Міністерство освіти і науки України

Харківський національний університет імені В.Н. Каразіна

Факультет комп'ютерних наук

Спеціальність 125 «Кібербезпека»

Освітня програма «Безпека інформаційних та комунікаційних систем»

«Допущено до захисту»

Зав. кафедрою БІСТ

Рассомахін С.Г. _____

« 02 » 12 2022 р.

Пояснювальна записка

до кваліфікаційної роботи магістра

на тему: «Математична модель і програмна імплементація біометричної системи автоматичного розпізнавання голосу»

оцінка « » Керівник проф. Рассомахін С.Г.

Голова ЕК Рецензент проф. Краснобаєв В.А.

Доценко С.І. _____ Виконавець: студентка групи КБ-61

Дегтяр І.О.

Харків – 2022

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи магістра, 60 сторінок, 48 рисунків, 2 таблиці, 27 джерел, 2 додатки.

Об’єкт дослідження – процес автоматичного розпізнавання по голосу.

Предмет дослідження – біометрична ідентифікація та автентифікація особи за голосом.

Мета цієї роботи полягала у розробці математичної моделі та програмної імплементації біометричної системи автоматичного розпізнавання по голосу.

Основними методами досліджень були теорія цифрової обробки сигналів, теорія алгоритмів, автоматична обробка розмовної мови.

В ході роботи були проаналізовані основні види систем розпізнавання, методи зменшення рівня шуму, алгоритм детектування активного мовлення, мел-частотні кепстральні коефіцієнти та модель суміші Гауса. Розроблена математична модель та програмна імплементація біометричної системи автоматичного розпізнавання по голосу за допомогою мови програмування Python. За допомогою сервісу Descript було згенеровано мовлення на основі попередньо записаного аудіофрагменту та перевірено роботу алгоритму за умови використання підробки. Проведений аналіз результатів та наведені рекомендації щодо значень параметрів системи, які дозволять отримати найвищі можливі показники ефективності.

Результати роботи можуть використовуватись у розробці майбутніх систем автоматичного розпізнавання по голосу або бути частиною складних систем. Для покращення ефективності розробленої системи запропоновано використання додаткового випадкового паролю та часового ліміту. В майбутніх дослідженнях з цієї теми потрібно передбачити використання більшої кількості учасників для розуміння універсальності системи.

Ключові слова: БІОМЕТРІЯ, АВТЕНТИФІКАЦІЯ, ІДЕНТИФІКАЦІЯ, МЕЛ-ЧАСТОТНІ КЕПСТРАЛЬНІ КОЕФІЦІЄНТИ, СПЕКТР, КЕПСТР, АВТОМАТИЧНЕ РОЗПІЗНАВАННЯ, МОДЕЛЬ СУМІШІ ГАУСА.

ABSTRACT

Explanatory note to the qualification work of the master's degree, 60 pages, 48 figures, 2 tables, 27 references, 2 applications.

Object of research – the process of automatic recognition by voice.

Subject of research – biometric identification and authentication of a person by voice.

The purpose of this work was to develop a mathematical model and program implementation of a biometric system for automatic voice recognition.

The main research methods are: theory of digital signal processing, theory of algorithms, automatic speech processing.

In the course of the work, the main types of recognition systems, noise reduction methods, voice active detection algorithm, mel-frequency cepstral coefficients and Gaussian mixture models were analyzed. A mathematical model and program implementation of a biometric system of automatic voice recognition was developed using Python programming language. With the help of the Descript service, the speech was generated based on a pre-recorded audio fragment, and the algorithm was tested using a fake. The results are analyzed, and recommendations are given on the values of the system parameters that will allow for obtaining the highest possible performance indicators.

The work results can be used in developing future automatic voice recognition systems or be part of complex systems. To improve the efficiency of the developed system, it is proposed to use an additional random password and time limit. Future research on this topic should include the use of a more significant number of participants to understand the system's universality.

Keywords: BIOMETRICS, AUTHENTICATION, IDENTIFICATION, MEL-FREQUENCY CEPSTRAL COEFFICIENTS, SPECTRUM, CEPSTRUM, AUTOMATIC RECOGNITION, GAUSSIAN MIXTURE MODEL.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

АЧХ	–	амплітудно-частотна характеристика
БД	–	база даних
ДКП	–	дискретне косинусне перетворення
ШПФ	–	швидке перетворення Фур’є
EM	–	expectation maximization
GMM	–	Gaussian mixture model
LPC	–	linear predictive coefficients
LPCC	–	linear predictive cepstral coefficients
MFCC	–	mel-frequency cepstral coefficients
MLE	–	maximum likelihood estimation
STFT	–	short-time Fourier transform
VAD	–	voice active detection

ВСТУП

Особливе місце у розвитку біометричних технологій посідає розробка рішень з голосової біометрії. Слід зазначити, що донедавна системи біометричної ідентифікації та автентифікації за голосом мали значно гірші показники порівняно з іншими системами. Однак за останні кілька років у цій галузі було досягнуто значних успіхів.

Недоліком більшості методів біометрії є незмінність використовуваної ознаки. Це є перешкодою для використання подібних методів у випадках, коли потрібна більша надійність ідентифікації. У свою чергу верифікація за голосом може використовуватися в темряві, на відстані (наприклад, по телефонному каналу). При цьому така система дає можливість отримати більше вхідних даних за рахунок довших мовних повідомлень.

Унікальність голосу людини зумовлена безліччю фізіологічних особливостей: будова голосових зв'язок, трахеї, манера вимови, розташування зубів. Комбінація цих особливостей індивідуальна, як і відбитки пальців. Проте варто зауважити, що на практиці жодна із систем біометричної ідентифікації, зокрема й голосова, не може гарантувати 100-відсоткової ідентифікації особистості.

Методи розпізнавання диктора можуть як залежати від тексту, так і приймати довільне мовлення. Текстово-залежна система розпізнає особу на основі деяких слів або фраз, які раніше були записані та збережені у базі даних. Текстово-незалежна система розпізнає диктора без наявності певного слова у базі даних. Система вилучає унікальні характеристики голосу, що робить можливим розпізнавання без вимови конкретного слова. При використанні текстово-залежного методу складність процесу зменшується, оскільки наявна база даних для порівняння голосів.

У сучасних системах голосової ідентифікації для підвищення надійності часто використовується система розпізнавання з додатковими запитами, що

вимагає, наприклад, вимовляння парольної фрази, яка щоразу генерується випадковим чином. Використання індивідуальних ознак і збіг згенерованої та розпізнаної парольних фраз підвищують надійність. Таку систему можна назвати окремим випадком текстово-залежного розпізнавання. При цьому текстово-незалежна ідентифікація передбачає використання тільки індивідуальних ознак.

Розпізнавання спонтанного довільного мовлення залишається важкодосяжною метою. Ця задача ускладняється тим, що мовний сигнал спотворюється багатьма джерелами: шум на задньому фоні, характеристики мікрофону, особливості голосового тракту та відмінності у вимові. Для вирішення цієї проблеми ефективна система повинна намагатись використовувати усі доступні джерела знань: акустику, фонетику, лексичну, синтаксичну та семантичну структуру мови.

1 ОСНОВНІ ТЕОРЕТИЧНІ ВІДОМОСТІ З ТЕМИ ДОСЛІДЖЕННЯ

1.1 Системи розпізнавання мовлення. Основні різновиди

Розпізнавання мовлення – це процес автоматичного визначення особи, що говорить, на основі індивідуальної інформації, що міститься у мовленні. Така техніка робить можливим використання голосу людини для контролю доступу до таких сервісів як, наприклад, банківські операції по телефону, голосова пошта, доступ до бази даних і так далі [1].

Процес розпізнавання мовлення поділяється на дві категорії: ідентифікація та автентифікація мовця. Головна відмінність між ними полягає в тому, що результат автентифікації є бінарним, тобто визначається чи є людина, що говорить, тою, за яку себе видає. Ідентифікація мовця полягає у порівнянні голосу людини, що говорить, з базою даних еталонних моделей з метою ідентифікувати особу.

Процес розпізнавання може бути поділений на дві фази: навчання та верифікація. Протягом навчання будуються еталонні моделі для кожного мовця. Під час верифікації невідома модель мовця порівнюється з наявними моделями для визначення особи мовця. Нижче (Рисунок 1.1 та Рисунок 1.2) зображено схеми виконання цих процесів.

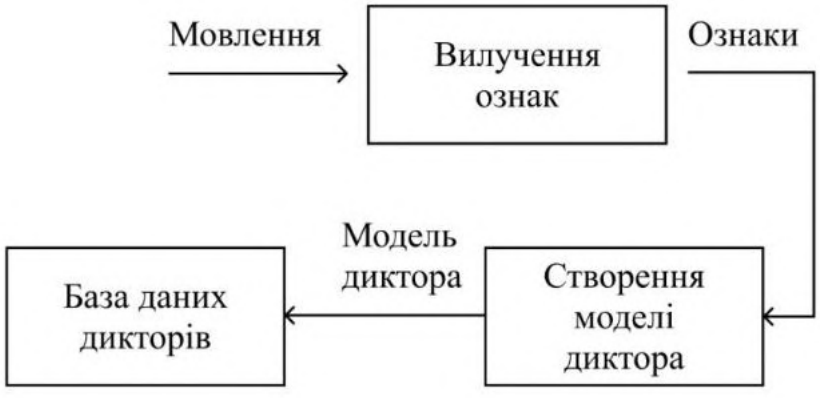

Рисунок 1.1 – Етап навчання системи

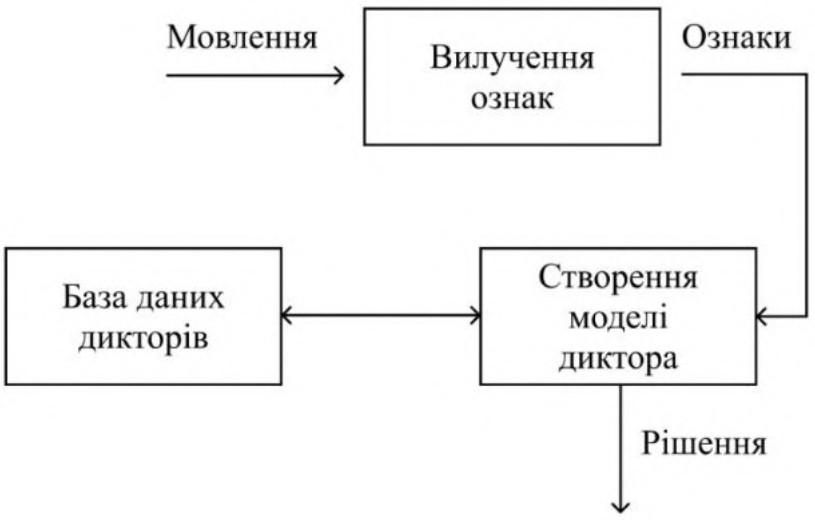

Рисунок 1.2 – Етап верифікації

Зазвичай системи розпізнавання поділяються на два типи: текстово-залежні та текстово-незалежні. Перший тип, як видно із назви, залежить від тексту, що використовується для навчання системи та процесу верифікації. Другий тип, відповідно, може приймати довільний текст. Іноді виділяють третій тип систем верифікації – з додатковими запитами. Такі системи дозволяють вирішити проблему, пов’язану з можливістю використання записаного раніше голосу для несанкціонованого доступу, шляхом необхідності промови певної послідовності слів, що генерується випадковим чином під час перевірки [2].

В текстово-залежних системах еталонна модель мовця будується на основі встановленого паролю. Оскільки пароль є попередньо відомим системі, текстово-залежна модель може засновуватись як на характеристиках мовця, так і на паролі. Приклад роботи зображено нижче (Рисунок 1.3).

Система:	Будь-ласка, скажіть номер Вашого рахунку
Користувач:	234 557
Система:	Дякую.

Рисунок 1.3 – Приклад роботи текстово-залежної системи

Окремим випадком текстово-залежної перевірки є система з додатковими запитами, яка просить мовця повторити серію випадково вибраних рядків цифр,

послідовностей слів або фраз. Як показано нижче (Рисунок 1.4), перевірка займає більше часу.

Етап 1:	Система:	Будь ласка, введіть номер Вашого рахунку за допомогою сенсорної клавіатури
	Користувач:	234 557
Етап 2:	Система:	Будь-ласка, скажіть 46-89
	Користувач:	46-89
	Система:	Будь-ласка, скажіть 99-07
	Користувач:	99-07
	Система:	Дякую.

Рисунок 1.4 – Приклад роботи системи з додатковими запитами

Така перевірка добре підходить для систем з високим рівнем ризику, а також є корисною, коли є занепокоєння щодо використання підроблених записів, оскільки питання, які задаються користувачеві, вибираються випадковим чином, що ускладнює генерацію та відтворення запису.

Незалежна від тексту перевірка приймає будь-які голосові дані, що дає змогу створювати приховані програми перевірки. Здатність текстово-незалежної технології працювати непомітно робить її привабливою для випадків, пов’язаних із клієнтами. Наприклад, нижче (Рисунок 1.5), клієнт банку має бажання виконати транзакцію. Він оголошує своє бажання, та при цьому система перевіряє чи абонент є авторизованим клієнтом. Проте текстово-незалежна технологія вимагає більше зразків мовлення та є більш чутливою до якості вхідного сигналу.

