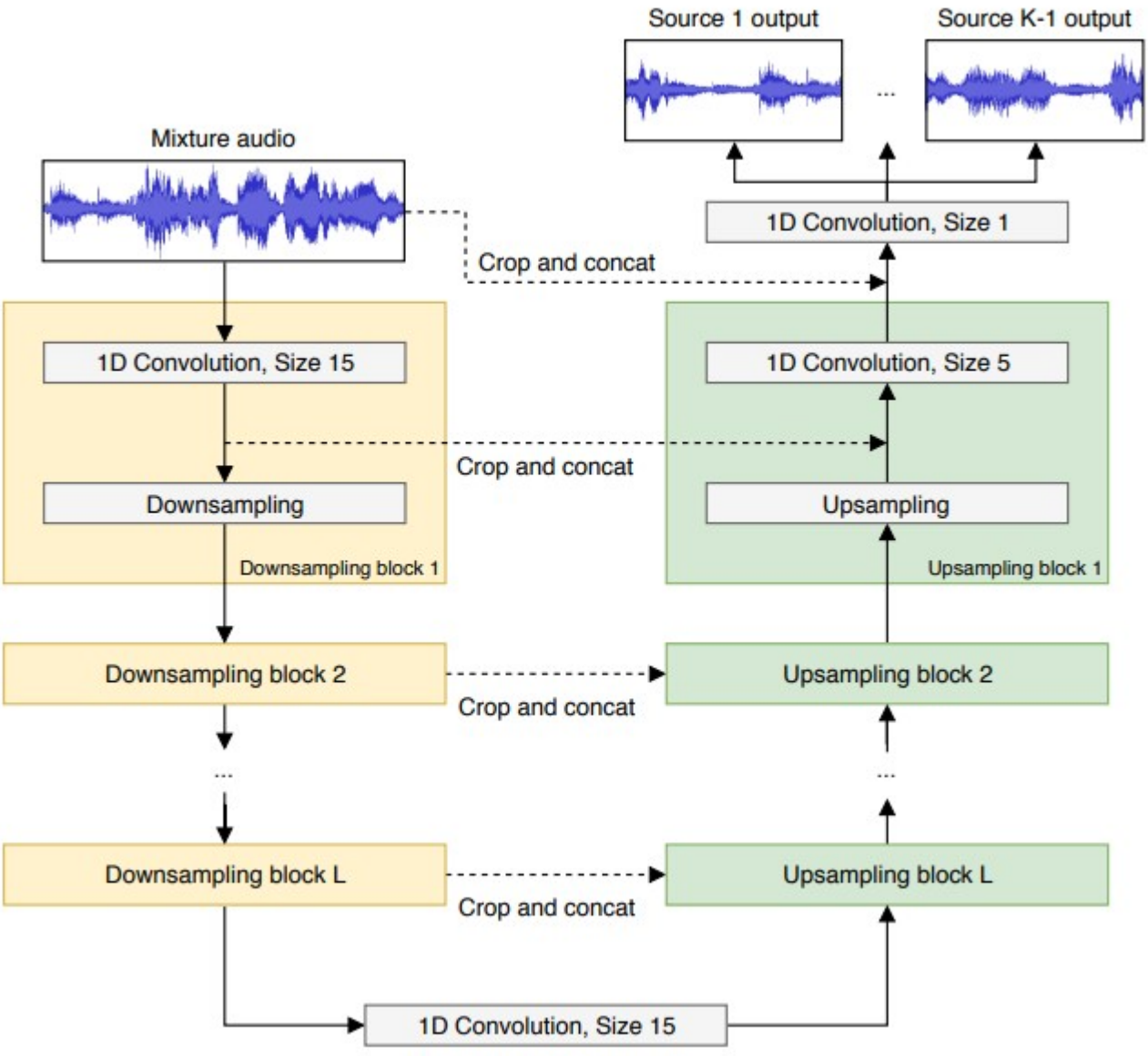


Рис. 2.4 Архітектура моделі Wave-U-Net

Модель показала конкурентні результати порівняно з методами, що використовують часочастотне представлення, зокрема у задачах виділення вокалу. Звідси слідує, що модель зменшує втрати деталізації сигналу та легше масштабується для роботи з довгими аудіосигналами. Оцінка за метриками SDR та SAR (Signal-to-Artifacts Ratio) виявила, що Wave-U-Net суттєво зменшує артефакти. Wave-U-Net виявився універсальним, демонструючи ефективність у задачах розділення джерел різного типу. Модель працює швидше завдяки відсутності потреби у перетворенні сигналів між часовою та частотною областями.

Часове представлення сигналу є високорозмірним, що потребує значних обчислювальних ресурсів. Для треків із сильним накладанням джерел модель іноді втрачає точність. Як і більшість сучасних моделей, Wave-U-Net вразливий до недостатньо представлених варіацій у навчальному наборі.

2.2.3 DPRNN-based Speaker Separator

Стаття демонструє інноваційний підхід (Dual-path RNN based Speaker Separator) до розділення голосів із невідомою кількістю мовців, що є важливим кроком у напрямку розвитку технологій обробки мови [1]. Результати підходу мають практичну цінність для багатьох сфер, включаючи розробку розумних пристроїв та удосконалення систем телекомунікації.

Архітектура нейронної мережі включає End-to-End підхід та Deep Clustering і Mask Inference. Модель не потребує попередньої обробки сигналу. Аудіосигнал представляється у вигляді спектрограм, що полегшує виділення індивідуальних компонентів. Для розділення голосів застосовуються техніки кластеризації ознак та масок, що дозволяє розподіляти звукові сигнали по різних каналах. Введено механізм, який адаптивно визначає кількість мовців у сигналі. Використовуються рекурентні моделі, що дозволяють "розуміти" зміну кількості джерел у часі. Для навчання використовуються спеціалізовані функції втрат, які орієнтовані на покращення сегрегації голосів навіть у складних сценаріях.

Запропонована модель перевершує сучасні аналоги на стандартних наборах даних (WSJ0-2mix та WSJ0-3mix) для задач розділення голосів. Модель успішно працює для записів із різною кількістю мовців. Вимірювання за метриками SDR та SIR показали значне поліпшення порівняно з попередніми методами. Хоча модель потребує значних обчислювальних ресурсів, автори запропонували стратегії для її оптимізації.

2.2.4 SepTDA

Запропонована модель (Separation Transformer Decoder-based Attractor) є суттєвим кроком у сфері розділення мовлення, зокрема в умовах невідомої кількості мовців [16]. Завдяки використанню трансформерів, метод демонструє високу якість і потенціал для подальшого розвитку.

Модель складається з блоку подвійної обробки (Dual-Path Processing Block), модуля атракторів на основі трансформер-декодера (Transformer decoder-based attractor), а також блоку потрійної обробки (Triple-Path Processing Block). Блок подвійної обробки відповідає за моделювання часочастотних закономірностей у сигналі. Модуль атракторів на основі трансформер-декодера використовує невеликий набір попередньо визначених запитів мовців для визначення кількості джерел. Генерує вектори атракторів для кожного мовця, які використовуються для виділення відповідних сигналів. Блоку потрійної обробки забезпечує обробку внутрішньочастотних, міжчастотних і міжмовцевих взаємодій. Для тренування використовується функція втрат, заснована на метриках SI-SDR і SDR, яка забезпечує точність реконструкції звуків.

2.2.5 Порівняльна характеристика

Найкраща модель для роботи з багатоканальними аудіо із розглянутих є DNN-based HOA Source Separation. Високий потенціал для розділення з великою кількістю мовців на основі трансформерів демонструє модель SepTDA. Простоту та ефективність надає модель Wave-U-Net. В таблиці 2.2 проведено підсумковий аналіз характеристик найсучасніших методик.

Таблиця 2.2

Порівняльна характеристика найсучасніших методик мультиголосового поділу

Методика	Переваги	Недоліки	Методи навчання	Обмеження
DNN-based HOA Source Separation	Інтеграція просторової інформації	Висока обчислювальна складність з великим обсягом HOA	LSTM із просторовими ознаками	Важке масштабувати для реального часу
Wave-U-Net	Прямий аналіз у часовій області	Важке масштабувати на багатоканальні сигнали	Обернене STFT і маскування в часовій область	Не працює з багатоканаль-ними даних
DPRNN-based Speaker Separator	Висока точність сегментації	Висока обчислювальна складність з багатьма мовцями	End-to-end підхід із кластеризацією та масками	Висока складність в оптимізації для великих систем
SepTDA	Висока продуктивність із великою кількістю мовців	Висока обчислювальна складність через використання трансформерів	SI-SDR, SDR, та моделювання	Ефективність залежить від якості навчальних даних

3 РОЗРОБКА МЕТОДИКИ МУЛЬТИГОЛОСОВОГО ПОДІЛУ НА ОСНОВІ НЕЙРОННИХ МЕРЕЖ

3.1 Розробка методики

В даній методиці особлива увага приділяється розділенню схожих голосів, а це в свою чергу потребує використання креативних рішень до вирішуваної проблеми.

У дослідженні [4] представлена модель працює з вхідним сигналом, застосовуючи при цьому механізми маскування, для виявлення ознак. На жаль такий підхід не може бути застосований, оскільки поділ поступово втрачає свою точність при сильному перекритті джерел. Відповідно до цього вхідними даними до нейронної мережі будуть спектральні ознаки, вилучені зі спектрограми сигналу за допомогою згорткової нейронної мережі. Такий підхід дозволить не лише зменшити кількість вхідних даних, а й допоможе моделі працювати як з локальними так і глобальними часочастотними характеристиками сигналу.

Новітня модель [1] значно перевершує існуючі рішення, однак нещодавнє дослідження [16] задало новий стандарт у задачах поділу голосів з невідомою кількість мовців. Адаптована архітектура трансформерів для роботи зі спектральним представленням об’єднує у собі найсучасніші підходи обробки цифрових сигналів: FFN, CNN, LSTM та FilM (Feature-wise Linear Modulation).

Значна перевага архітектури трансформерів над RNN стала вирішальним критерієм у виборі базової методики мультиголосового поділу. По-перше, це здатність адаптувати поділ при сильному спектральному перекритті джерел. В контексті схожих голосів, перекриттям можна вважати наявність спільних або подібних спектральних ознак. По-друге, ця архітектура здатна виявляти довготривалі залежності, що є не менш важливим ніж адаптація. По-третє, трансформери підтримують паралельну обробку даних (в шарах мультиуваги), що значно прискорює процес навчання.

3.1.1 Огляд методики

Модифікація трансформера для мультиголосового поділу схожих голосів потребує врахування специфічних характеристик мовлення, таких як основний тон, гармоніки і форманти. Запропоноване рішення включає передобробку даних, модифікація шарів трансформера та постобробку.

Передобробка складається з передобробки даних із використанням спектрограм і аугментація аудіо для покращення узагальнюваної здатності моделі [1], генерації мультирезолюційних спектрограм, з метою виявити різні аспекти сигналу. Також присутній розширений ембеддинг шляхом застосування згорткової нейронної мережі для вилучення ознак, адаптований позиційний ембеддинги, що враховує часові та частотні аспекти сигналу.

Модифікація шарів трансформера включає ієрархічну увага (побудова багаторівневої структури уваги для захоплення як локального, так і глобального контексту мовлення) [16], та часочастотний позиційний ембеддинг (інтеграція специфічних ознак про положення у часочастотній області, дозволяючи моделі враховувати ці залежності в процесі навчання) [4].