Агент:	Будь-ласка, скажіть мету Вашого візиту
Користувач:	Я хочу відправити гроші з мого особистого рахунку
Агент:	Дякую
	Відбувається верифікація системою особи, що говорить
Агент:	Дякую за Ваше очікування. Я оброблюю Ваш запит

Рисунок 1.5 – Приклад роботи текстово-незалежної системи

1.2 Вибір ознак для вилучення

Мовний сигнал містить безліч ознак, частина яких не є важливою для визначення мовця. Ідеальні ознаки повинні бути стійкими до шуму або спотворень та імітації, незалежними від стану здоров’я мовця, часто та природно зустрічатись у мовленні [3].

Є різні шляхи категоризації ознак. Наприклад, їх можна розділити на:

- Короткострокові спектральні ознаки;
- Ознаки джерела голосу;
- Спектрально-часові;
- Просодичні;
- Високорівневі.

Короткострокові спектральні ознаки обчислюються над короткими фреймами тривалістю 20-30 мс. У свою чергу, ознаки джерела голосу характеризують його джерело. Просодичні та спектрально-часові ознаки охоплюють десятки або сотні мілісекунд, включаючи, наприклад, інтонацію та ритм. Високорівневі ознаки намагаються фіксувати характеристики мовця на розмовному рівні, наприклад, персональний лексикон. Короткий опис ознак наведено нижче (Рисунок 1.6).

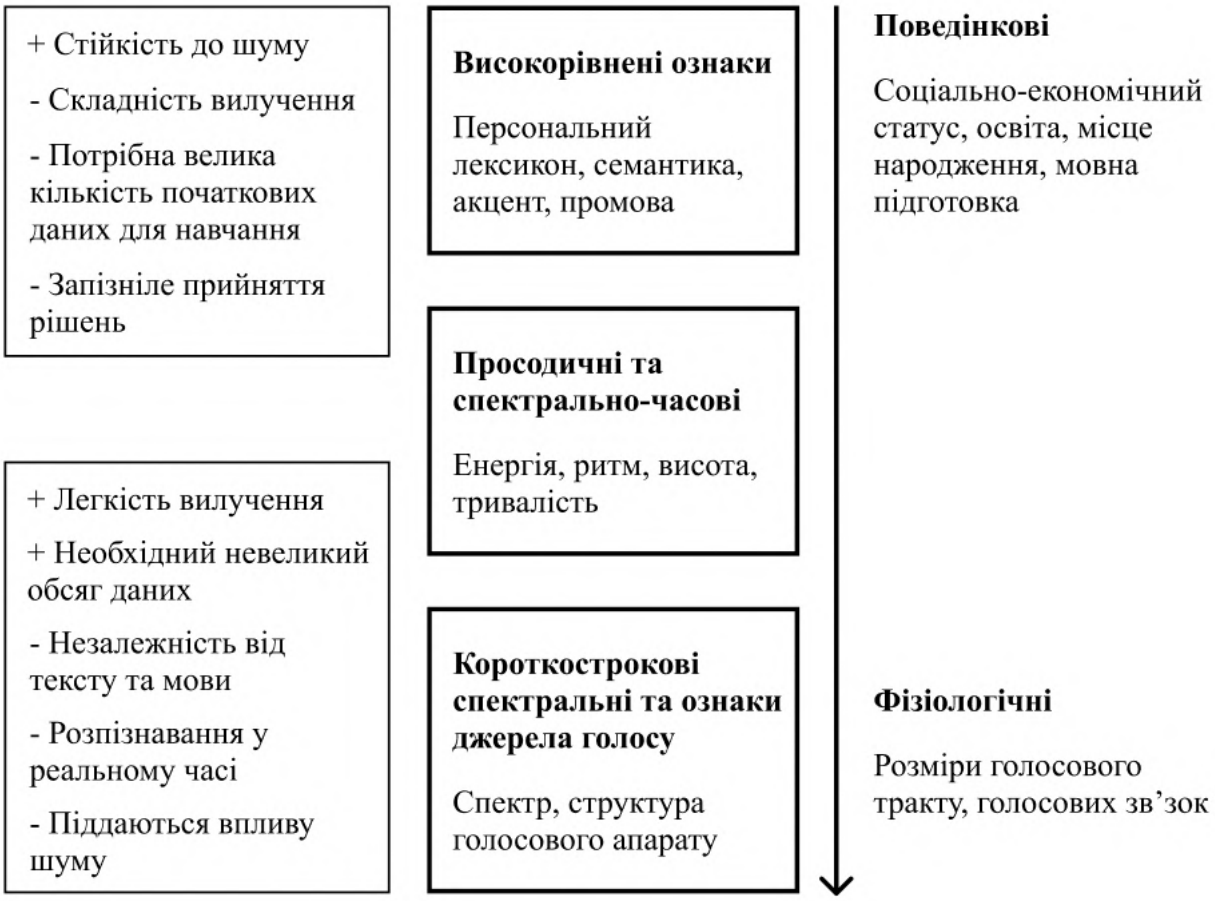

Рисунок 1.6 – Опис ознак

Вибір ознак залежить від випадку застосування, обчислювальних ресурсів, обсягу доступних вхідних даних. Найчастіше використовуються короткострокові спектральні ознаки через їх легкість обчислення та високий рівень продуктивності. Популярними ознаками є MFCC (мел-частотні кепстральні коефіцієнти). Вони були представлені у 1980-х роках для розпізнавання мовлення, а потім стали використовуватись для розпізнавання мовця.

1.3 Зменшення рівня шуму

Зазвичай, для запису звуку використовують непрофесійні засоби. Також місце запису може бути не ідеально ізольованим. Це може спричинити появу шумів у записі, які можуть завадити правильній роботі системи розпізнавання по голосу.

У цій роботі було проаналізовано два алгоритми зниження рівня шуму, які базуються на методі спектрального стробування. Вони засновані на обчисленні спектрограми сигналу (і додатково шумового сигналу) та оцінці порогу шуму для кожної смуги частот цього сигналу/шуму.

1.3.1 Метод зменшення рівня стаціонарного шуму

Алгоритми зменшення рівня стаціонарного шуму оперують шумом як стаціонарними сигналами, наприклад, постійним гудінням від розташованого поруч електронного пристрою в лабораторних умовах або шумом комах у польових умовах.

Одним із підходів до зменшення рівня стаціонарного шуму є спектральне стробування – алгоритм спектрального віднімання. Загальна ідея полягає в тому, що для кожної частотної складової сигналу будь-яка часово-частотна складова нижче порогового значення відкидається як шум. Спектральне стробування обчислює середнє та стандартне відхилення кожного частотного каналу віконного перетворення Фур'є (STFT) сигналу (наприклад, спектрограми). Потім встановлюється поріг для кожної частотної складової на деякому рівні вище середнього значення (наприклад, три стандартних відхилення). Цей поріг визначає, чи вважається частотно-часовий компонент спектрограми мовленням або шумом. Потім спектрограма маскується на основі цього порогу та інвертується за допомогою зворотного віконного перетворення Фур’є назад у часову область.

Цей алгоритм складається з таких кроків:

1) Розраховується спектрограма над аудіозаписом з шумом;
2) На основі цієї спектрограми обчислюється статистика по частоті;
3) На основі статистики шуму та бажаної чутливості алгоритму розраховується поріг;
4) Знову розраховується спектрограма аудіозапису;
5) Шляхом порівняння спектрограми сигналу з порогом визначається маска;
6) Маска згладжується фільтром по частоті і часу;
7) Маска накладається на спектрограму сигналу та інвертується назад у часову область.

Основною особливістю алгоритму є те, що визначений поріг шуму зберігається на одному рівні для всього сигналу.

1.3.2 Метод зменшення рівня нестаціонарного шуму

Алгоритми зниження рівня нестаціонарного шуму націлені на фоновий шум, який є нестаціонарним і може коливатися в часі, наприклад, літак, що пролітає поруч. Ці алгоритми є розширенням алгоритмів зменшення рівня стаціонарного шуму – вони оновлюють значення порогу шуму з плином часу.

Один з підходів до визначення межі між сигналом і нестаціонарним шумом полягає у визначенні часового інтервалу, на якому діє сигнал, і визначенні всього, що знаходиться за межами цього часового інтервалу, як шум. Наприклад, крик птаха може тривати кілька сотень мілісекунд, тому можна встановити поріг шуму на основі припущення, що події, які відбуваються на більших часових шкалах, є шумом. Розглянутий алгоритм базується на методі під назвою Per-Channel Energy Normalization, який використовується у біоакустиці [4].

Цей алгоритм складається з таких кроків:

1) Розраховується спектрограма над аудіозаписом з шумом;
2) Згладжена в часі версія спектрограми обчислюється за допомогою
3) фільтра з нескінченною імпульсною характеристикою;
4) На основі цієї згладженої в часі спектрограми обчислюється маска;
5) Маска згладжується фільтром по частоті і часу;
6) Маска накладається на спектрограму сигналу та інвертується назад у часову область [5].

1.4 Детектування активного мовлення. Алгоритм WebRTC VAD

Основна функція алгоритму VAD (Voice Activity Detection) полягає в тому, щоб виділити деякі характеристики або величини із вхідного сигналу та порівняти ці значення з пороговими [6].

VAD система, зазвичай, складається з двох частин: вилучення ознак та визначення чи є сигнал мовленням або шумом. Методи вилучення ознак можна розділити на 5 категорій:

- Енергетичні

Енергетичний критерій полягає у виявленні сили сигналу та припускає, що енергія мовлення більша, ніж енергія фонового шуму. Таким чином, коли значення

енергії є більшим, ніж певний вказаний поріг, можна вважати, шо це є мовлення. Проте якщо енергії шуму та мовлення рівні, система не зможе їх відрізнити.

- Частотні

Шляхом перетворення сигналу часової області на сигнал частотної області за допомогою віконного перетворення Фур'є довгострокова огинаюча деяких частотних діапазонів може розрізняти мову та шум навіть при співвідношенні сигнал/шум близького до 0ДБ.

- Кепстральні

Для VAD пікова енергія кепстру визначає висоту тону мовного сигналу. Також у якості ознак використовуються MFCC.

- На основі гармонік

Особливістю мовлення є включення основної частоти та її гармонійних частот, навіть при сильному шумі. Основна частота може були знайдена за допомого методу автокореляції.

- Довгострокові

Мовлення є нестаціонарним сигналом. Нормальний темп мовлення зазвичай становить від 10 до 15 фонем в секунду, при цьому спектральний розподіл фонем різний, що призводить до змін статистичних характеристик мовлення у часі. З іншого боку, більшість шумів є повсякденними та стаціонарними.

При цьому ознаки повинні мати наступні властивості:

- Розрізнювальна здатність (мається на увазі здатність розділяти фрейми, які містять як мовлення, так і шум, та ті, що містять лише шум);
- Завадостійкість (фоновий шум може призвести до спотворення мовлення, що вплине на розрізнювальну здатність).

В алгоритмі WebRTC VAD використовується ідея кластеризації. Є тільки два класи: мовлення та шум. Обчислюється ймовірність того, чи є фрейм сигналу мовленням чи шумом. Відповідно до ймовірності, відбувається кластеризація [7].

1) Частота дискретизації встановлюється як 8кГц;

2) Мовний сигнал ділиться на фрейми довжиною 10мс, при цьому немає перекриття між ними;

3) Фрейм сигналу розкладається на шість діапазонів частот: 80~250, 250~500, 500~1000, 1000~2000, 2000~3000, 3000~4000;

4) Обчислюється значення енергії для кожного діапазону. Кожне із цих значень логарифмується;

5) У кожній смузі частот логарифм енергії підпорядковується певному розподілу ймовірностей. На цьому етапі за допомогою моделі суміші Гауса обчислюється зважений логарифм ймовірності. Якщо хоч одне із шести отриманих значень перевищує задані порогові величини, робиться висновок про те, що фрейм є фрагментом мовлення;

6) На виході маємо сигнал, розбитий на фрейми, кожен з яких помічений як шум або мовлення.

1.5 MFCC – мел-частотні кепстральні коефіцієнти

Першим кроком у будь-якій системі автоматичного розпізнавання мовлення є виділення ознак, тобто визначення компонентів аудіосигналу, які є корисними для ідентифікації, і відкидання всього іншого, що не має практичної цінності, наприклад, фонового шуму.

До 1980-х років для автоматичного розпізнавання мовлення частіше використовувались лінійні коефіцієнти передбачення (LPC) та лінійні кепстральні коефіцієнти передбачення (LPCC). Принцип лінійного передбачення полягає в тому, що ділянка мовного сигналу може бути апроксимована за допомогою лінійної комбінації попередніх ділянок.

Проте домінуючим алгоритмом обробки мовних сигналів є знаходження кепстральних коефіцієнтів. Кепстром називається спектр логарифму спектра часової хвилі та визначається за формулою 1.1.

$$c[n] = F^{-1}\{\log|F\{x(n)\}|\} \tag{1.1}$$

де F та F^{-1} – пряме та зворотне дискретне перетворення Фур'є.

Першим етапом мовний сигнал проходить передобробку фільтром, який підсилює високі частоти спектру. Звуковий сигнал постійно змінюється, тому для спрощення припускаємо, що на коротких часових інтервалах звуковий сигнал мало змінюється. Таким чином, сигнал розбивається на фрейми тривалістю 10-40 мс з

перекриттям. Для послаблення спотворень сигналу застосовується віконна функція. За допомогою перетворення Фур'є для кожного вікна знаходиться спектр, який множиться зі спектром прийнятого набору фільтрів (гребінка фільтрів) для отримання середнього значення в певній смузі частот [8]. Потім обчислюється логарифм від отриманої обгортки спектру. Останнім кроком є застосування дискретного косинусного перетворення.

Найбільше застосування отримали мел-частотні кепстральні коефіцієнти (MFCC). Як було сказано раніше, звуки, які генеруються людиною, фільтруються формою голосового тракту, включаючи зуби, язик та інше. Форма голосового тракту проявляється в огинаючій короткострокового спектру потужності, і завданням MFCC є точне представлення цієї огинаючої. Як видно нижче (Рисунок 1.7), процес отримання MEL-коефіцієнтів складається з 7 кроків.

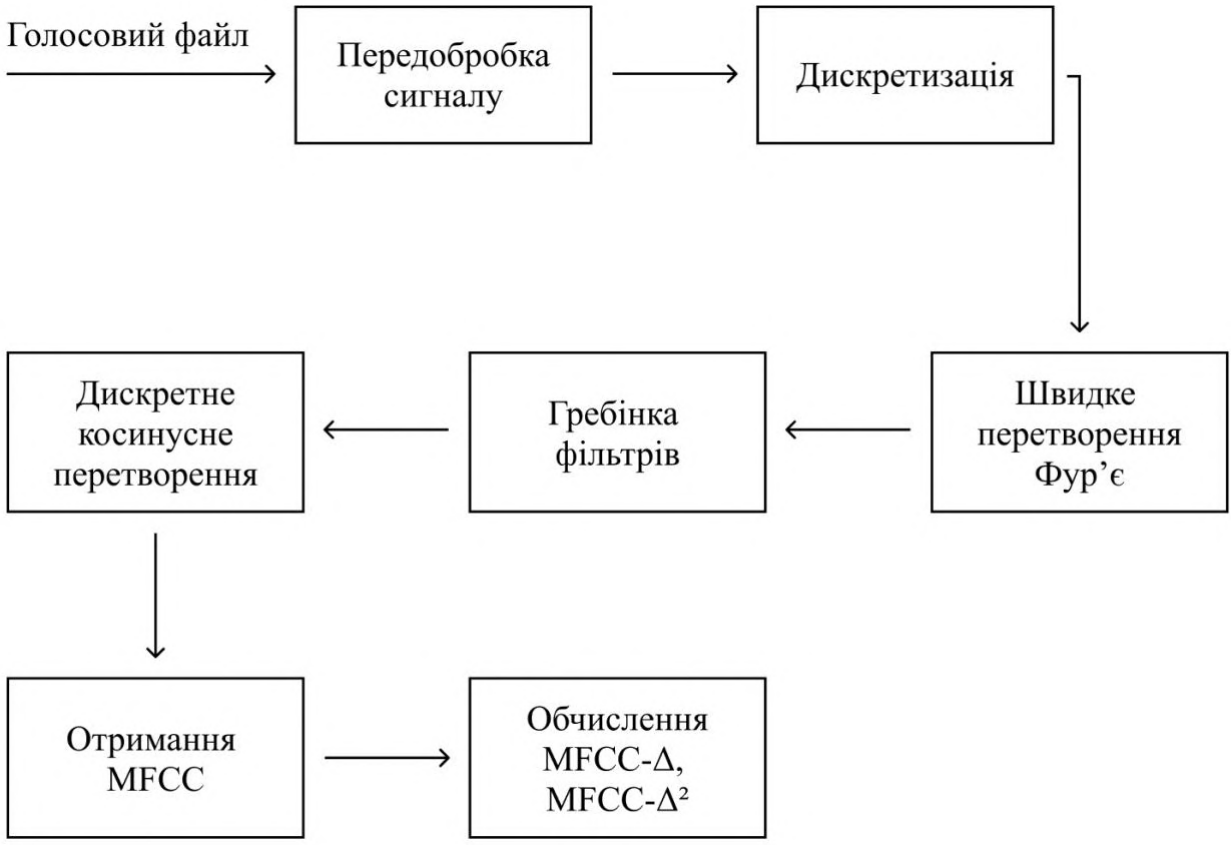

Рисунок 1.7 – Процес отримання MEL-коефіцієнтів

Опишемо кожен крок більш детально.

1) Передобробка сигналу здійснюється шляхом пропускання через фільтр, який підкреслює його вищу частоту (формула 1.2). Енергія сигналу збільшується на вищій частоті. Метою цього процесу є зменшення шумів у вхідному сигналі.

$$Y[n] = X[n] - a * X[n-1] \quad (1.2)$$

1) Процес дискретизації полягає в розбитті мовного сигналу на короткі фрейми. Мовний сигнал є нестабільним, проте звукові відрізки такої довжини можуть вважатись стабільними. Отже, метою цього процесу є отримання відносно стабільних частотних характеристик. Час, протягом якого передбачена обробка сигналу, називається вікном, а дані, отримані у вікні – фреймом.

Зазвичай, розрахунки проводяться над вікнами довжиною 25 мс з покроковим зміщенням на 10 мс [9,10]. Зміщення часового вікна кожні 10 мс дає змогу відстежувати часові характеристики окремих звуків мовлення, і вікна довжиною 25 мс зазвичай достатньо, щоб забезпечити потрібну спектральну роздільну здатність цих звуків. Мета перекриття фреймів полягає в тому, щоб кожен звук мовлення вхідної послідовності знаходився у центрі одного з фреймів. Таким чином, два послідовні фрейми мають область перекриття, як зображено нижче (Рисунок 1.8).

Рисунок 1.8 – Область перекриття двох послідовних фреймів

До кожного фрейму застосовується віконна функція для посилення гармонік і згладжування країв перед застосуванням ШПФ (швидке перетворення Фур’є) до сигналу. Зазвичай використовуються вікна Хеммінга (формула 1.3) [11].

$$w(n) = 0.53836 - 0.46164\left(\cos\left(\frac{2n\pi}{N-1}\right)\right) \quad (1.3)$$

2) Швидке перетворення Фур'є використовується для вилучення та стиснення деяких ознак мовного сигналу без втрати важливої інформації для подальшого полегшення обробки мовлення. За допомогою цього перетворення сигнал представляється у частотній області, при цьому вся інформація зберігається, змінюється лише представлення.

$$X[k] = \sum_{n=0}^{N-1} x[n] * \exp\left(\frac{-2*\pi*i*k*n}{N}\right), \ \ 0 \le k \le N \quad (1.4)$$

3) Мел-спектр обчислюється шляхом пропускання перетвореного Фур'є сигналу через набір смугових фільтрів, які мають назву гребінка фільтрів. Мел - це одиниця виміру, заснована на частоті, що сприймається людським вухом.

Для переведення значення частоти сигналу з герц в мел-шкалу потрібно скористатись формулою 1.5:

$$M = 2595\log_{10}\left(1 + \frac{f}{700}\right) \quad (1.5)$$

При цьому зворотне перетворення має такий вигляд (формула 1.6):

$$F = 700 * \exp\left(\frac{M}{1127} - 1\right) \quad (1.6)$$

4) Для обчислення MFCC гребінка фільтрів зазвичай реалізовується в частотній області (Рисунок 1.9). Центральні частоти фільтрів зазвичай рівномірно розташовані на осі частот. Проте, з метою зімітувати сприйняття людського вуха реалізується викривлена вісь. Найчастіше використовуваною формою фільтрів є трикутна. Зазвичай, гребінка фільтрів складається з 20-40 (стандарт – 26) трикутних фільтрів. Мел-спектр спектра амплітуд обчислюється шляхом множення амплітудного спектру на кожен з трикутних вагових мел-фільтрів. Гребінка фільтрів будується за формулою 1.7:

$$H_m(k) = \begin{cases} 0, k < f(m-1) \\ \frac{k - f(m-1)}{f(m) - f(m-1)}, \ \ f(m-1) \le k \le f(m) \\ \frac{f(m+1) - k}{f(m+1) - f(m)}, \ \ f(m) \le k \le f(m+1) \\ 0, \ \ k > f(m+1) \end{cases} \quad (1.7)$$

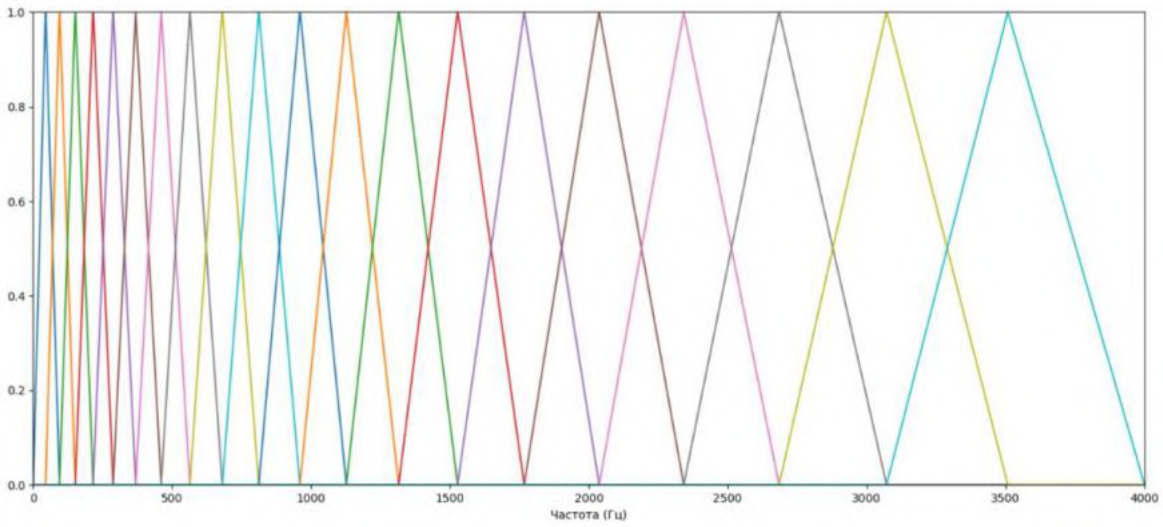

Рисунок 1.9 – Гребінка мел-фільтрів

Продемонструємо процес отримання гребінки фільтрів. Першим кроком потрібно визначити найнижчу та найвище частоти. Нехай, це будуть 300 Гц та 8000 Гц відповідно. Використавши формулу 1.5 отримаємо значення 401,25 Мел та 2834,99 Мел.

Для цього прикладу будемо використовувати 10 фільтрів, для яких потрібно 12 точок, лінійно розташованих між значеннями 401,25 та 2834,99. Це матиме такий вигляд:

$$M(i) = 401.25, 622{,}5, 843.75, 1065.00, 1286.25, 1507.50, 1728.74, 1949.99,$$
$$2171.24, 2392.49, 2613.74, 2834.99$$

Конвертуємо ці значення в герци:

$$F(i) = 300, 517.33, 781.90, 1101.97, 1496.04, 1973.32, 2554.33, 3261.62,$$
$$4122.64, 5170.76, 6446.70, 8000$$

Тепер нам потрібно накласти отриману шкалу на спектр нашого фрейму. Знаючи розмір ШПФ та частоту дискретизації отримаємо значення:

$$f_i = floor((FrameSize + 1) * M(i)/SampleRate) \quad (1.8)$$

У нашому випадку отримаємо:

$$f_i = 9, 16, 25, 35, 47, 63, 81, 104, 132, 165, 205, 256$$

Остаточна гребінка фільтрів закінчується на значенні 256, що відповідає 8 кГц з розміром ШПФ 512 точок. Перший фільтр почнеться в початковій точці, досягне свого піку в другій точці, а потім повернеться до нуля в 3-й точці. Другий

банк фільтрів почнеться в 2-й точці, досягне максимуму в 3-й, потім буде нульовим в 4-й і т.д. Формула для їх розрахунку виглядає наступним чином:

5) Дискретне косинусне перетворення (ДКП) застосовується до перетворених мел-частотних коефіцієнтів. Сигнал частотної області перетворюється в сигнал часової області, а ознаки визначаються як мел-частотні кепстральні коефіцієнти. Формула ДКП має такий вигляд:

$$C(n) = \sum_{m=0}^{M-1} S(m) * \cos\left(\frac{\pi n*(m+0.5)}{M}\right) \quad (1.9)$$

де M – кількість мел-коефіцієнтів, $0 \leq n < M$.

6) Оскільки, крім самого спектра, індивідуальність голосу формується швидкістю та прискореннями, MFCC комбінують з першою та другою похідними (MFCC-Δ, MFCC-Δ^2).

MFCC-Δ та MFCC-Δ^2 відомі як диференціальні коефіцієнти та коефіцієнти прискорення відповідно. MFCC описують лише спектральну огинаючу потужності одного фрейму. Проте мовлення також має корисну інформацію в динаміці. Тому обчислення MFCC-Δ та MFCC-Δ^2 і додавання їх до вихідного вектора ознак досить сильно підвищує продуктивність системи розпізнавання (якщо ми маємо 12 MFCC коефіцієнтів, то отримаємо додатково 24 коефіцієнти, які в сукупності дадуть вектор ознак довжиною 36).