Постобробка містить декодер, оптимізований для відновлення голосів у часочастотному просторі та артефактну фільтрацію та відновлення гармонічної структури сигналу.

На рис. 3.1 схематично зображено модифіковану архітектуру трансформерів.

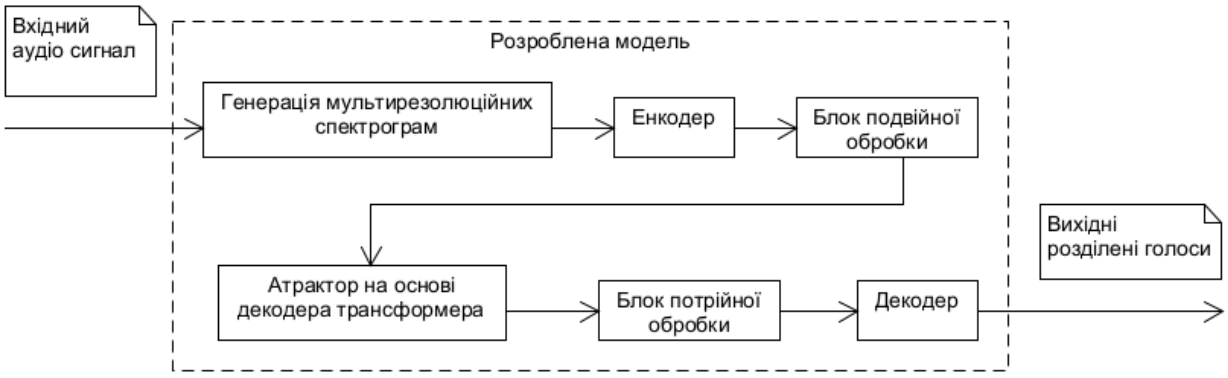


Рис. 3.1 Модифікована архітектура трансформерів для мультиголосового поділу

Методика представляє собою реалізацію моделі глибокого навчання для задачі розділення мовлення, зокрема модифікованої моделі під назвою SepSimTDA. Розглянемо основні компоненти моделі в порядку їх застосування.

Для покращення узагальнюючої здатності моделі застосовуються різні аугментації аудіо сигналів та генерує мультирезолюційні спектрограми.

2D Позиційне Ембеддінгування - реалізує двовимірне позиційне ембеддінгування для врахування позиційної інформації як по частоті, так і по часу в спектрограмі. Ініціалізація потребує наступних параметрів: розмірність ембеддінгу, максимальна довжина для позицій. Вхідний тензор "x" має форму "(batch_size, channels, freq_bins, time_steps)".

Енкодер з CNN та 2D Позиційними Ембеддінгами - перетворює вхідний аудіосигнал у високорівневі ознаки. Має кілька послідовних згорткових шарів з нормалізацією та активацією ReLU та позиційне ембеддінгування додає позиційну інформацію до виходу згорткових шарів. Екодер переформатовує тензор для подальшої обробки, який має розмір "(batch_size, seq_length, embedding_dim)". На рис. 3.2 схематично зображено модифікацію енкодера трансформера.

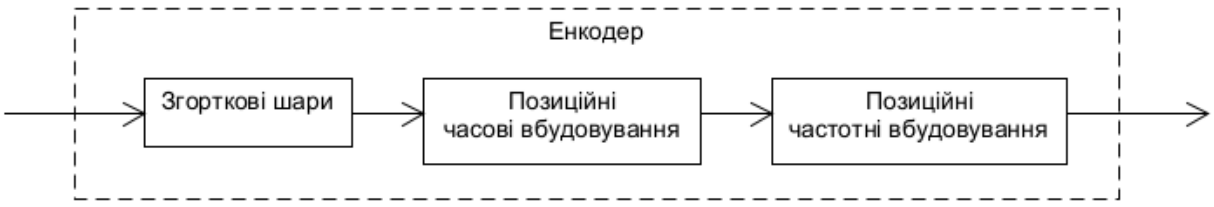


Рис. 3.2 Модифікований енкодер трансформера для мультиголосового поділу

Блок подвійної обробки (LSTM з механізмом уваги) - цей блок поєднує двонаправлений LSTM з механізмом багаторазової уваги для захоплення довготривалих залежностей у даних. Складається з двонаправленого LSTM для обробки послідовностей, механізму мультиголової уваги, для фокусування на різних частинах послідовності, FFN з активацією GELU. Додатково включає "LayerNorm" для стабілізації навчання.

Трансформер-Декодер Атрактор - використовується для отримання атракторів (представлень спікерів) із контексту. Містить параметричні запити, які навчаються, декодерні шари (послідовність шарів трансформер-декодера), лінійний шар, для проекції виходу (атракторів) декодера.

Блок потрійної обробки - реалізує потрійний шлях обробки для покращення моделювання послідовностей. Містить компоненти для внутрішньо-блочної та міжблочної обробки та шар трансформера для взаємодії між спікерами.

Декодер - відновлює розділені аудіосигнали зі зворотним перетворенням ознак. Включає шари транспонованої згортки, для збільшення розмірності та відновлення сигналу (фільтрація артефактів,. відновлення гармонічної структури аудіо). На рис. 3.3 схематично зображено модифікацію декодера трансформера.

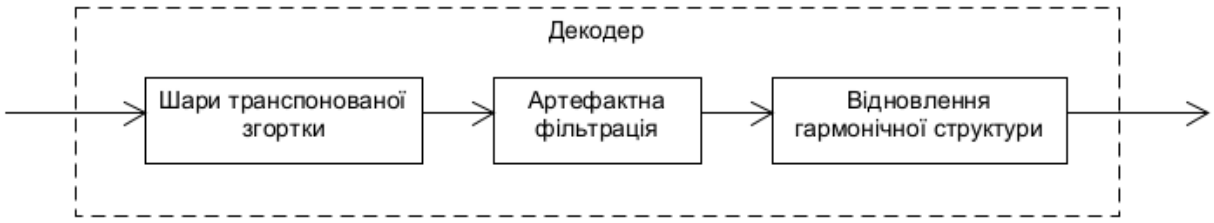


Рис. 3.3 Модифікований денкодер трансформера для мультиголосового поділу

Транспонована згортка (обернена згортка) використовується для збільшення розмірності даних (протилежно до згортки з підвибіркою). У Декодері вони дозволяють відновити просторову (часову та частотну) роздільну здатність сигналу з латентного простору (місце, де система зберігає ключову інформацію). Транспонована згортка допомагає відтворити деталі аудіосигналу, які були стиснуті або зменшені в процесі енкодингу, забезпечуючи плавність та точність відновленого сигналу.

На рис. 3.4 представлено діаграму класів, які реалізують модифіковану архітектуру трансформера.

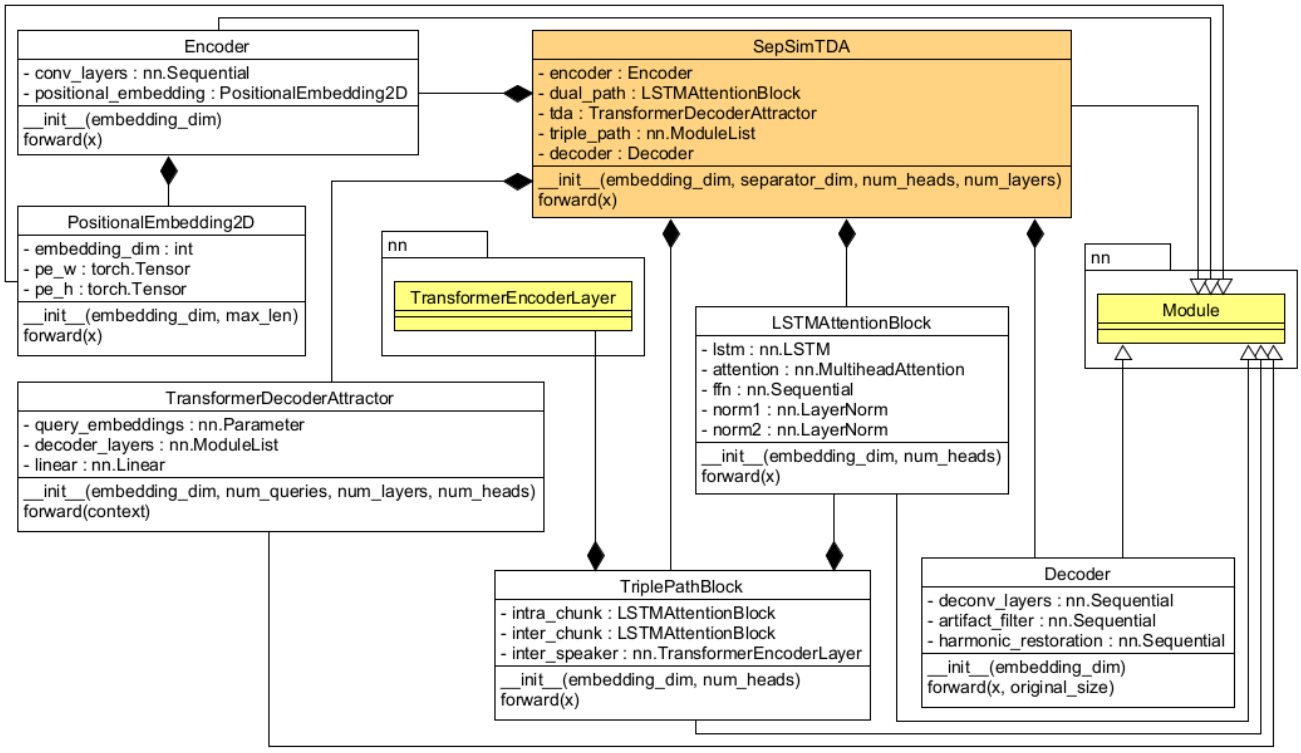


Рис 3.4 Діаграма класів розробленої моделі SepSimTDA

Кожен з класів (PositionalEmbedding2D, Encoder, LSTMAttentionBlock, TransformerDecoderAttractor, TriplePathBlock, Decoder, SepSimTDA) наслідується від базового класу nn.Module. Це означає, що вони всі використовують інтерфейс і функціональність, що надається базовим класом, такі як методи ініціалізації та виклику.

Encoder містить екземпляр PositionalEmbedding2D. Це означає, що PositionalEmbedding2D є частиною Encoder, і його життєвий цикл керується Encoder.

SepSimTDA містить екземпляри Encoder, LSTMAttentionBlock, TransformerDecoderAttractor, TriplePathBlock і Decoder. SepSimTDA є головним класом моделі, який включає в себе всі ці компоненти, координуючи їх взаємодію для виконання завдання розділення мови. SepSimTDA також може бути розглянутий як агрегатор TriplePathBlock, збираючи виходи з кількох екземплярів

цього блоку (через nn.ModuleList), що дозволяє моделі масштабувати обробку в залежності від конкретних потреб завдання.

TriplePathBlock використовує два екземпляри LSTMAttentionBlock і один nn.TransformerEncoderLayer. Це вказує на те, що TriplePathBlock об'єднує функціональність цих двох блоків для виконання більш складних завдань обробки мови, розподіляючи обробку сигналів на різні шляхи (внутрішній, між чанками, і між спікерами).