Для обчислення дельта-коефіцієнтів використовується формула:

$$d_t = \frac{\sum_{n=1}^{N}(c_{t+n} - c_{t-n})}{2\sum_{n=1}^{N} n^2} \quad (1.10)$$

де d_t – дельта-коефіцієнт з фрейму t, обчисленого для статичних коефіцієнтів c_{t+n} та c_{t-n}. Зазвичай, значення N дорівнює 2. MFCC-Δ^2 обчислюються аналогічним чином, проте на основі дельт, а не статичних коефіцієнтів.

1.6 GMM – модель суміші Гауса

Модель суміші Гауса - це ймовірнісна модель м’якої кластеризації, яка дозволяє описати приналежність точок до множини кластерів за допомогою суміші гауссових щільностей. GMM зазвичай використовується як параметрична модель розподілу ймовірностей безперервних вимірювань або ознак у біометричній

системі, таких як спектральні ознаки, пов’язані з голосовим трактом, у системі розпізнавання мовців [12].

Модель суміші Гауса задається рівнянням:

$$P(x|\Theta) = \sum_{j=1}^{K} w_j \mathcal{N}(x|\mu_j, \Sigma_j) \quad (1.11)$$

де x – D-вимірний вектор даних, w_j – ваги суміші, Σ_j – матриця коваріації суміші, μ_j – математичне сподівання суміші.

При цьому:

$$\mathcal{N}(x|\mu_j, \Sigma_j) = \frac{1}{\sqrt{(2\pi)^K |\Sigma_j|}} \exp\left\{-\frac{1}{2}(x-\mu_j)^T \Sigma_j^{-1}(x-\mu_j)\right\} \quad (1.12)$$

Ваги суміші задовольняють умові $\sum_{j=1}^{K} w_j = 1$.

Параметри повної моделі суміші Гауса записуються у такому вигляді:

$$\Theta = \{w_j, \mu_j, \Sigma_j\}, j = 1, \dots, K.$$

Параметри моделі суміші Гауса проілюстровані графічно нижче (Рисунок 1.10).

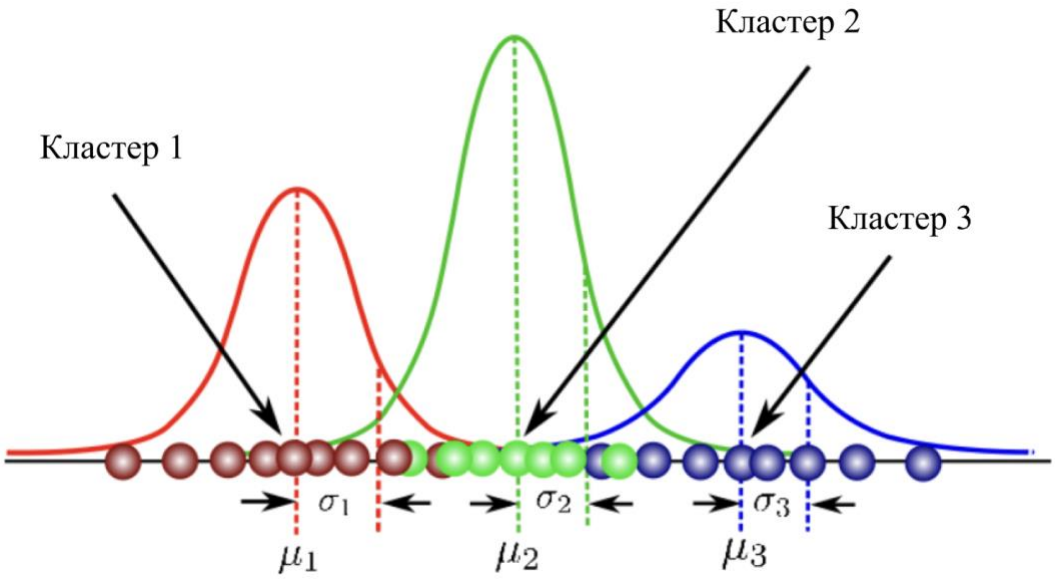

Рисунок 1.10 – Графічне представлення параметрів моделі суміші Гауса

Задача полягає в тому, щоб знайти такі значення параметрів, які забезпечать максимальне значення функції правдоподібності для набору даних. Це можливо за допомогою вирішення такої проблеми:

$$\max_{\Theta} P(X|\Theta) = \max_{\Theta} \prod_{i} P(x_i|\Theta) = \max_{w_j,\mu_j,\Sigma_j} \prod_{i} \left(\sum_{j=1}^{K} w_j \, \mathcal{N}\left(x|\mu_j, \Sigma_j\right) \right) \quad (1.13)$$

Окрім функції правдоподібності, зазвичай, максимізується логарифмічна функція правдоподібності, тому що це дає змогу перейти від добутку ймовірностей до суми, що полегшує обчислення. Це відбувається тому, що натуральний логарифм є монотонно зростаючою увігнутою функцією і не змінює розташування максимуму (місце, де похідна дорівнює нулю, залишиться незмінним).

$$\begin{aligned} \max_{\Theta} \log P(X|\Theta) &= \max_{\Theta} \log \left(\prod_{i} P(x_i|\Theta) \right) = \max_{\Theta} \sum_{i} \log\left(P(x_i|\Theta)\right) \\ &= \max_{\Theta} \sum_{i} \log \left(\sum_{j=1}^{K} w_i \, \mathcal{N}\left(x|\mu_j, \Sigma_j\right) \right) \end{aligned} \quad (1.14)$$

Розглянемо приклад кластеризації даних в одно-вимірному випадку (Рисунок 1.11).

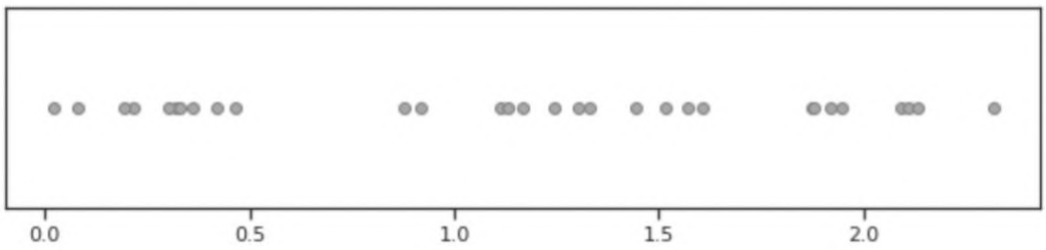

Рисунок 1.11 – Розподіл даних на осі X

Припустимо, що відомими є параметри розподілів (Рисунок 1.12).

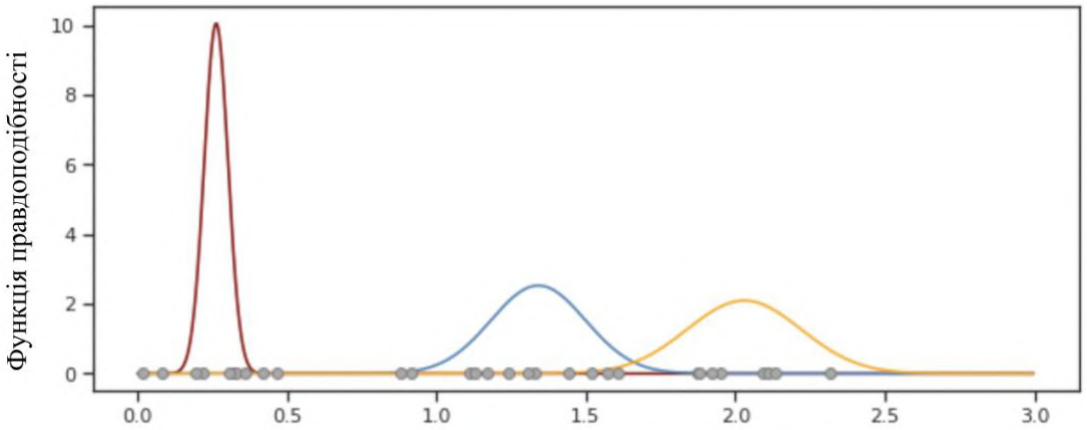

Рисунок 1.12 – Графічне зображення нормальних розподілів

Якщо розташування гаусіан відоме, то можна обчислити ймовірності приналежності даних до кластерів. Для обчислення ймовірності приналежності точки до кожної з гаусіан використовуються формули:

$$P(x_i \text{ належить до кластеру } 1) = \frac{w_1 \mathcal{N}(x_i|\mu_1, \Sigma_1)}{\sum_{j=1}^{K} w_j \mathcal{N}(x_i|\mu_j, \Sigma_j)} \quad (1.15)$$

$$P(x_i \text{ належить до кластеру } 2) = \frac{w_2 \mathcal{N}(x_i|\mu_2, \Sigma_2)}{\sum_{j=1}^{K} w_j \mathcal{N}(x_i|\mu_j, \Sigma_j)} \quad (1.16)$$

$$P(x_i \text{ належить до кластеру } 3) = \frac{w_3 \mathcal{N}(x_i|\mu_3, \Sigma_3)}{\sum_{j=1}^{K} w_j \mathcal{N}(x_i|\mu_j, \Sigma_j)} \quad (1.17)$$

Тепер припустимо, що доступною є інформація про приналежність даних до кластеру (Рисунок 1.13).

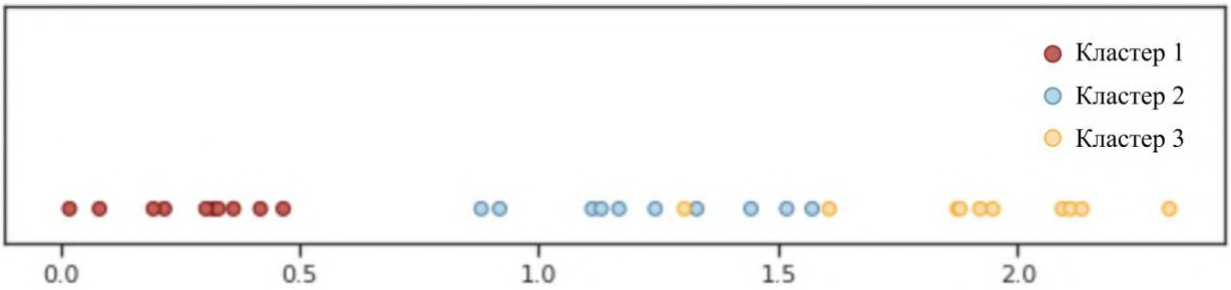

Рисунок 1.13 – Розподіл даних на осі X з маркуванням відношення до кластерів

У такому випадку досить легко можна знайти параметри та зобразити гаусіани:

$$\mu_{MLE} = \frac{1}{N} \sum_{i}^{N} x_i = \bar{x} \quad (1.18)$$

$$\sigma_{MLE} = \sqrt{\frac{1}{N} \sum_{i}^{N} (x_i - \bar{x})^2} \quad (1.19)$$

Застосувавши ці формули до кожної з груп точок, отримаємо ті ж дані, що і в попередньому випадку (Рисунок 1.12).

З цього можна зробити висновок, що, якщо відомі параметри нормального розподілу, то можна провести кластеризацію даних. І навпаки, маючи дані та

інформацію до якого кластеру вони відносяться, можна знайти параметри нормального розподілу.

На практиці попередньо розглянуті сценарії зустрічаються доволі рідко. У такому випадку для оцінки параметрів GMM використовується EM-алгоритм (expectation maximization). Головна ідея полягає в тому, щоб максимізувати значення функції правдоподібності:

$$\max_{\Theta} \log P(X|\Theta) = \max_{\Theta} \log \left(\prod_{i} P(x_i|\Theta) \right) = \max_{\Theta} \sum_{i} \log\big(P(x_i|\Theta)\big) \tag{1.20}$$

Алгоритм шукає параметри моделі ітеративно. Його перевагою є те, що з кожною ітерацією максимальна правдоподібність зростає, тобто гарантовано наближається до свого максимуму. Кожна ітерація складається з двох кроків: E-крок (expectation) та M-крок (maximization).

Отже, потрібно запускати обидва кроки ітеративно та максимізувати функцію правдоподібності, доки вона не зійдеться. EM-алгоритм починається з ініціалізації, коли випадковим чином параметрам моделі призначаються прийнятні значення на основі даних.

На кожній EM ітерації використовуються наступні формули, які гарантують монотонне підвищення значення функції правдоподібності:

$$P(j|x_i, \Theta) = \frac{w_j \mathcal{N}\big(x_i|\mu_j, \Sigma_j\big)}{\sum_{k=1}^{K} w_k \mathcal{N}(x_i|\mu_k, \Sigma_k)} \tag{1.21}$$

$$\bar{w}_j = \frac{1}{N} \sum_{i=1}^{N} P(j|x_i, \Theta) \tag{1.22}$$

$$\bar{\mu}_j = \frac{\sum_{i=1}^{N} P(j|x_i, \Theta) x_i}{\sum_{i=1}^{N} (j|x_i, \Theta)} \tag{1.23}$$

$$\bar{\Sigma}_j = \frac{\sum_{i=1}^{N} P(j|x_i, \Theta) x_i^2}{\sum_{i=1}^{N} (j|x_i, \Theta)} - \bar{\mu}_j^2 \tag{1.24}$$

Приклад роботи ЕМ-алгоритму зображено нижче (Рисунок 1.14 та 1.15).

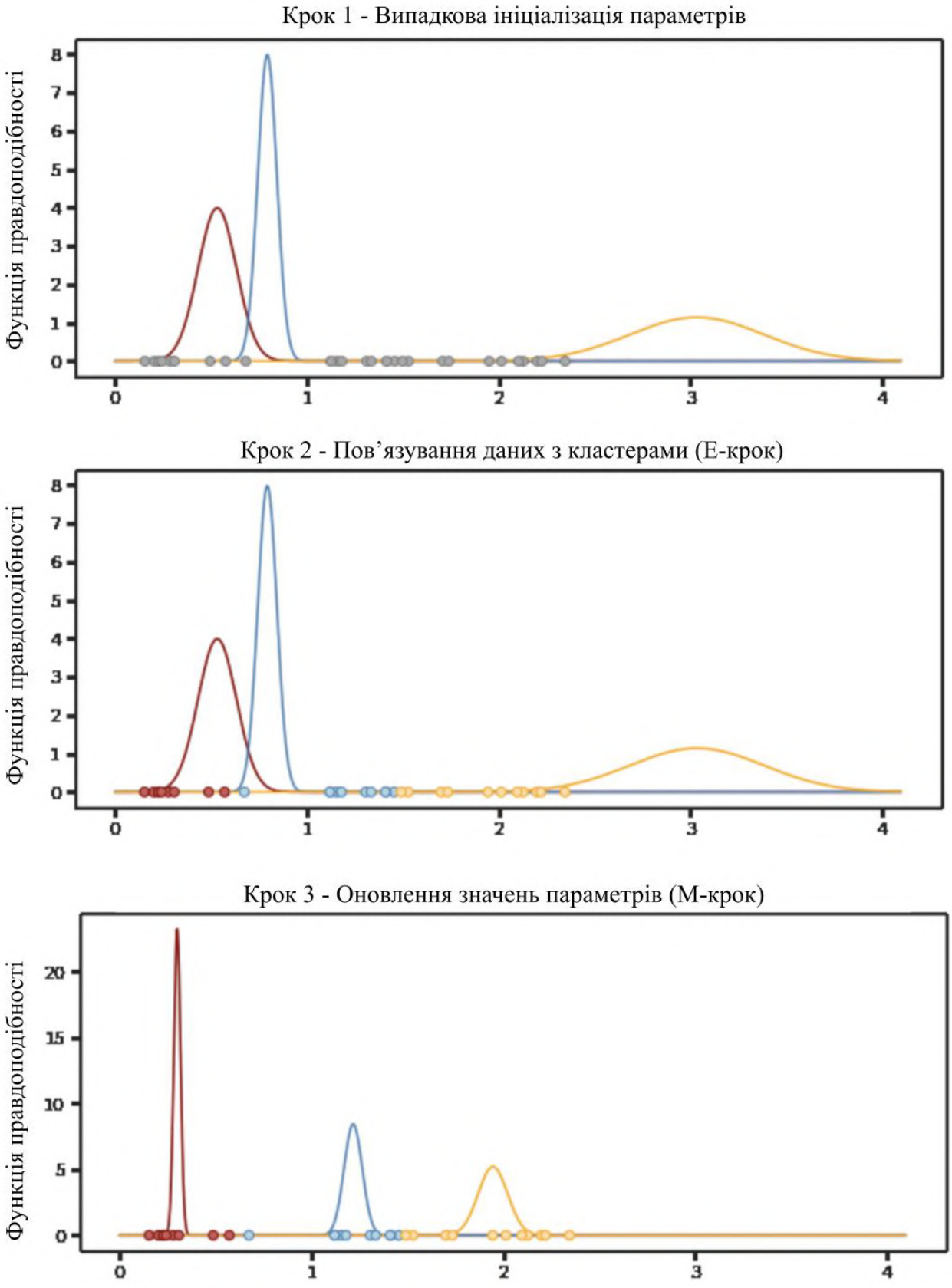

Рисунок 1.14 – Схематичне зображення першої ітерації ЕМ-алгоритму

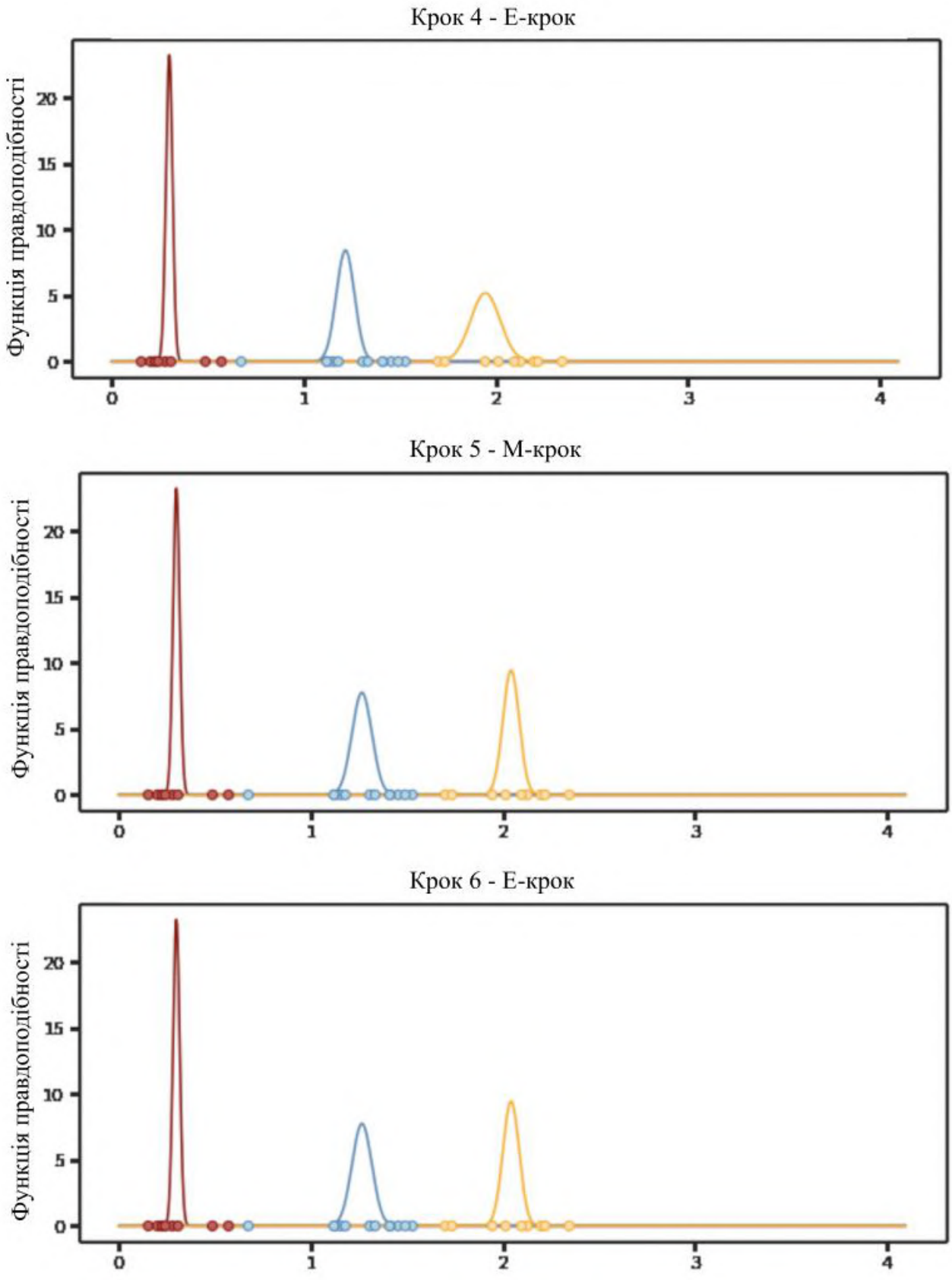

Рисунок 1.15 – Схематичне зображення наступних ітерацій ЕМ-алгоритму

Збіжність, зазвичай, виявляється шляхом обчислення значення логарифмічної правдоподібності після кожної ітерації. Якщо значення після ітерації змінилося несуттєво у порівнянні з попереднім, то ЕМ-алгоритм закінчує роботу. Також одним із способів визначення моменту зупинки алгоритму є вказання кількості ітерацій, які потрібно виконати. Ці два методи є найбільш вживаними на практиці.

Якщо порівняти рисунки 1.14 та 1.15, очевидно, що на кроці 6 вже не відбувається помітних змін. Це означає, що роботу ЕМ-алгоритму можна завершувати.

Загалом, ЕМ-алгоритм можна схематично представити так:

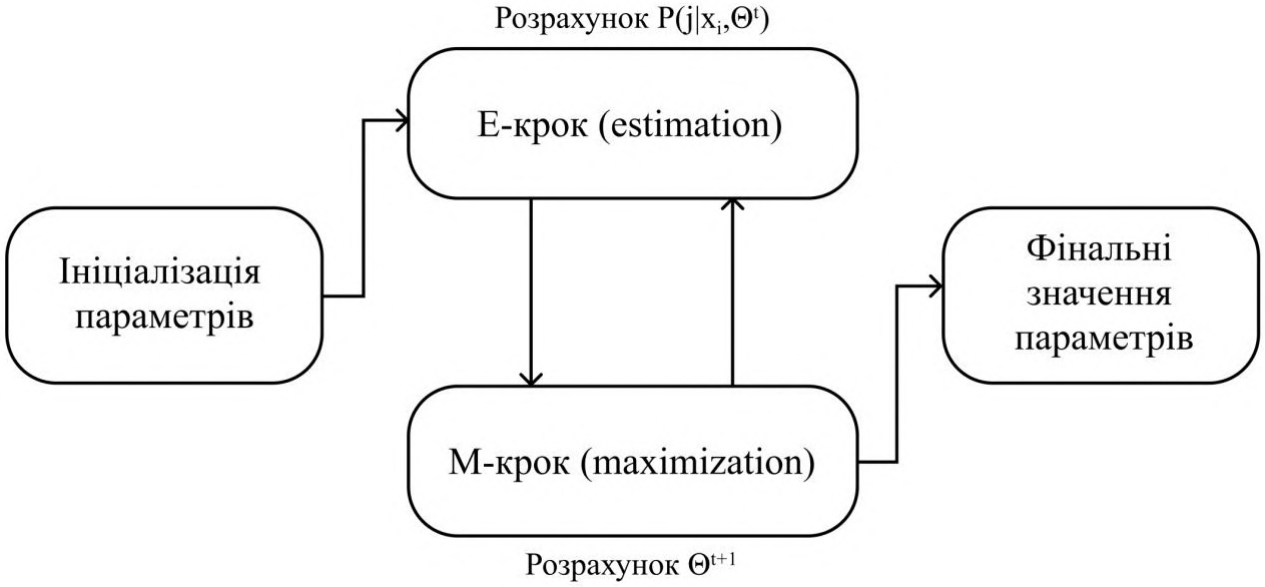

Рисунок 1.16 – Загальна схема роботи ЕМ-алгоритму

2 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

2.1 Аналіз роботи алгоритмів зниження рівня шуму

Для аналізу роботи алгоритмів було записано фрагмент мовлення довжиною 30 с. Спектрограма та АЧХ (амплітудно-частотна характеристика) сигналу зображені нижче (Рисунок 2.1).

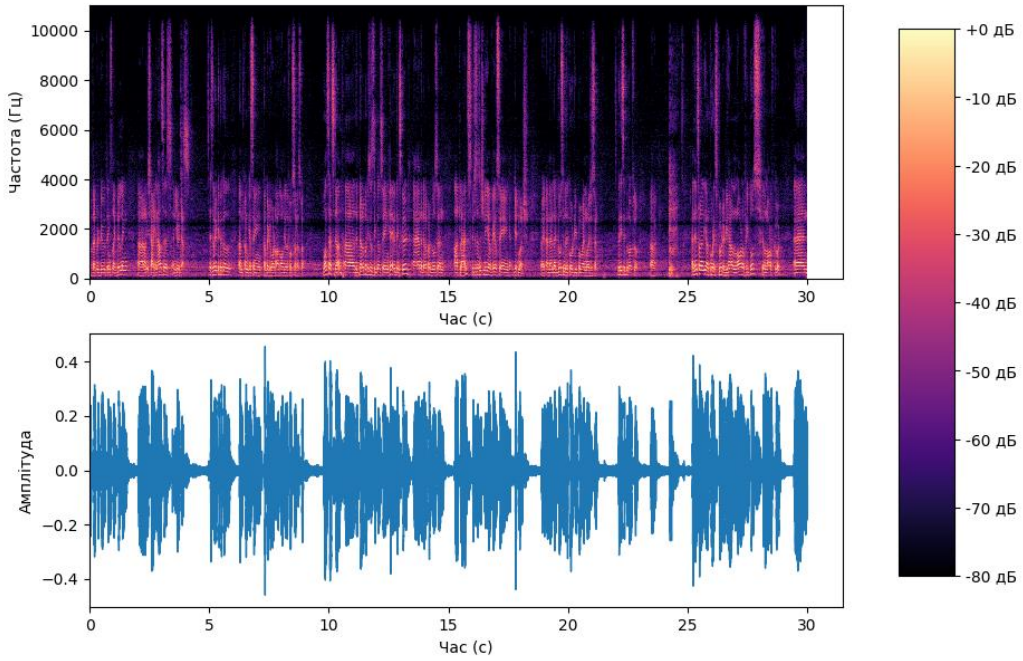

Рисунок 2.1 – Спектрограма та АЧХ оригінального запису

Проаналізувавши спектрограму, можна зробити висновок про наявність шуму. Скоріше за все, цей шум був зумовлений мікрофоном.