3.1.2 Синтез набору даних зі схожими голосами

Існуючі набори даних не можуть створити необхідне мовне середовище зі схожими голосами. Для створення власного датасету схожих і дуже реалістичних голосів можна використовувати попередньонавчені моделі Tacotron2 та WaveGlow, як дозволяють перетворювати текст в мовлення. Використовуються в поєднанні з Python (дозволяє створювати кастомізовані аудіофайли) та потребує певного налаштування, даючи велику гнучкість.

Для створення набору даних необхідно декілька мовців (голосів) та певна кількість записів однакової довжини для їх міксування в одноканальне аудіо. Загальну кількість комбінацій аудіо з 2-4 мовцями (із 6 можливих, де для кожного є 5 записів) становить 12 250. У створенні набору даних, який ефективно розділяє схожі голоси, особливо важливо залучати різні елементи вокального тракту. Відповідно кожна репліка мовця складається з речення, тривалість якого приблизно 5 секунд.

Мішані аудіо зберігаються з частотою дискретизації 16,000 Гц, що є стандартною частотою для обробки мовлення. Нормалізація аудіо забезпечує, що всі аудіосигнали мають однаковий рівень гучності перед змішуванням, щоб уникнути домінування одного сигналу над іншими. Метадані, збережені у CSV-файлі, дозволяючи відстежувати, які джерела були використані для кожного мішаного запису.

3.1.3 Огляд програмного застосунку

Користувач може завантажувати аудіофайли, запускати процес розділення, зберігати результати та переглядати спектрограми як оригінального аудіо, так і розділених джерел. Інтерфейс забезпечує зручну взаємодію з користувачем та демонструє основні можливості моделі. Коли користувач натискає кнопку "LoadAudio" і вибирає аудіофайл формату WAV. Після завантаження аудіо розблоковується кнопка "ShowSpectrogram". Користувач може натиснути цю кнопку для відображення спектрограми завантаженого аудіо у новому вікні. Кнопка "Separation" розблоковується після завантаження аудіо. На рис. 3.5 показано стани програмного застосунку до і після завантаження аудіо.

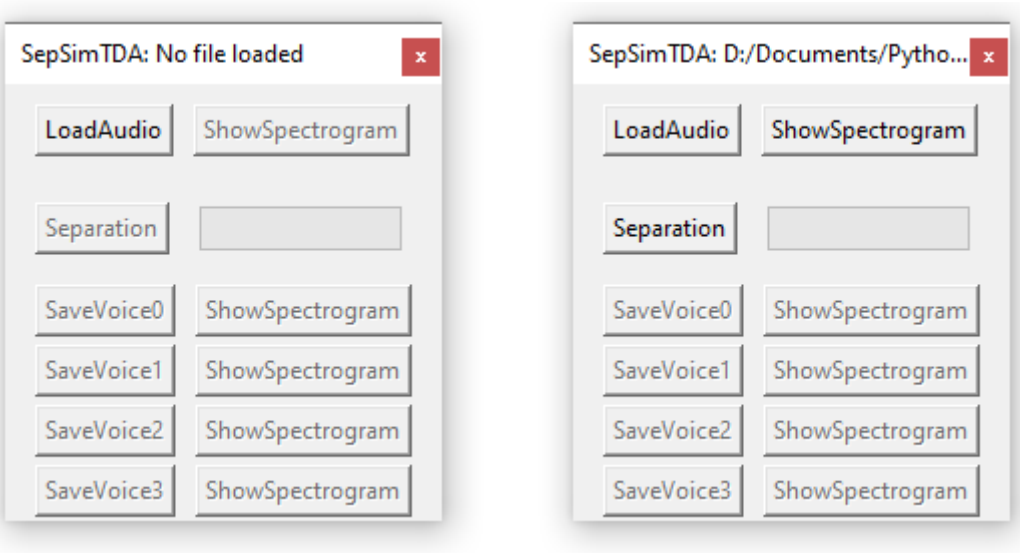


Рис. 3.5 Стан програмного застосунку до (зліва) і після (справа) завантаження аудіо

Користувач натискає кнопку "Separation" для запуску процесу розділення. Під час розділення всі кнопки блокуються (крім кнопок "ShowSpectrogram"), а прогрес-бар відображає прогрес. Після завершення процесу розділення розблоковуються кнопки для збереження та відображення спектрограм джерел. На рис. 3.6 зображено стан програмного застосунку під час і після запуску розділення аудіо.

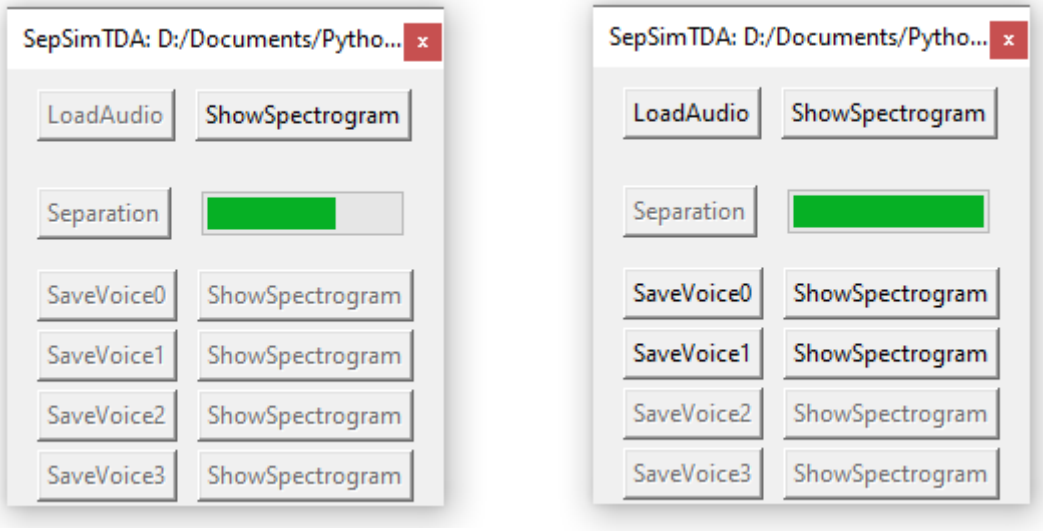


Рис. 3.6 Стан програмного застосунку під час (зліва) і після (справа) розділення аудіо

Користувач може натиснути кнопку "SaveVoiceX" для збереження розділеного аудіо джерела (де X — номер розділеного голосу). Кнопка "ShowSpectrogram" дозволяє відобразити спектрограму відповідного джерела у новому вікні. На рис. 3.7 зображено приклад спектрограми вхідного аудіо.

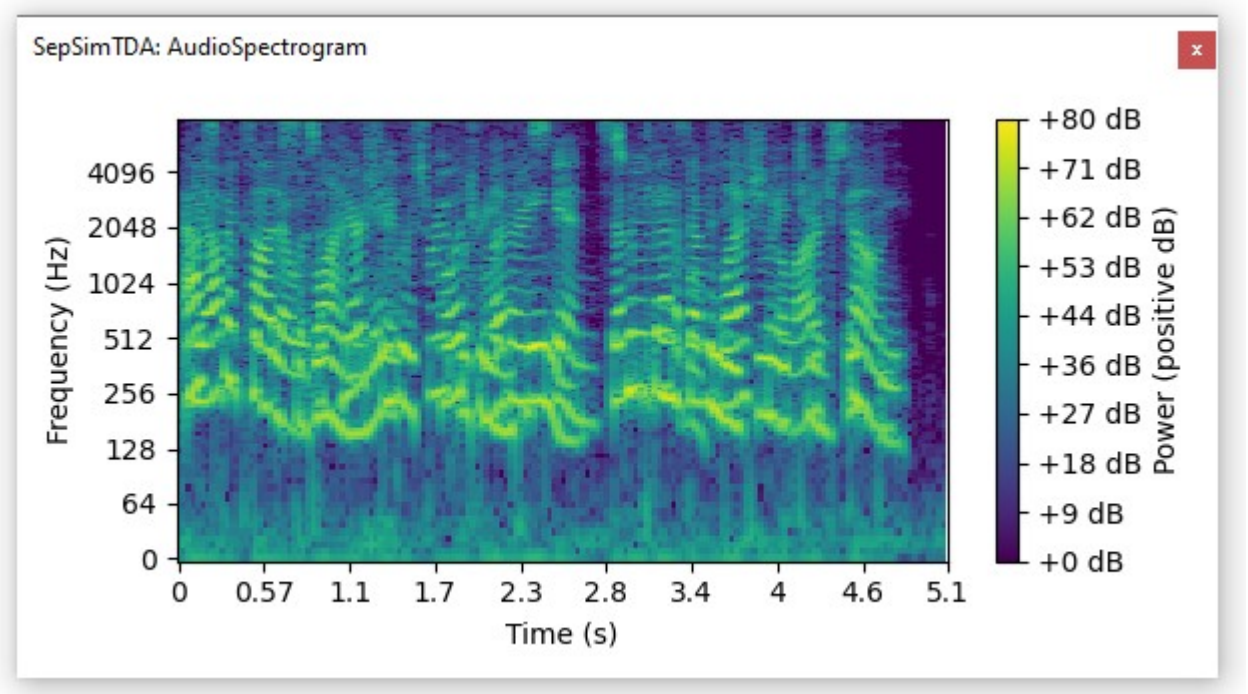


Рис. 3.7 Приклад спектрограми вхідного аудіо

Оскільки вхідне аудіо містить лише двох мовців на рис. 3.8 зображено дві спектрограми розділених голосів із вхідного аудіо.

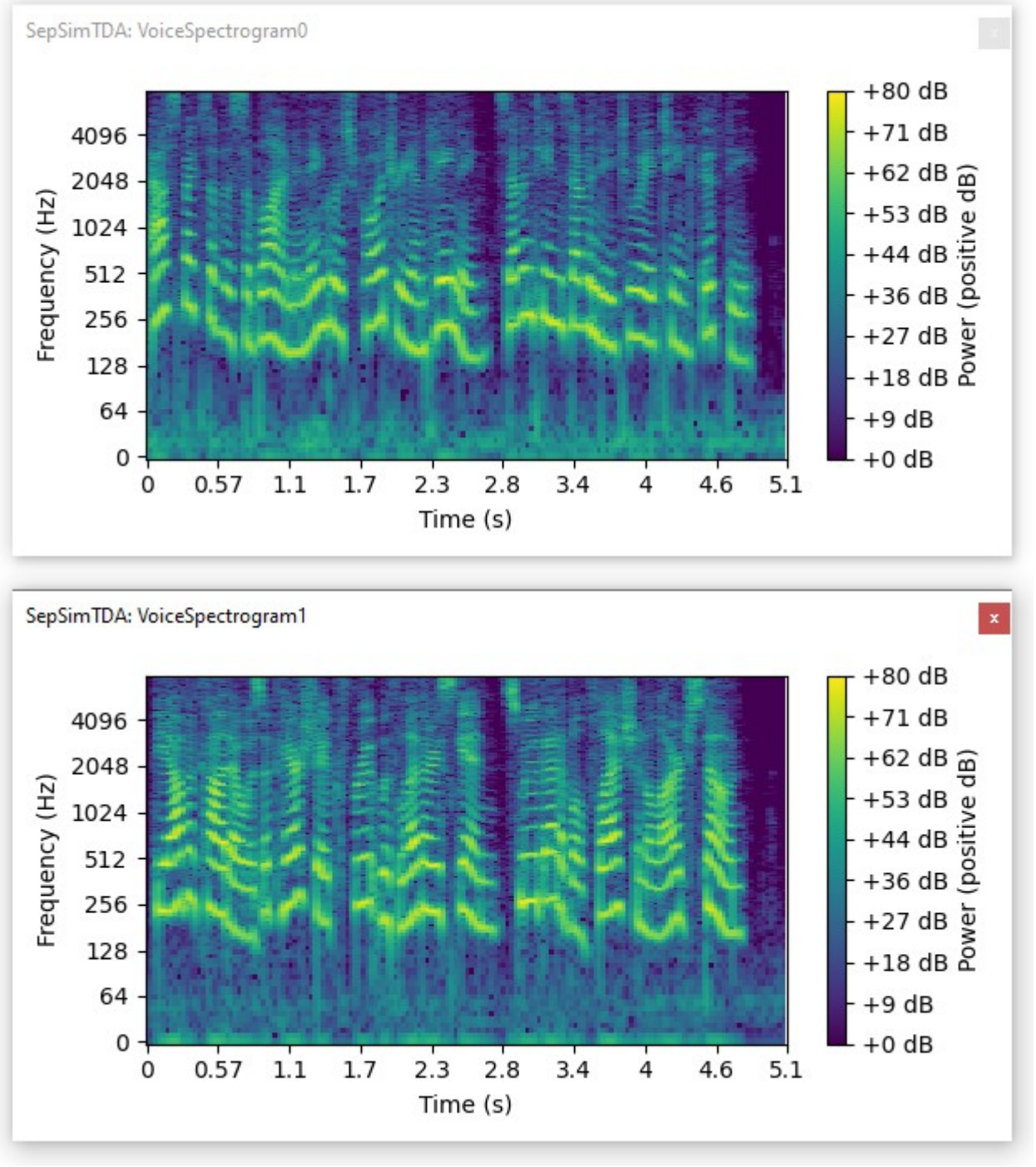


Рис. 3.8 Спектрограми першого (вгорі) та другого (знизу) розділених голосів

На рис. 3.9 зображено діаграму діяльності користувача в програмному застосунку.

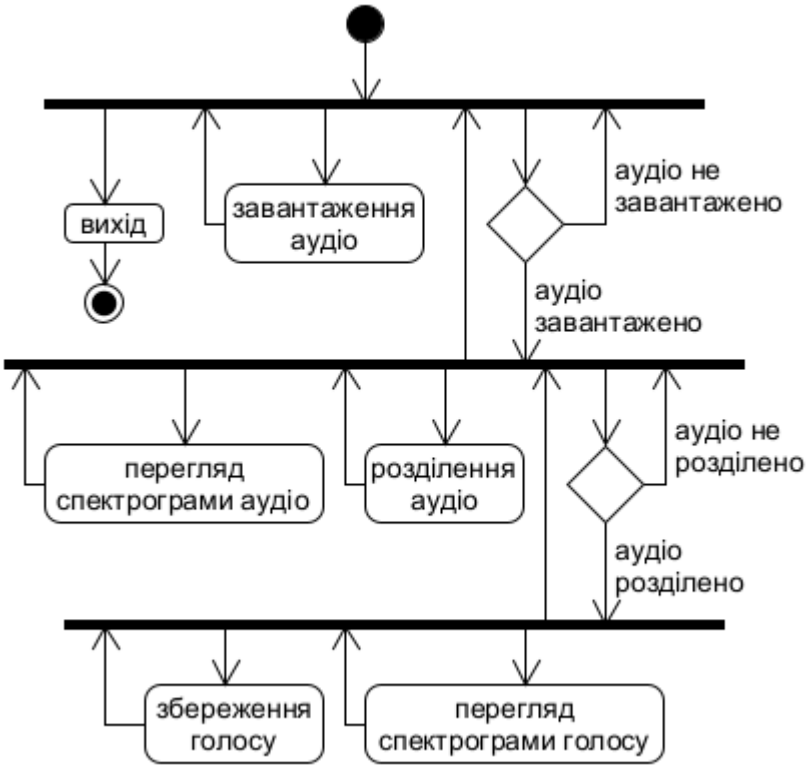


Рис. 3.9 Діаграма діяльності для застосунку SepSimTDA

В підсумок на рис. 3.10 представлено діаграму прецедентів, яка описує взаємодію користувача з усіма функціями програмного застосунку.

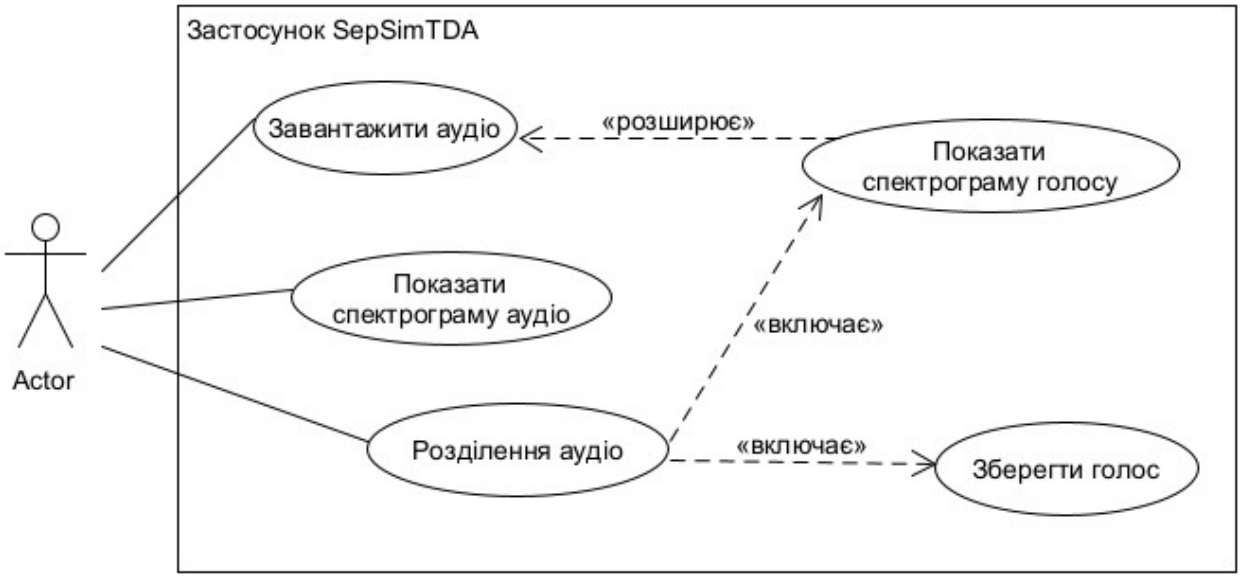


Рис. 3.10 Діаграма прецедентів для застосунку SepSimTDA

3.2 Опис використаних програмних засобів

Детальний аналіз використаних програмних засобів, які забезпечують функціональність для обробки аудіо, побудови моделей глибокого навчання, роботи з графічним інтерфейсом та управління даними. Нижче наведено детальний аналіз кожної з використаних бібліотек та фреймворків.

PyTorch — це фреймворк з відкритим кодом для глибокого навчання, розроблений компанією Facebook's AI Research lab. Підтримує динамічні обчислювальні графи, що робить його зручним для досліджень та розробки моделей. Має потужну підтримку GPU та API, схожий на NumPy, але з можливістю автоматичного диференціювання.

PyTorch використовується для створення складної моделі глибокого навчання для задачі розділення мовлення. Модулі "torch.nn", "torch.nn.functional" та "torch.optim" використовуються для визначення нейронних мереж, активаційних функцій та оптимізаторів. "nn.Module" служить базовим класом для всіх модулів (шарів) моделі. Фреймворк використовується для роботи з тензорами при нормалізації аудіо та змішуванні аудіосигналів."torch.sum", "torch.abs", "torch.max" та інші функції використовуються для обробки аудіоданих.

Динамічні обчислювальні графи роблять PyTorch зручним для дослідницьких проектів та швидкого прототипування. Потужне автоматичне обчислення градієнтів (диференціювання) спрощує процес навчання моделей. Широке використання в науковому середовищі забезпечує наявність ресурсів та допомоги.

TorchAudio — це бібліотека для обробки аудіо, інтегрована з PyTorch. Забезпечує завантаження, збереження та трансформації аудіоданих у вигляді тензорів PyTorch. Підтримує різні аудіоформати та частоти дискретизації.

Використовується для завантаження аудіоданих та обробки спектрограм у класі "SpeechSeparationDataset". Забезпечує функції для перетворення аудіо та аугментації. Використовується для завантаження та збереження аудіофайлів при

створенні мішаних записів. Функції "torchaudio.load" та "torchaudio.save" дозволяють працювати з аудіофайлами різних форматів.

Інтеграція з PyTorch забезпечує безшовну роботу з тензорами PyTorch, спрощуючи обробку аудіо в рамках нейронних мереж. Дозволяє легко завантажувати та зберігати аудіо у різних форматах. Містить готові інструменти для обробки та аугментації аудіосигналів.

NumPy — бібліотека для наукових обчислень у Python, яка забезпечує підтримку багатовимірних масивів та матриць, а також велику кількість високорівневих математичних функцій.

Використовується для роботи з масивами при обчисленні спектрограм. Функції "np.abs", "np.max", "np.linspace", "np.hanning" та інші використовуються у процесі обробки аудіо. Внутрішньо використовується в PyTorch для обробки тензорів.

Базова бібліотека для наукових обчислень NumPy є стандартом де-факто для числових обчислень у Python. Оптимізовані C-функції забезпечують швидку обробку великих масивів даних. Широко підтримується іншими бібліотеками, такими як librosa та matplotlib.

Pandas — це бібліотека для аналізу та маніпуляції даними, яка забезпечує структури даних та функції високого рівня. Основні структури даних — "DataFrame" та "Series".

Використовується для збереження метаданих про мішані аудіозаписи у форматі CSV. "pandas.DataFrame" створюється з метаданих, а потім зберігається за допомогою "to_csv". Завантаження метаданих з CSV-файлу для підготовки датасету. Обробка та парсинг метаданих про шляхи до аудіофайлів.

Pandas забезпечує зручність у роботі з табличними даними (просте завантаження, маніпуляція та збереження даних у різних форматах), включає фільтрацію, групування, агрегацію та інші операції над даними. Легко працює з NumPy та іншими науковими бібліотеками.

Matplotlib — це бібліотека для створення статичних, анімованих та інтерактивних візуалізацій у Python. Підтримує різноманітні типи графіків та налаштувань.

Використовується для побудови та відображення спектрограм аудіосигналів. Створює фігури та осі для візуалізації за допомогою "plt.subplots". Інтегрується з tkinter через "FigureCanvasTkAgg" для відображення графіків у GUI.

Matplotlib обрано через широкі можливості візуалізації, оскільки підтримує складні графіки та налаштування, працює з NumPy, pandas, та іншими науковими бібліотеками. Дозволяє вбудовувати графіки в tkinter та інші GUI-фреймворки.

Tkinter — це стандартний інтерфейс Python до інструментарію Tk GUI. Дозволяє створювати графічні інтерфейси користувача з використанням стандартних віджетів.

Використовується для створення графічного інтерфейсу користувача (GUI) для моделі SepSimTDA. Забезпечує вікна, кнопки, прогрес-бари, діалогові вікна та інші елементи інтерфейсу. Віджети "Tk", "Button", "Frame", "Label", "Progressbar" та інші використовуються для побудови інтерфейсу.