Для подальшого аналізу алгоритмів зниження рівня шуму було згенеровано шум із частотою в діапазоні від 0 до 2 кГц. Характеристики згенерованого шуму зображено нижче (Рисунок 2.2).

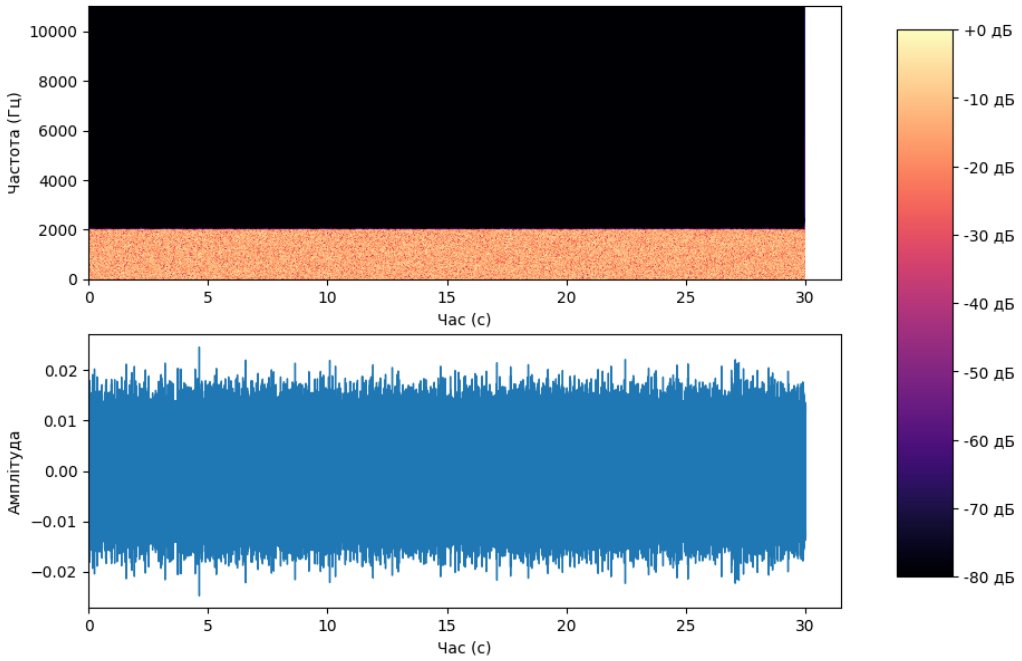

Рисунок 2.2 – Спектрограма та АЧХ запису шуму

Наступним кроком до оригінального аудіофайлу було додано згенерований шум. Характеристики отриманого сигналу зображені нижче (Рисунок 2.3).

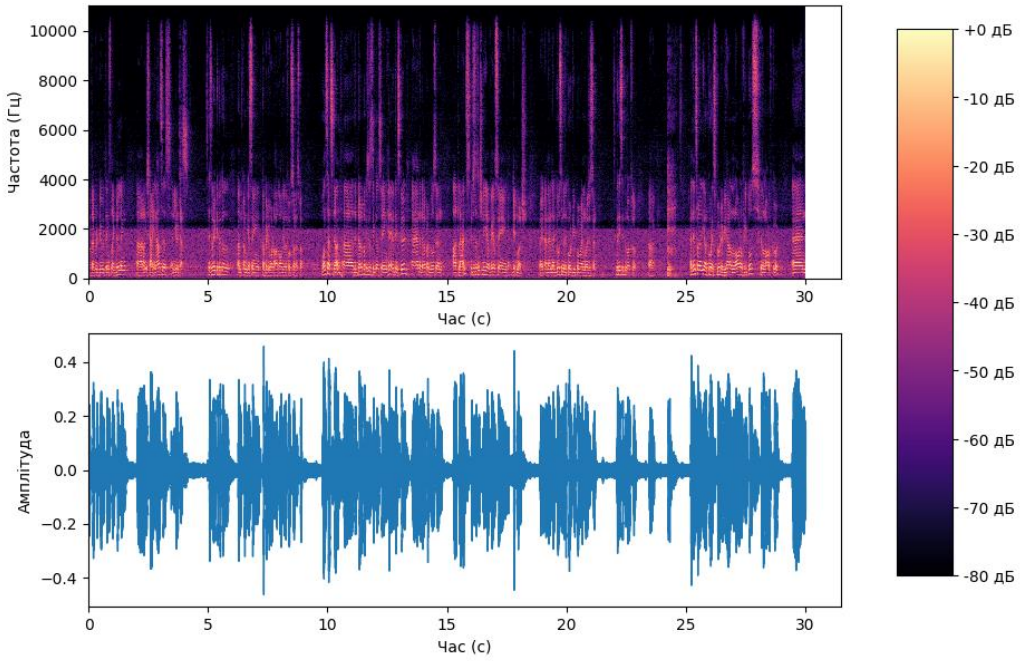

Рисунок 2.3 – Спектрограма та АЧХ оригінального запису, об’єднаного з шумом

Далі аудіофайл пройшов процедуру позбавлення від шуму за допомогою кожного з алгоритмів. Характеристики отриманих сигналів наведено нижче (Рисунок 2.4 та Рисунок 2.5).

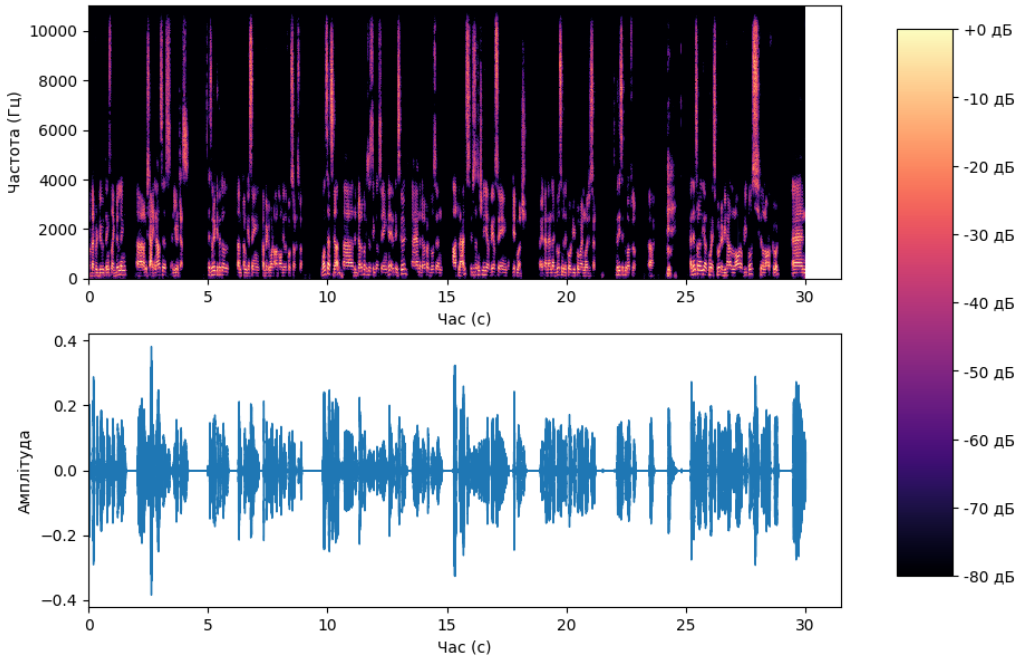

Рисунок 2.4 – Спектрограма та АЧХ отриманого запису після застосування методу позбавлення від нестаціонарного шуму

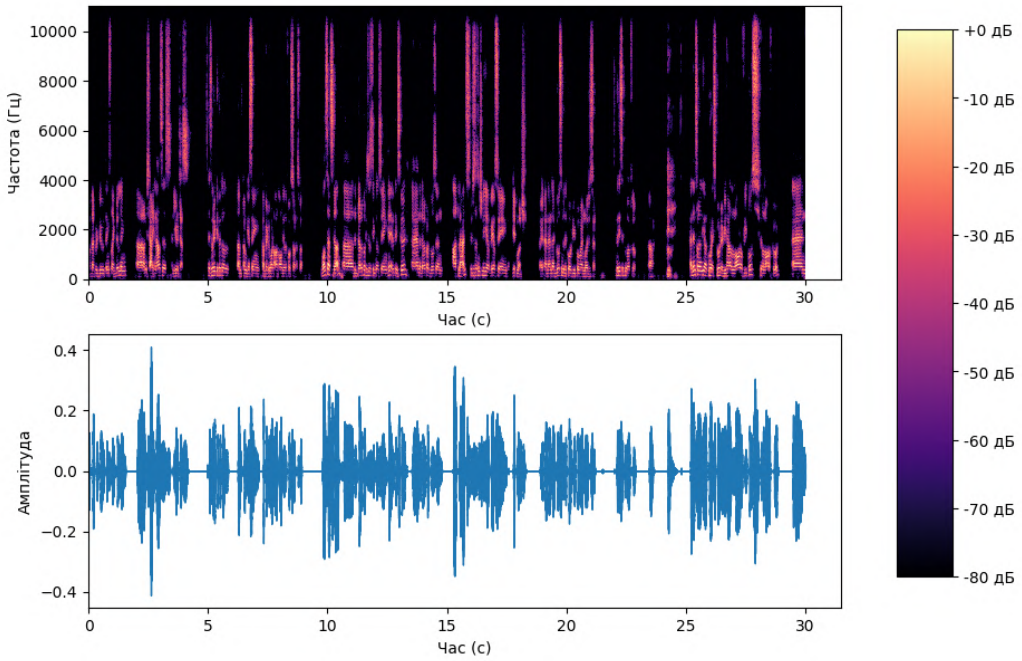

Рисунок 2.5 – Спектрограма та АЧХ отриманого запису після застосування методу позбавлення від стаціонарного шуму

Проаналізувавши отримані результати можна стверджувати, що обидва алгоритми позбавилися не тільки від доданого шуму, а і від того, який був присутній в оригінальному аудіозаписі. На перший погляд обидва алгоритми дали однакові результати, але проаналізувавши вихідні аудіофайли можна

стверджувати, що алгоритм позбавлення від нестаціонарного шуму є більш агресивним і видалив не тільки шум, але і деякі фрагменти мовлення. В контексті цієї роботи, втрата цих фрагментів може негативно вплинути на подальший аналіз і вилучення ознак абонентів, що призведе до збільшення помилкових спрацювань. Тому було прийнято рішення про використання алгоритму позбавлення від стаціонарного шуму. Цей вибір також зумовлений тим, що при розробці системи ідентифікації по голосу робиться припущення, що при записі голосів абонентів буде присутній тільки стаціонарний шум. Наприклад, шум мікрофону, кондиціонера, системного блоку, тощо.

2.2 Аналіз алгоритму детектування голосу

Для аналізу роботи алгоритму VAD було записано фрагмент мовлення довжиною 30 с. Характеристики сигналу зображені нижче (Рисунок 2.6).

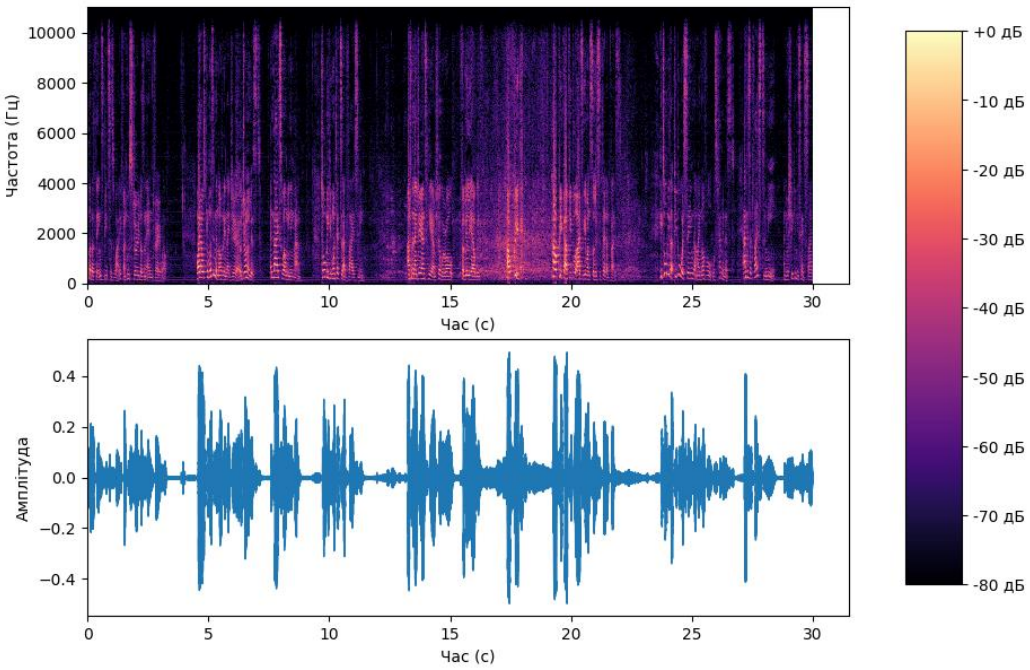

Рисунок 2.6 – Спектрограма та АЧХ оригінального запису

Візуально проаналізувавши зображення сигналу, можна помітити велику кількість пауз між мовленням. При аналізі мовлення це може завадити його правильній обробці і призвести до хибних результатів на подальших етапах.