Стандартна бібліотека tkinter входить до стандартної бібліотеки Python, тому не потребує додаткової інсталяції. Легкий для вивчення та використання, підходить для створення простих GUI. Достатня функціональність забезпечує необхідні елементи для побудови базових графічних інтерфейсів.

Ttk — модуль в складі tkinter, який надає сучасніші та покращені віджети для GUI. Має покращений дизайн, включає більш сучасні та стильні віджети. Розширює функціональність стандартних віджетів tkinter.

Використовується для створення прогрес-бару за допомогою "ttk.Progressbar". Забезпечує більш сучасний вигляд та додаткові можливості порівняно з класичними віджетами tkinter.

Librosa — це бібліотека Python для аналізу та обробки музичних та аудіосигналів. Забезпечує функції для завантаження аудіо, обчислення спектрограм, мел-спектрограм, MFCC та інших аудіо характеристик.

Використовується для завантаження аудіофайлів у функціях "load_mix" та "spectrogram". Обчислює спектрограми аудіосигналів за допомогою функцій "librosa.stft" і "librosa.amplitude_to_db". "librosa.display.specshow" використовується для візуалізації спектрограм. Може використовуватися в частині, де згадується "генерація спектрограми", хоча в коді це явно не показано.

Librosa пропонує широкий набір інструментів саме для аудіо аналізу. Інтуїтивний API для завантаження та обробки аудіо. Дозволяє легко створювати спектрограми та інші аудіо візуалізації.

Threading — стандартний модуль Python для роботи з потоками. Дозволяє запускати паралельні операції в межах одного процесу.

Використовується для запуску процесу розділення аудіо в окремому потоці, щоб не блокувати головний потік GUI. Забезпечує плавність роботи інтерфейсу під час виконання тривалих операцій. Об'єкт "Event" використовується для контролю зупинки потоку при закритті програми.

Дозволяє виконувати декілька завдань одночасно, що особливо корисно для GUI-програм. Модуль threading простий у використанні та інтеграції з іншими частинами програми.

OS — стандартний модуль Python для взаємодії з операційною системою. Забезпечує функції для роботи з файловою системою та шляхами.

Використовується для отримання списків файлів у директоріях мовців. Функції "os.listdir", "os.path.join", "os.makedirs", "os.path.isfile" використовуються для управління файлами та папками. Використовується для перевірки наявності файлів аудіо при підготовці датасету.

Модулю OS було обрано, оскільки він забезпечує всі необхідні функції для роботи з файловою системою, дозволяє писати код, що працює на різних операційних системах.

Random — стандартний модуль Python для генерації випадкових чисел та вибору випадкових елементів.

Використовується для випадкового вибору кількості мовців у мішаному записі. Функції "random.randint", "random.sample", "random.choice" застосовуються

для генерації випадкових комбінацій. Використовується в функціях аугментації аудіо, таких як "add_noise", "change_volume", "time_stretch", щоб додати випадкові зміни до аудіосигналів.

Модуль Random легкий у використанні для базових випадкових операцій. Надає всі необхідні методи для генерації випадковості в програмі.

Переваги вибору цих бібліотек:

— ефективність та швидкість розробки (використання готових бібліотек дозволяє розробникам зосередитися на логіці програми, не витрачаючи час на реалізацію базових функцій);

— більшість використаних бібліотек мають велику спільноту та добре задокументовані, що спрощує процес розробки та вирішення проблем;

— можливість масштабування (використання таких фреймворків, як PyTorch, дозволяє легко масштабувати моделі та використовувати GPU для прискорення обчислень).

3.3 Проведення тестування

Для проведення порівняння якості розділення схожих голосів кожної із моделей, необхідно попередньо навчити їх на синтезованому наборі даних. Для навчання використовувалася вибірка у розмірі 80% від усього набору, 64 епохи навчання, оптимізатор "AdamW", центральний процесор 13th Gen Intel(R) Core(TM) i5-13500H, графічний процесор NVIDIA GeForce RTX 3050 з обсягом пам’яті — 4 гБ. Для оцінки якості розділення мовлення краще всього застосовувати метрики SDR, SI-SDR (одиниці вимірювання — дБ, чим більше значення тим краща якість поділу).

SDR — метрика співвідношення сигналу до спотворення, обчислюється за формулою (3.1):

$$SDR(\hat{S}, S) = 10 \log 10 \left(\frac{\|S\|^2}{\|S - \hat{S}\|^2} \right), \tag{3.1}$$

де: S — реальний сигнал у часоамплітудній області;

$\hat{S}$ — спрогнозований сигнал у часоамплітудній області;

$\| \, . \, \|^2$ — квадрат норми сигналу (сума квадратів амплітуд сигналу).

Для обчислення SI-SDR необхідно знайти оптимальний масштабний коефіцієнт, який обчислюється за формулою (3.2):

$$\alpha = \frac{\langle S, \hat{S} \rangle}{\|S\|^2}. \tag{3.2}$$

SI-SDR — метрика масштабно-незалежного співвідношення сигналу до спотворення, обчислюється за формулою (3.3):

$$SI\text{-}SDR(\hat{S}, S) = 10 \log 10 \left(\frac{|\alpha S|^2}{|\alpha S - \hat{S}|^2} \right). \tag{3.3}$$

Якщо різниця між реальним та спрогнозованим сигналом велика, значення метрик SDR, SI-SDR будуть прямувати до нуля, та навпаки в противному випадку.

В таблиці 3.1 нижче наведені результати моделювання кожної моделі.

Таблиця 3.1

Результати моделювання моделей мультиголосового поділу

Модель	Час навчання (хв)	SDR навчальний набір (дБ)	SI-SDR навчальний набір (дБ)	Розмір моделі (мБ)
DNN-based HOA Source Separation	120	7.34	6.93	9.02
Wave-U-Net	90	8.61	8.14	76.29
DPRNN-based Speaker Separator	240	10.25	9.75	28.61

Продовження таблиці 3.1

Результати моделювання моделей мультиголосового поділу

Модель	Час навчання (хв)	SDR навчальний набір (дБ)	SI-SDR навчальний набір (дБ)	Розмір моделі (мБ)
SepTDA	210	11.16	10.67	80.87
SepSimTDA	330	11.66	11.16	336.75

Для моделі DNN-based HOA Source Separation отримано найгірші результати якості поділу, оскільки вона погано працює з одноканальними аудіо [22]. Розмір оригінальної моделі відносно малий через пряме використання RNN, відповідно до цього тривалість навчання доволі мала.

Для моделі Wave-U-Net отримано кращі результати якості поділу та швидкості навчання, ніж попередня модель, через пряму обробку сигналу в часоамплітудній області [4]. Обсяг використаної пам’яті значно більший, через застосування CNN, де використовується комплексна багатошарова структура.

Для моделі DPRNN-based Speaker Separator отримано середні значення обсягу використаної пам’яті та результатів якості поділу, через застосування глибоких RNN [1]. Оскільки модель використовує RNN та працює з часочастотною областю, тривалість навчання є доволі повільним.

Для моделі SepTDA отримано хороші результати точності поділу, часу навчання, а також обсягу використаної пам’яті [16]. Не зважаючи на те, що модель має більше параметрів ніж попередня, швидкість навчання значно більша завдяки використанню блоків мультиголової уваги, які обробляють дані незалежно (паралельно).

Для розробленої моделі SepSimTDA отримано найкращі результати точності поділу, однак по всім інших критеріям отримано найгірші результати. Проблема полягає в тому, що вище зазначена модель використовує велику кількість згорткових шарів, де пониження розмірності значно збільшує кількість

навчальних параметрів. Завдяки паралельній обробці, тривалість навчання доволі невелика.

Розмір кожної моделі розраховано за наступною формулою (3.4), при $fs=4$:

$$MS(M, fs) = \frac{fs \cdot PC(M)}{2^{20}}, \tag{3.4}$$

де MS — функція обчислення розміру моделі M в мБ;

M — модель, для якої обчислюється розмір;

fs — розмір числа плаваючою комою в байтах;

PC — функція обчислення кількості параметрів моделі M.

В таблиці 3.2 наведено результати тестування кожної моделі на тестовому наборі даних, який складає різницю між цілим та навчальним наборами.

Таблиця 3.2

Результати тестування моделей мультиголосового поділу

Модель	SDR тестовий набір (дБ)	SI-SDR тестовий набір (дБ)	Середня швидкість поділу аудіо (с)
DNN-based HOA Source Separation	7.0	6.5	2.5
Wave-U-Net	8.0	7.5	2.0
DPRNN-based Speaker Separator	10.5	10.0	3.0
SepTDA	11.0	10.5	2.8
SepSimTDA	11.9	11.3	4.1

Моделі DNN-based HOA Source Separation, Wave-U-Net та SepTDA дають гірші результати якості поділу на тестових даних, ніж на навчальних, оскільки

кожна із моделей потребує значний обсяг навчальних даних для якіснішого поділу [22, 1, 16].

Моделі DPRNN-based Speaker Separator та SepSimTDA дають кращі результати якості поділу на тестових даних, ніж на навчальних [4]. Перша модель завдяки глибоким RNN змогла виявити залежності в мовленні, однак друга модель має кращі результати завдяки використанню сучасніших технологій поділу мовлення.

3.4 Порівняльна характеристика розробленої методики

Для обчислення ефективності моделей було взято модель DPRNN-based Speaker Separator, яка має найбільшу кількість середніх значень. В таблиці 3.3 наведено результати оцінення якості поділу інших моделей відносно опосередкованої моделі [4].

Таблиця 3.3

Відсоткова різниця якості поділу моделей відносно моделі [4]

Модель	SDR навчальний набір (%)	SI-SDR навчальний набір (%)	SDR тестовий набір (%)	SI-SDR тестовий набір (%)
DNN-based HOA Source Separation	-28.39	-28.92	-33.33	-35
Wave-U-Net	-16	-16.51	-23.81	-25
SepTDA	+8.88	+9.44	+4.76	+5
SepSimTDA	+13.76	+14.46	+13.33	+13

В таблиці 3.4 наведено результати оцінення загальних характеристик інших моделей відносно опосередкованої моделі [4].

Таблиця 3.4

Відсоткова різниця характеристик моделей відносно моделі [4]

Модель	Час навчання (%)	Середня швидкість поділу аудіо (%)	Розмір моделі (%)
DNN-based HOA Source Separation	-50	-16.67	-68.47
Wave-U-Net	-62.5	-33.33	+166.66
SepTDA	-12.5	-6.67	+182.66
SepSimTDA	+37.5	+36.67	+1077.04

Розроблена модель SepSimTDA значно погіршує загальні характеристики, відносно інших моделей, натомість критичними для покращення елементами залишаються час навчання та в особливості розмір моделі, який перевищує [4] понад 1000%. Варто зазначити, що час, витрачений на обробку аудіо під час навчання та після (в попередньо навченій моделі), радикально відрізняється через розбіжність у використанні засобів обчислення (графічний процесор для навчання, та центральний процесор для використання).

В таблиці 3.5 наведено відсоткову різницю якості поділу SepTDA та розробленої моделей відносно один одного [16].

Таблиця 3.5

Відсоткова різниця якості поділу моделей SepTDA та SepSimTDA відносно один одного

Модель	SDR навчальний набір (%)	SI-SDR навчальний набір (%)	SDR тестовий набір (%)	SI-SDR тестовий набір (%)
SepTDA	-4.29	-4.39	-7.56	-7.1
SepSimTDA	+4.48	+4.59	+8.18	+7.62

Розроблена модель має приріст якості мультиголосового поділу, відносно найкращої [16] серед моделей, в середньому на 6.33%, використовуючи метрику SDR, та в середньому на 6.11%, використовуючи метрику SI-SDR.

В таблиці 3.6 наведено відсоткову різницю основних характеристик SepTDA та розробленої моделі відносно один одного [16].

Таблиця 3.6

Відсоткова різниця основних характеристик моделей SepTDA та SepSimTDA відносно один одного

Модель	Час навчання (%)	Розмір моделі (%)	Середня швидкість поділу аудіо (%)
SepTDA	-36.36	-75.99	-31.71
SepSimTDA	+57.14	+316.41	+46.43

Розроблена модель має значний приріст у використанні пам’яті, необхідної для збереження моделі, понад 300%, та не значний приріст в часі навчання та поділі аудіо.

В таблиці 3.7 наведено підсумковий аналіз моделювання та тестування усіх вище зазначених моделей мультиголосового поділу. Кожна із моделей має як слабкі так і сильні сторони, використання яких залежить від специфіки вирішуваної задачі.

Таблиця 3.7

Порівняльна характеристика найкращих моделей мультиголосового поділу

Модель	Переваги	Недоліки	Особливості	Обмеження
DNN-based HOA Source Separation	Використовує просторову інформацію	Обмежена ефективність для одноканальних аудіо	Адаптована для багатоканальних аудіо	Вимагає точні просторові вимірювання

Продовження таблиці 3.7

Порівняльна характеристика найкращих моделей мультиголосового поділу

Модель	Переваги	Недоліки	Особливості	Обмеження
Wave-U-Net	Пряма обробка часоамплітудної області	Низька точність при великому перекритті джерел	Проста архітектура для масштабування	Погана точність в динамічних аудіо
DPRNN-based Speaker Separator	Висока точність розділення джерел	Складна архітектура, дороговизна навчання	Стійкість в умовах перекриття джерел	Потребує великих обсягів навчальних даних
SepTDA	Адаптована для виявлення часових ознак	Складне налаштування для реальних умов	Висока чутливість до динаміки мовлення	Чутливість до фонових шумів
SepSimTDA	Висока точність розділення джерел	Вимагає значних обчислювальних ресурсів	Адаптована для розпізнавання схожих голосів	Працює лише з одноканальними аудіо, фіксована кількість мовців

ВИСНОВКИ

1. Розроблено та впроваджено методику мультиголосового поділу для схожих голосів на основі нейронних мереж.
2. Створено модифікацію архітектури трансформерів (SepTDA), яка покращує точність розділення схожих голосів.
3. Синтезовано власний штучний набір даних зі схожими голосами.
4. Навчено розроблену та існуючих архітектури на власному наборі даних.
5. Проведено аналіз ефективності та точності розділення мовців з існуючими архітектурами, й отримано приріст якості поділу відносно найкращої (SepTDA) на 6.33% та 6.11% з використанням метрик SDR та SI-SDR відповідно.
6. Розроблено застосунок для розділення мовців в аудіо файлі та перегляду спектрограм вхідного та розділеного сигналів.

ПЕРЕЛІК ПОСИЛАНЬ

1. Nachmani E., Adi Y., Wolf L. Voice Separation with an Unknown Number of Multiple Speakers. *PMLR 119 : Proceedings of the 37th International Conference on Machine Learning*, Vienna, 13-18 July 2020. P. 7164-7175.

2. McCallum C., Korzeniowski F., Oramas S., Gouyon F., Ehmann F. Supervised and Unsupervised Learning of Audio Representations for Music Understanding. *Proceedings of the 23rd International Society for Music Information Retrieval Conference*, Bengaluru, 4-8 December 2022. P. 256-263.

3. Huang P., Kim M., Hasegawa-Johnson M., Smaragdis P. Singing-Voice Separation from Monaural Recordings using Deep Recurrent Neural Networks. *Proceedings of the 15th International Society for Music Information Retrieval Conference*, Taipei, 27-31 October 2014. P. 477-482.

4. Stoller D., Ewert S., Dixon S. Wave-U-Net: A Multi-Scale Neural Network for End-To-End Audio Source Separation. *Proceedings of the 19th International Society for Music Information Retrieval Conference*, Paris, 23-27 September 2018. P. 334-340.

5. Kum S., Oh C., Nam J. Melody Extraction on Vocal Segments Using Multi-Column Deep Neural Networks. *Proceedings of the 17th International Society for Music Information Retrieval Conference*, New York City, 7-11 August 2016. P. 819-825.

6. Jun D., Yanhui T., Yong X., Lirong D., Chin-Hui L. Speech Separation of A Target Speaker Based on Deep Neural Networks. *Proceedings of the 12th International Conference on Signal Processing*, Hangzhou, 19-23 October 2014. P. 473-477.

7. Grais. E, Roma G., Simpson A., Plumbley M. Two Stage Single Channel Audio Source Separation using Deep Neural Networks. *Proceedings of the IEEE/ACM Transactions on Audio*, *Speech, and Language Processing*, September 2017. Vol. 25, No. 9. P. 1469-1479.

8. Yan-Hui T., Jun D., Li-Rong D., Chin-Hui L. Speech Separation Based on Signal-Noise-Dependent Deep Neural Networks for Robust Speech Recognition. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal*

Processing, 19-24 April 2015. P. 61-65.

9. Erdogan H., Hershey J., Watanabe S., Le Roux J. Phase-Sensitive and Recognition-Boosted Speech Separation using Deep Recurrent Neural Networks. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 19-24 April 2015. P. 708-712.

10. Nugraha A., Liutkus A., Vincent E. Multichannel Music Separation with Deep Neural Networks. *Proceedings of the 24rd European Signal Processing Conference*, Budapest, 28 August-2 September 2016. P. 1748-1752.

11. Yanhui T., Jun D., Yong X., Lirong D., Chin-Hui L. Speech Separation Based on Improved Deep Neural Networks with Dual Outputs of Speech Features for Both Target and Interfering Speakers. *Proceedings of the 9th International Symposium on Chinese Spoken Language Processing*, 12-14 September 2014. P. 250-254.

12. Iseli M., Shue Y., Alwan A. Age-And Gender-Dependent Analysis of Voice Source Characteristics. *Proceedings of the IEEE International Conference on Acoustics Speech and Signal Processing,* Toulouse, 14-19 May 2006. P. 389-392.

13. Tian G., Jun D., Li X., Cong L., Li-Rong D., Chin-Hui L. A Unified Speaker-Dependent Speech Separation And Enhancement System Based On Deep Neural Networks. *Proceedings of the IEEE China Summit and International Conference on Signal and Information Processing*, Chengdu, 12-15 July 2015. P. 687-691.

14. Valk R., Weyde T. Deep Neural Networks With Voice Entry Estimation Heuristics For Voice Separation In Symbolic Music Representations. *Proceedings of the 19th International Society for Music Information Retrieval Conference*, Paris, 23-27 September 2018. P. 281-288.

15. Shota I., Hirokazu K., Li L., Shoji M. Sepnet: A Deep Separation Matrix Prediction Network For Multichannel Audio Source Separation. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 6-11 June 2021. P. 191-195.

16. Younglo L., Shukjae C., Byeong-Yeol K., Zhong-Qiu W., Shinji W. Boosting Unknown-Number Speaker Separation with Transformer Decoder-Based Attractor. *Proceedings of the IEEE International Conference on Acoustics, Speech and*

Signal Processing, Seoul, 14-19 April 2024. P. 446-450.

17. Yuan W., Wang S., Li X., Unoki M., Wang W. A Skip Attention Mechanism for Monaural Singing Voice Separation. *Proceedings of the* I*EEE Signal Processing Letters*, October 2019, Vol. 26, No. 10. P. 1481-1485.

18. Yong X., Jun D., Li-Rong D., Chin-Hui L. An Experimental Study on Speech Enhancement Based on Deep Neural Networks. *Proceedings of the IEEE Signal Processing Letters*, January 2014, Vol. 21, No. 1. P. 65-68.

19. Huang P., Kim M., Hasegawa-Johnson M., Smaragdis P. Deep Learning for Monaural Speech Separation. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Florence, 4-9 May 2014. P. 1562-1566.

20. Hu X., Li K., Zhang W., Luo Y., Lemercier J., Gerkmann T. Speech Separation using an Asynchronous Fully Recurrent Convolutional Neural Network. *Proceedings of the 35th Conference on Neural Information Processing Systems*, 6-14 December 2021. P. 22509-22523.

21. Weninger F., Eyben F., Schuller B. Single-Channel Speech Separation with Memory-Enhanced Recurrent Neural Networks. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Florence, 4-9 May 2014. P. 3709-3713.

22. Perotin L., Serizel R., Vincent E., Guérin A. Multichannel speech separation with recurrent neural networks from high-order ambisonics recordings. *Proceedings of the 43rd IEEE International Conference on Acoustics, Speech, and Signal Processing*, Calgary, 15-20 April 2018. P. 36-40.

23. Jansson A., Bittner R., Ewert S., Weyde T. Joint Singing Voice Separation and F0 estimation with Deep U-Net Architectures. *Proceedings of the 27th European Signal Processing Conference*, A Coruna, 2-6 September 2019. P. 1-5.

24. Xian Y., Sun Y., Wang W., Naqvi M. Two Stage Audio-Video Speech Separation using Multimodal Convolutional Neural Networks. *Proceedings of the Sensor Signal Processing for Defence Conference*, Brighton, 9-10 May 2019. P. 1-5.

25. Wang Z., Watanabe S. UNSSOR: Unsupervised Neural Speech Separation by Leveraging over-determined Training Mixtures. *Proceedings of the 37th*

International Conference on Neural Information Processing Systems, New Orleans, 10-16 December 2023. P. 34021-34042.

26. Nakano T., Koyama Y., Hamasaki M., Goto M. Interactive Deep Singing-Voice Separation Based on Human-in-the-Loop Adaptation. *Proceedings of the 25th International Conference on Intelligent User Interfaces*, Cagliari, 17-20 March 2020. P. 78-82.

27. Nakamura T., Kozuka S., Saruwatari H. Time-Domain Audio Source Separation With Neural Networks Based on Multiresolution Analysis. *Proceedings of the IEEE/ACM Transactions on Audio, Speech, and Language Processing*, April 2021. Vol. 29. P. 1687-1701.

28. Yong X,, Jun D., Li-Rong D., Chin-Hui L. A Regression Approach to Speech Enhancement Based on Deep Neural Networks. *Proceedings of the IEEE/ACM Transactions on Audio, Speech, and Language Processing*, January 2015. Vol. 23, No. 1. P. 7-19.

29. Weninger F., Hershey J., Roux J., Schuller B. Discriminatively Trained Recurrent Neural Networks for Single-Channel Speech Separation. *Proceedings of the IEEE Global Conference on Signal and Information Processing*, Atlanta, 3-5 December 2014. P. 577-581.

30. Weninger F., Erdogan H., Watanabe S., Vincent E,, Roux J., Hershey J., Schuller B. Speech enhancement with LSTM recurrent neural networks and its application to noise-robust ASR. *Proceedings of the 12th International Conference on Latent Variable Analysis and Signal Separation,* Liberec, August 2015. P. 91-99.