В розробленій системі було використано бібліотеку py-webrtcvad. Ця бібліотека надає можливість вибрати один з трьох режимів, які визначають степінь агресивності алгоритму.

Всього є три режими, в яких змінюється ступінь агресивності. Вони позначаються таким чином: 1 – нормальний, 2 – агресивний, 3 – дуже агресивний.

Результати обробки оригінального аудіофайлу за допомогою алгоритму зі встановленим режимом агресивності 1 наведено нижче (Рисунок 2.7 та Рисунок 2.8).

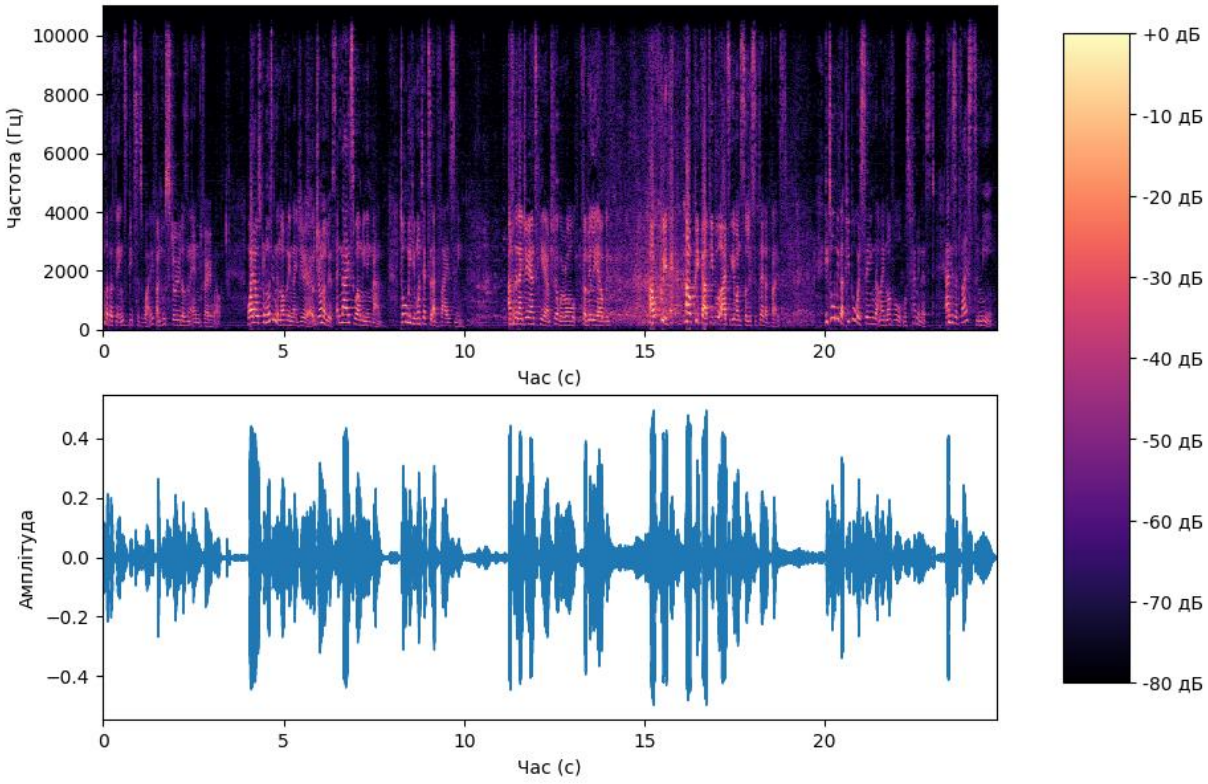

Рисунок 2.7 – Спектрограма та АЧХ сигналу, отриманого після використання алгоритму зі встановленим режимом агресивності 1

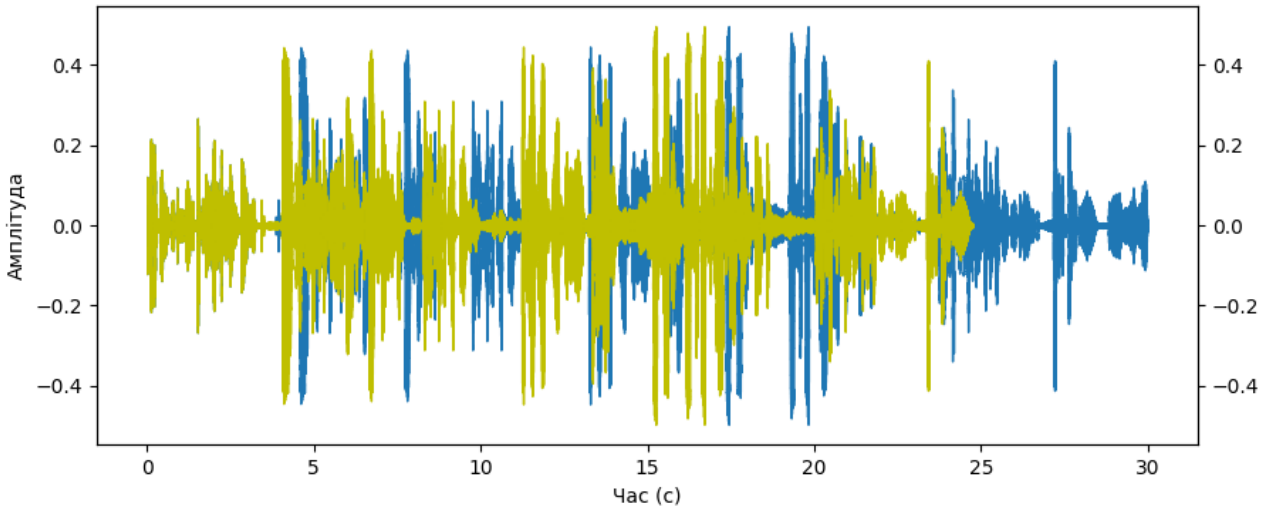

Рисунок 2.8 – Порівняння отриманого та оригінального сигналів

Результати обробки оригінального аудіофайлу за допомогою алгоритму зі встановленим режимом агресивності 2 наведено нижче (Рисунок 2.9 та Рисунок 2.10).

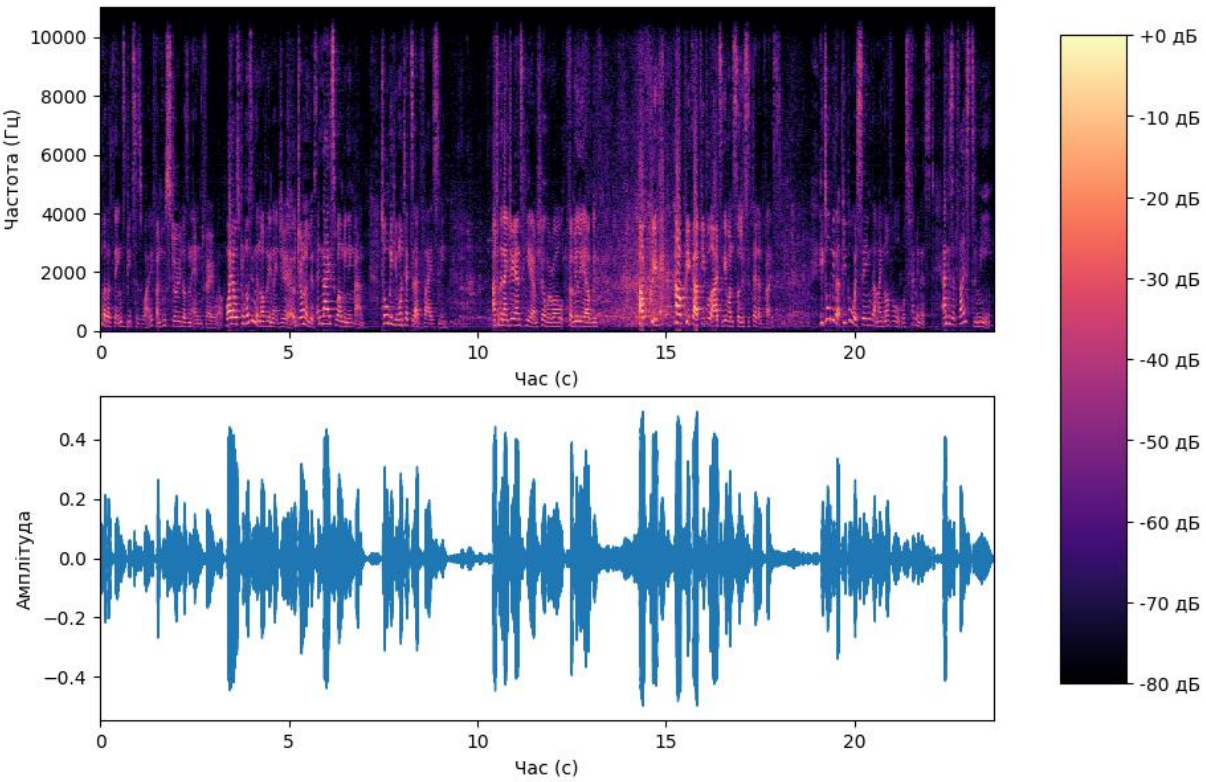

Рисунок 2.9 – Спектрограма та АЧХ сигналу, отриманого після використання алгоритму зі встановленим режимом агресивності 2

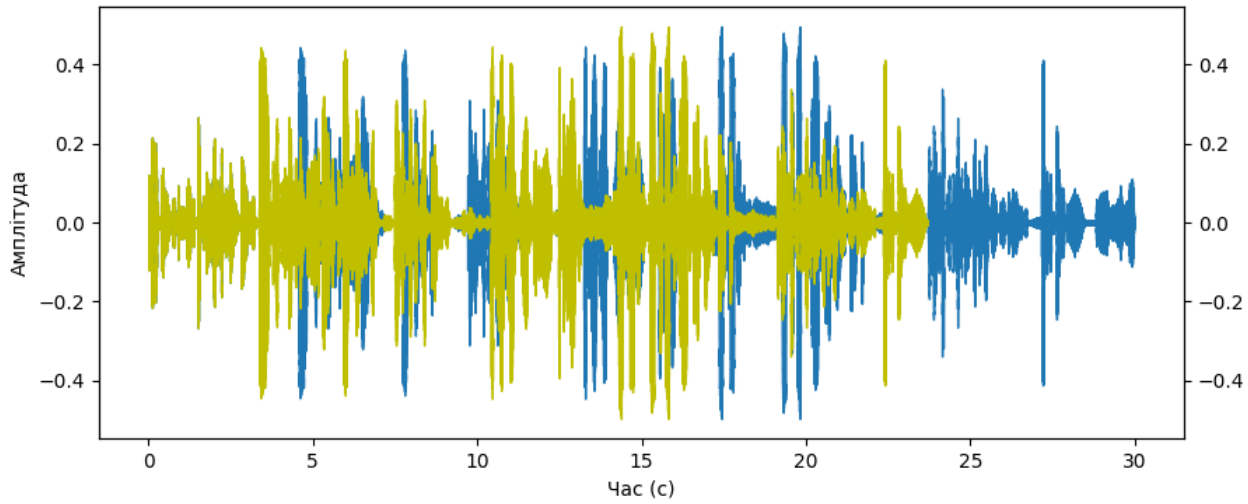

Рисунок 2.10 – Порівняння отриманого та оригінального сигналів

Результати обробки оригінального аудіофайлу за допомогою алгоритму зі встановленим режимом агресивності 3 наведено нижче (Рисунок 2.11 та Рисунок 2.12).