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Магістерська робота

МЕТОДИКА МУЛЬТИГОЛОСОВОГО ПОДІЛУ НА ОСНОВІ НЕЙРОННИХ МЕРЕЖ

Виконав: студент групи ПДМ-61 Гапей Максим Юрійович

Керівник: к.т.н., доц., доцент кафедри ШІ Фесенко Максим Анатолійович

Київ - 2025

МЕТА, ОБ'ЄКТА ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета дослідження: підвищення точності мультиголосового поділу для схожих голосів за рахунок використання нейронних мереж.

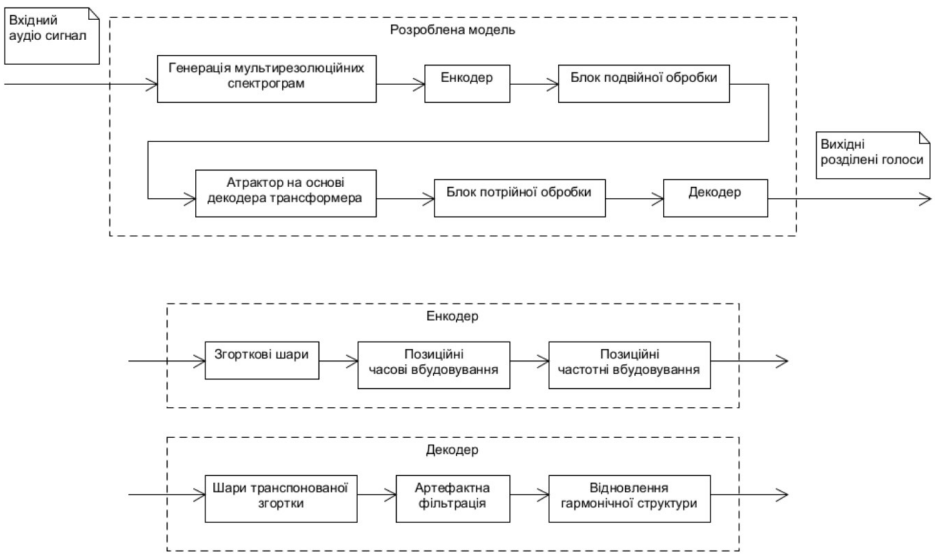
Об'єкт дослідження: процес мультиголосового поділу.

Предмет дослідження: методика мультиголосового поділу на основі нейронних мереж.

ПОРІВНЯЛЬНА ХАРАКТЕРИСТИКА

Модель	Переваги	Недоліки	Особливості	Обмеження
DNN-based HOA Source Separation	Використовує просторову інформацію	Обмежена ефективність для одноканальних аудіо	Адаптована для багатоканальних аудіо	Вимагає точні просторові вимірювання
Wave-U-Net	Пряма обробка часоамплітудної області	Низька точність при великому перекритті джерел	Проста архітектура для масштабування	Погана точність в динамічних аудіо
DPRNN-based Speaker Separator	Висока точність розділення джерел	Складна архітектура, дороговизна навчання	Стійкість в умовах перекриття джерел	Потребує великих обсягів навчальних даних
SepTDA	Адаптована для виявлення часових ознак	Складне налаштування для реальних умов	Висока чутливість до динаміки мовлення	Чутливість до фонових шумів
SepSimTDA	Висока точність розділення джерел	Вимагає значних обчислювальних ресурсів	Адаптована для розпізнавання схожих голосів	Працює лише з одноканальними аудіо, фіксована кількість мовців

МОДИФІКОВАНА АРХІТЕКТУРА ТРАНСФОРМЕРІВ



ВИКОРИСТАНІ ПРОГРАМНІ ЗАСОБИ

Мова програмування Python

Фреймворк PyTorch

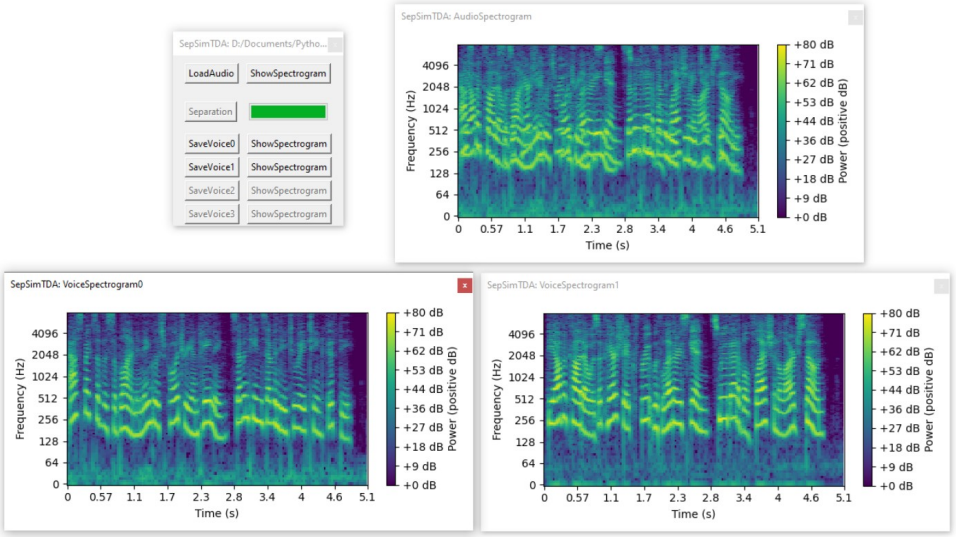
Бібліотека NumPy

Бібліотека Pandas

Бібліотека Matplotlib

Бібліотека Tkinter

ЕКРАННІ ФОРМИ



Приклад використання розробленого застосунку

РЕЗУЛЬТАТИ МОДЕЛЮВАННЯ

Набір даних: складається з 12250 одноканальних аудіо файлів тривалістю 5 секунд, кожен з яких містить від 2 до 4 мовців (із 6 можливих).

SDR – метрика співвідношення сигналу до спотворення.
SI-SDR – метрика масштабно-незалежного співвідношення сигналу до спотворення.

Модель	**Час навчання (хв)**	**SDR навчальний набір (дБ)**	**SI-SDR навчальний набір (дБ)**	**SDR тестовий набір (дБ)**	**SI-SDR тестовий набір (дБ)**	**Середня швидкість поділу аудіо (с)**	**Розмір моделі (мБ)**
DNN-based HOA Source Separation	120	7.34	6.93	7.0	6.5	2.5	9.02
Wave-U-Net	90	8.61	8.14	8.0	7.5	2.0	76.29
DPRNN-based Speaker Separator	240	10.25	9.75	10.5	10.0	3.0	28.61
SepTDA	210	11.16	10.67	11.0	10.5	2.8	80.87
SepSimTDA	330	11.66	11.16	11.9	11.3	4.1	336.75

ВИСНОВКИ

1. Розроблено та впроваджено методику мультиголосового поділу для схожих голосів на основі нейронних мереж.

2. Створено модифікацію архітектури трансформерів (SepTDA), яка покращує точність розділення схожих голосів.

3. Синтезовано власний штучний набір даних зі схожими голосами.

4. Навчено розроблену та існуючих архітектури на власному наборі даних.

5. Проведено аналіз ефективності та точності розділення мовців з існуючими архітектурами, й отримано приріст якості поділу відносно найкращої (SepTDA) на 6.33% та 6.11% з використанням метрик SDR та SI-SDR відповідно.

6. Розроблено застосунок для розділення мовців в аудіо файлі та перегляду спектрограм вхідного та розділеного сигналів.

ПУБЛІКАЦІЇ ТА АПРОБАЦІЯ РОБОТИ

Тези доповідей:

1. Гапей М.Ю., Фесенко М.А. Огляд особливостей мультиголосового поділу для схожих голосів. // Всеукраїнська науково-технічна конференція «Виклики та рішення в програмній інженерії». – Київ: ДУІКТ, 2024. С. 29-30.
2. Гапей М.Ю., Фесенко М.А. Модифікація архітектури трансформера для мультиголосового поділу схожих голосів. // Всеукраїнська науково-технічна конференція «Виклики та рішення в програмній інженерії». – Київ: ДУІКТ, 2024. С. 89-91.

ДОДАТОК Б. ЛІСТИНГ ПРОГРАМНИХ МОДУЛІВ

```
import torch
import torch.nn as nn
import torch.nn.functional as F
import torchaudio

"""Модифікована Модель"""
# 1. 2D Positional Embedding
class PositionalEmbedding2D(nn.Module):
    def __init__(self, embedding_dim, max_len=5000):
        super(PositionalEmbedding2D, self).__init__()
        self.embedding_dim = embedding_dim

        # Позиційні ембеддинги для часу
        position_w = torch.arange(0, max_len).unsqueeze(1).float()
        div_term_w = torch.exp(torch.arange(0, embedding_dim // 2, 2).float() * (-
torch.log(torch.tensor(10000.0)) / (embedding_dim // 2)))
        pe_w = torch.zeros(max_len, embedding_dim // 2)
        pe_w[:, 0::2] = torch.sin(position_w * div_term_w)
        pe_w[:, 1::2] = torch.cos(position_w * div_term_w)
        pe_w = pe_w.unsqueeze(2)

        # Позиційні ембеддинги для частоти
        position_h = torch.arange(0, max_len).unsqueeze(1).float()
        div_term_h = torch.exp(torch.arange(0, embedding_dim // 2, 2).float() * (-
torch.log(torch.tensor(10000.0)) / (embedding_dim // 2)))
        pe_h = torch.zeros(max_len, embedding_dim // 2)
        pe_h[:, 0::2] = torch.sin(position_h * div_term_h)
```

```
        pe_h[:, 1::2] = torch.cos(position_h * div_term_h)
        pe_h = pe_h.unsqueeze(1)
        self.register_buffer('pe_w', pe_w)
        self.register_buffer('pe_h', pe_h)

    def forward(self, x):
        # x shape: (batch_size, channels, freq_bins, time_steps)
        time_steps = x.size(3)
        freq_bins = x.size(2)

        pe_w = self.pe_w[:time_steps, :].permute(2, 0, 1)  # (1, time_steps, embedding_dim//2)
        pe_h = self.pe_h[:freq_bins, :].permute(2, 1, 0)   # (1, embedding_dim//2, freq_bins)
        pe = torch.cat([pe_w.expand(x.size(0), -1, -1, -1), pe_h.expand(x.size(0), -1, -1, -1)], dim=1)
        x = x + pe
        return x

# 2. Encoder з CNN для спектрограм та 2D позиційними ембеддингами
class Encoder(nn.Module):
    def __init__(self, embedding_dim=256):
        super(Encoder, self).__init__()
        self.conv_layers = nn.Sequential(
            nn.Conv2d(1, 64, kernel_size=3, stride=1, padding=1),
            nn.BatchNorm2d(64),
            nn.ReLU(),
            nn.Conv2d(64, 128, kernel_size=3, stride=2, padding=1),
            nn.BatchNorm2d(128),
            nn.ReLU(),
```

```
            nn.Conv2d(128, embedding_dim, kernel_size=3, stride=2, padding=1),
            nn.BatchNorm2d(embedding_dim),
            nn.ReLU(),
        )
        # Адаптований 2D позиційний ембеддинг
        self.positional_embedding = PositionalEmbedding2D(embedding_dim)

    def forward(self, x):
        x = self.conv_layers(x)  # Розмір: (batch_size, embedding_dim, freq_bins, time_steps)
        x = self.positional_embedding(x)
        batch_size, embedding_dim, freq_bins, time_steps = x.size()
        x = x.view(batch_size, embedding_dim, -1).permute(0, 2, 1)  # Розмір: (batch_size, seq_length, embedding_dim)
        return x

# 3. Інші блоки моделі залишаються майже без змін, але треба узгодити розміри
class LSTMAttentionBlock(nn.Module):
    def __init__(self, embedding_dim=256, num_heads=4):
        super(LSTMAttentionBlock, self).__init__()
        self.lstm = nn.LSTM(embedding_dim, embedding_dim, bidirectional=True, batch_first=True)
        self.attention = nn.MultiheadAttention(embedding_dim * 2, num_heads, batch_first=True)
        self.ffn = nn.Sequential(
            nn.Linear(embedding_dim * 2, embedding_dim * 4),
            nn.GELU(),
            nn.Linear(embedding_dim * 4, embedding_dim * 2),
        )
        self.norm1 = nn.LayerNorm(embedding_dim * 2)
```

```
        self.norm2 = nn.LayerNorm(embedding_dim * 2)

    def forward(self, x):
        lstm_out, _ = self.lstm(x)
        x = x + self.norm1(lstm_out)
        attn_out, _ = self.attention(x, x, x)
        x = x + self.norm2(self.ffn(attn_out))
        return x

class TransformerDecoderAttractor(nn.Module):
    def __init__(self, embedding_dim=256, num_queries=4, num_layers=2,
num_heads=4):
        super(TransformerDecoderAttractor, self).__init__()
        self.query_embeddings = nn.Parameter(torch.randn(num_queries, embedding_dim
* 2))
        self.decoder_layers = nn.ModuleList([
            nn.TransformerDecoderLayer(embedding_dim * 2, num_heads) for _ in
range(num_layers)
        ])
        self.linear = nn.Linear(embedding_dim * 2, embedding_dim * 2)

    def forward(self, context):
        queries = self.query_embeddings.unsqueeze(0).expand(context.size(0), -1, -1)
        for layer in self.decoder_layers:
            queries = layer(queries, context)
        return self.linear(queries)

class TriplePathBlock(nn.Module):
    def __init__(self, embedding_dim=256, num_heads=4):
        super(TriplePathBlock, self).__init__()
```

```
        self.intra_chunk = LSTMAttentionBlock(embedding_dim, num_heads)
        self.inter_chunk = LSTMAttentionBlock(embedding_dim, num_heads)
        self.inter_speaker = nn.TransformerEncoderLayer(embedding_dim * 2,
num_heads)

    def forward(self, x):
        x = self.intra_chunk(x)
        x = self.inter_chunk(x)
        x = self.inter_speaker(x)
        return x

class Decoder(nn.Module):
    def __init__(self, embedding_dim=256):
        super(Decoder, self).__init__()
        self.deconv_layers = nn.Sequential(
            nn.ConvTranspose2d(embedding_dim, 128, kernel_size=3, stride=2, padding=1,
output_padding=1),
            nn.BatchNorm2d(128),
            nn.ReLU(),
            nn.ConvTranspose2d(128, 64, kernel_size=3, stride=2, padding=1,
output_padding=1),
            nn.BatchNorm2d(64),
            nn.ReLU(),
            nn.ConvTranspose2d(64, 1, kernel_size=3, stride=1, padding=1),
        )
        # Фільтр для усунення артефактів
        self.artifact_filter = nn.Sequential(
            nn.Conv2d(1, 16, kernel_size=3, stride=1, padding=1),
            nn.BatchNorm2d(16),
            nn.ReLU(),
```

```
            nn.Conv2d(16, 1, kernel_size=3, stride=1, padding=1),
            nn.Tanh()
        )
        # Гармонічна структура
        self.harmonic_restoration = nn.Sequential(
            nn.Conv2d(1, 16, kernel_size=3, stride=1, padding=1),
            nn.ReLU(),
            nn.Conv2d(16, 1, kernel_size=3, stride=1, padding=1),
            nn.Sigmoid()  # Відновлення гармонічної амплітуди
        )

    def forward(self, x, original_size):
        x = x.permute(0, 2, 1)  # Розмір: (batch_size, embedding_dim, seq_length)
        batch_size, embedding_dim, seq_length = x.size()
        freq_bins, time_steps = original_size
        x = x.view(batch_size, embedding_dim, freq_bins, time_steps)
        # Основний декодер
        x = self.deconv_layers(x)
        # Артефактна фільтрація
        x = self.artifact_filter(x)
        # Відновлення гармонічної структури
        x = self.harmonic_restoration(x)
        return x

# SepSimTDA Model
class SepSimTDA(nn.Module):
    def __init__(self, embedding_dim=256, separator_dim=256, num_heads=4,
num_layers=8):
        super(SepSimTDA, self).__init__()
        self.encoder = Encoder(embedding_dim)
```

```
        self.dual_path = LSTMAttentionBlock(separator_dim, num_heads)
        self.tda = TransformerDecoderAttractor(separator_dim, num_queries=4,
num_layers=2, num_heads=num_heads)
        self.triple_path = nn.ModuleList([TriplePathBlock(separator_dim, num_heads) for
_ in range(num_layers)])
        self.decoder = Decoder(embedding_dim)

    def forward(self, x):
        # x shape: (batch_size, 1, time)
        # Генерація спектрограми
        spectrograms, original_size = x  # x тепер містить спектрограму та її
оригінальний розмір
        encoded = self.encoder(spectrograms)  # Розмір: (batch_size, seq_length,
embedding_dim)
        dual_path_output = self.dual_path(encoded)
        attractors = self.tda(dual_path_output)
        output = dual_path_output
        for block in self.triple_path:
            output = block(output)
        decoded = self.decoder(output, original_size)
        return decoded.squeeze(1)

"""Dataset Preparation з аугментацією та мультирезолюційними спектрограмами"""
from torch.utils.data import Dataset, DataLoader
import torchaudio
import random

class SpeechSeparationDataset(Dataset):
    def __init__(self, mixtures, sources, transform=None, augmentations=None):
        """
```

```
    Dataset для задачі розділення мовлення.
    Args:
        mixtures: список шляхів до мішаних аудіо файлів.
        sources: список списків шляхів до джерел мовлення (по одному на спікера).
        transform: опціональний трансформ для попередньої обробки.
        augmentations: список аугментацій для аудіо.
    """
    self.mixtures = mixtures
    self.sources = sources
    self.transform = transform
    self.augmentations = augmentations or []
    self.sample_rate = 21000  # Наприклад, 21 кГц

def __len__(self):
    return len(self.mixtures)

def __getitem__(self, idx):
    # Завантаження мішаного аудіо
    mixture, _ = torchaudio.load(self.mixtures[idx])

    # Завантаження чистих джерел
    sources = []
    for source_path in self.sources[idx]:
        source, _ = torchaudio.load(source_path)
        sources.append(source)
    # Вирівнювання тривалості
    min_length = min(mixture.size(1), *[s.size(1) for s in sources])
    mixture = mixture[:, :min_length]
    sources = [s[:, :min_length] for s in sources]
```

```
        # Аугментація (якщо необхідно)
        if self.augmentations:
            mixture = self.apply_augmentations(mixture)
            sources = [self.apply_augmentations(s) for s in sources]
        # Генерація спектрограм або інших ознак (якщо необхідно)
        # ...
        # Перетворення в тензор
        sources = torch.stack(sources, dim=0)  # (num_speakers, 1, time)
        return mixture.squeeze(0), sources.squeeze(1)  # (time), (num_speakers, time)

    def apply_augmentations(self, audio):
        for aug in self.augmentations:
            audio = aug(audio)
        return audio



# Аугментації
def add_noise(audio, noise_level=0.005):
    noise = torch.randn_like(audio) * noise_level
    return audio + noise

def change_volume(audio, gain_db):
    return audio * (10 ** (gain_db / 20))

def time_stretch(audio, rate):
    return torchaudio.transforms.Resample(orig_freq=16000, new_freq=int(16000 *
rate))(audio)

augmentations = [
    lambda x: add_noise(x, noise_level=random.uniform(0.001, 0.01)),
```

```
    lambda x: change_volume(x, gain_db=random.uniform(-6, 6)),
    lambda x: time_stretch(x, rate=random.uniform(0.8, 1.2)),
]

# Ініціалізація датасету
import pandas as pd

# Завантаження метаданих
metadata_df = pd.read_csv('data/metadata.csv')
# Отримання списку шляхів до мішаних файлів
mixture_paths = metadata_df['mixture_path'].tolist()
# Обробка рядків із шляхами до джерел
source_paths = []
for idx, row in metadata_df.iterrows():
    # Рядок із списком шляхів до джерел
    sources_str = row['source_paths']
    # Перетворення рядка в список (якщо збережено як рядок списку)
    sources_list = eval(sources_str)
    source_paths.append(sources_list)

# Ініціалізація датасету з оновленими шляхами
dataset = SpeechSeparationDataset(mixture_paths, source_paths,
augmentations=augmentations)

# Ініціалізація DataLoader
dataloader = DataLoader(dataset, batch_size=4, shuffle=True, num_workers=2)

# Перевірка наявності файлів
for path in mixture_paths:
    if not os.path.isfile(path):
```

```
        print(f'Файл не знайдено: {path}')

for sources in source_paths:
    for path in sources:
        if not os.path.isfile(path):
            print(f'Файл не знайдено: {path}')

"""SI-SDR Loss Function"""
def si_sdr_loss(estimated, target):
    """Calculate SI-SDR loss."""
    eps = 1e-8
    target_energy = torch.sum(target ** 2, dim=-1, keepdim=True)
    projection = torch.sum(target * estimated, dim=-1, keepdim=True) / (target_energy +
eps) * target
    noise = estimated - projection
    si_sdr = 10 * torch.log10((torch.sum(projection ** 2, dim=-1) + eps) /
(torch.sum(noise ** 2, dim=-1) + eps))
    return -torch.mean(si_sdr)

"""Training Script"""
from torch.optim import AdamW

# Initialize the model
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = SepTDA().to(device)

# Optimizer and scheduler
optimizer = AdamW(model.parameters(), lr=4e-4)
scheduler = torch.optim.lr_scheduler.ReduceLROnPlateau(optimizer, factor=0.5,
patience=5)
```

```
# Training Loop
num_epochs = 20
for epoch in range(num_epochs):
    model.train()
    total_loss = 0.0

    for batch_idx, (mixture, speakers) in enumerate(dataloader):
        mixture = mixture.to(device)  # Input mixture
        speakers = speakers.to(device)  # Ground truth speakers
        optimizer.zero_grad()
        # Forward pass
        estimated_sources = model(mixture.unsqueeze(1))
        # Calculate loss
        loss = sum(si_sdr_loss(estimated_sources[:, i, :], speakers[:, i, :]) for i in
range(speakers.size(1)))
        loss.backward()

        optimizer.step()
        total_loss += loss.item()

        # Logging
        if batch_idx % 10 == 0:
            print(f"Epoch [{epoch + 1}/{num_epochs}], Step
[{batch_idx}/{len(dataloader)}], Loss: {loss.item():.4f}")

    # Adjust learning rate
    scheduler.step(total_loss)
    print(f"Epoch {epoch + 1}, Average Loss: {total_loss / len(dataloader):.4f}")
# Save the trained model
torch.save(model.state_dict(), "sepsimtda_model.pth")
```