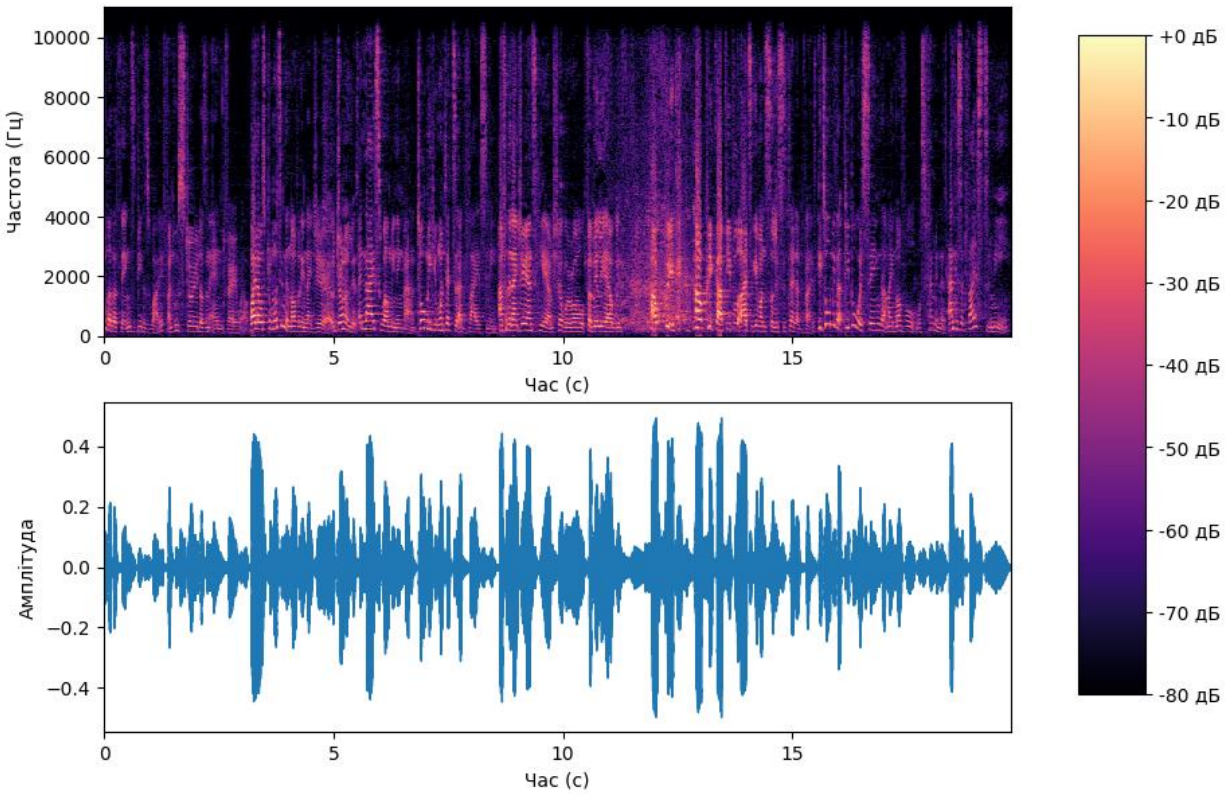

Рисунок 2.11 – Спектрограма та АЧХ сигналу, отриманого після використання алгоритму зі встановленим режимом агресивності 3

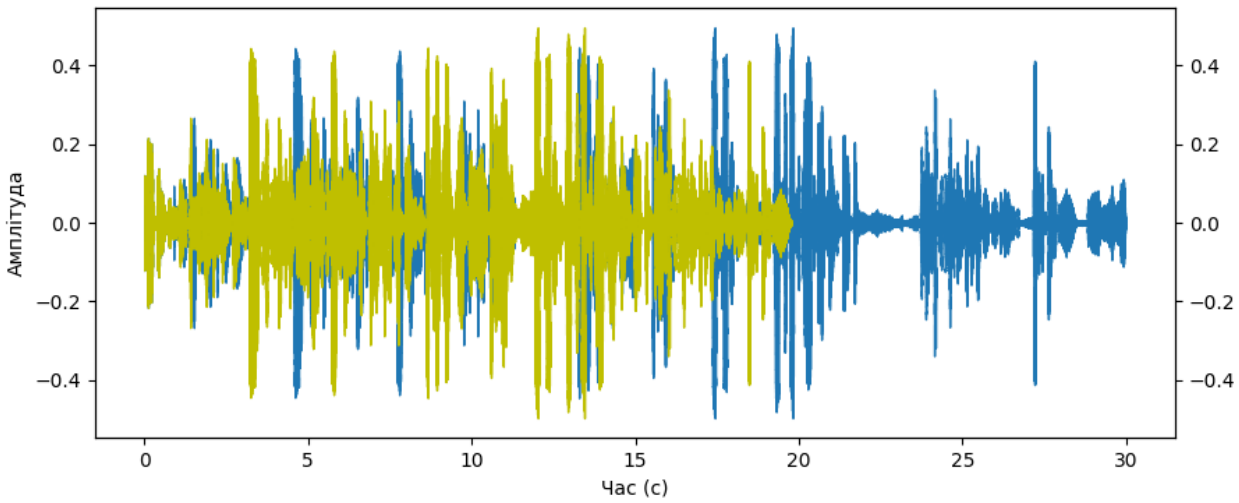

Рисунок 2.12 – Порівняння отриманого та оригінального сигналів

Проаналізувавши отримані результати, можна стверджувати, що режим 1 є найменш агресивним, що, у свою чергу, зумовлює наявність коротких пауз у вихідному аудіосигналі. Режим 2 виявився більш ефективним за режим 1, так як кількість коротких пауз зменшилась. Цей режим є оптимальним при необхідності збереження цілісності мовлення, оскільки в цьому випадку вона не порушується.

Режим 3 виявився найбільш агресивним. Він видалив не тільки паузи, а й деякі фрагменти мовлення. Наприклад, пом’якшені початки або закінчення слів.

В розробленій системі було прийнято рішення про використання алгоритму зі встановленим рівнем агресивності 3. Це рішення зумовлене тим, що у контексті цієї роботи збереження цілісності мовлення не є пріоритетною задачею, оскільки важливо отримати якісний зразок голосу людини для подальшого вилучення характерних ознак. При цьому втрата незначної кількості фрагментів мовлення кардинально не вплине на якість вилучених ознак в той час, як наявність пауз та сторонніх звуків призведе до погіршення результатів аналізу та подальших помилок.

2.3 Вилучення мел-частотних кепстральних коефіцієнтів

Для вилучення MFCC було записано фрагмент мовлення довжиною 23 секунди. Його характеристики зображено нижче (Рисунок 2.6).

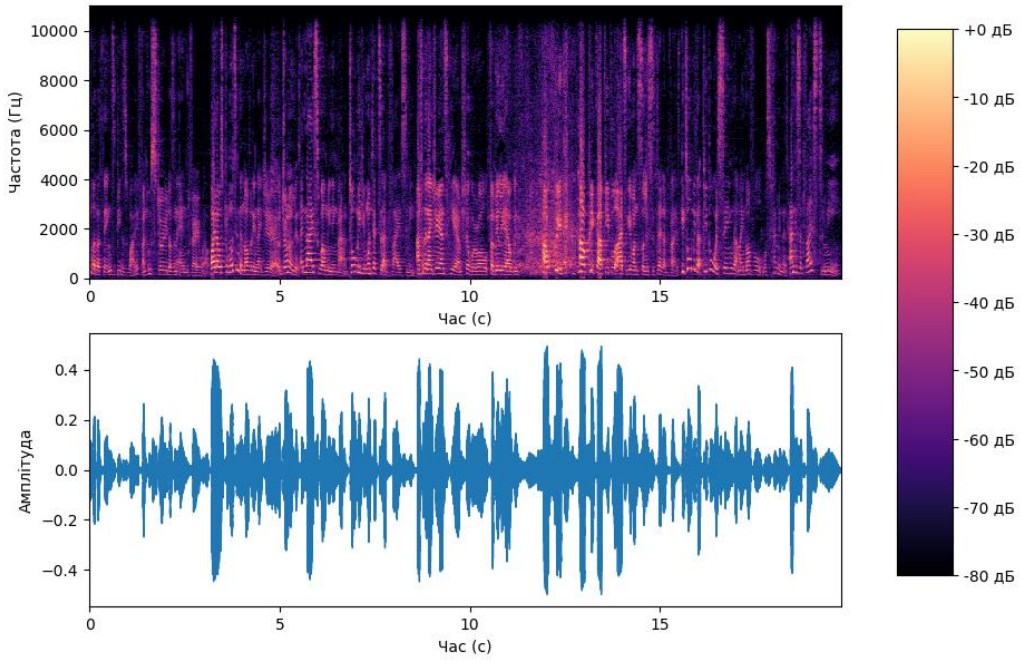

Рисунок 2.13 – Спектрограма та АЧХ оригінального запису

Після виконання всіх перетворень, що описані в пункті 1.5, було отримано мел-частотні кепстральні коефіцієнти. Оскільки, крім самого спектра, індивідуальність голосу формується швидкістю та прискореннями, було розраховано першу та другу похідні для MFCC (Рисунок 2.7).

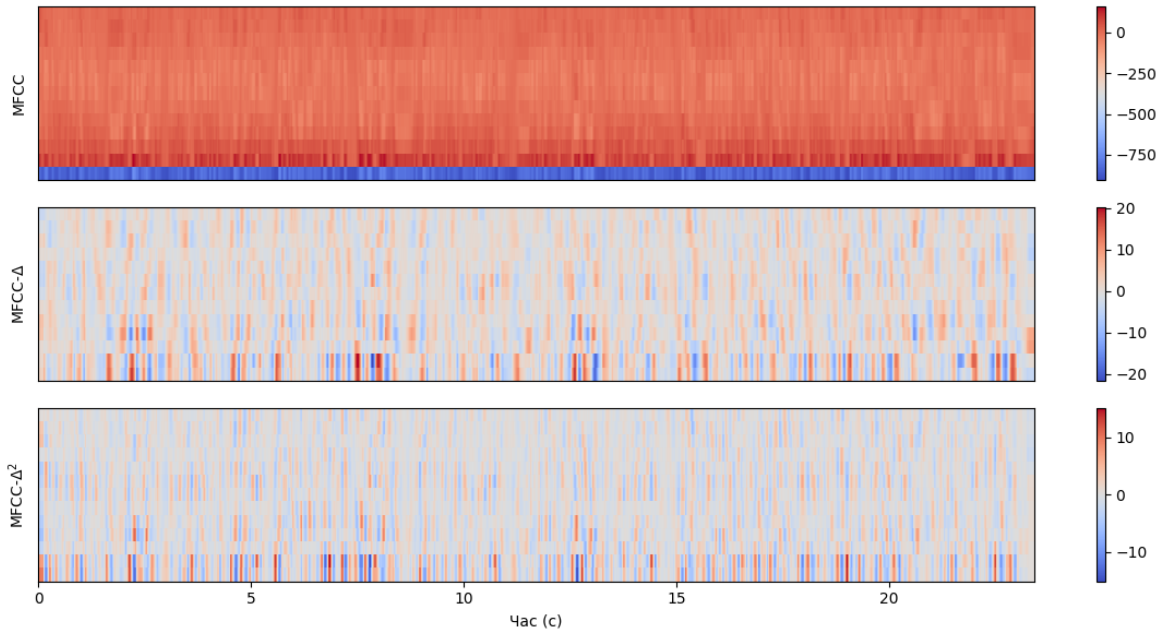

Рисунок 2.14 – Розраховані мел-частотні кепстральні коефіцієнти та похідні першого та другого порядків

3 ПЕРЕВІРКА ПРАЦЕЗДАТНОСТІ СИСТЕМИ

3.1 Побудова бази даних

Перед тим, як використовувати розроблену систему, потрібно створити базу даних, яка міститиме дані, необхідні для ідентифікації осіб. Отже, аудіодані кожної з осіб повинні бути оброблені за схемою, наведеною нижче (Рисунок 3.1).

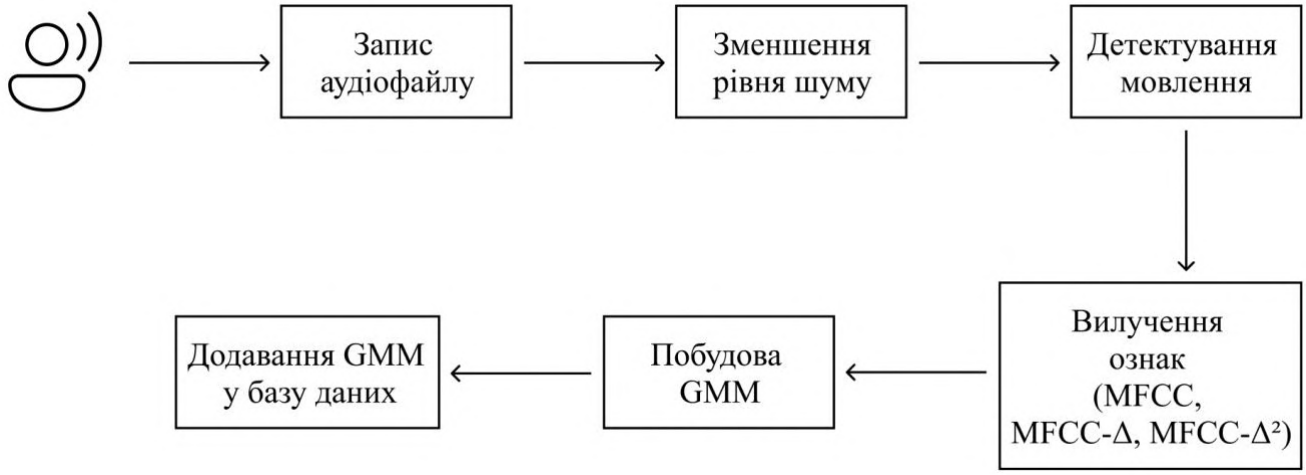

Рисунок 3.1 – Покрокова схема додавання біометричних даних особи у базу даних

Тестування системи проводилось на комп’ютері з такими характеристиками:

- Процесор – Apple M1 Pro;
- Об’єм ОЗУ – 16 ГБ.

Для побудови бази даних було зібрано зразки мовлення 100 осіб. До зразка мовлення кожної з осіб було застосовано алгоритм зменшення рівня шуму та алгоритм детектування мовлення. Далі з кожного зразка отримано групи по 3 та 5 аудіофрагментів тривалістю 2 с, 5 с, 10 с. Таким чином, для кожної особи створено 6 груп аудіофайлів.

Наступним кроком було проведення замірів часу, потрібного для побудови бази даних залежно від кількості аудіофайлів, їх довжини та кількості гауссіан у моделі суміші Гауса.

Час побудови баз даних для груп, які містять 3 та 5 аудіфайлів довжиною 2 секунди наведено нижче (відповідно, Рисунок 3.2 та Рисунок 3.3).

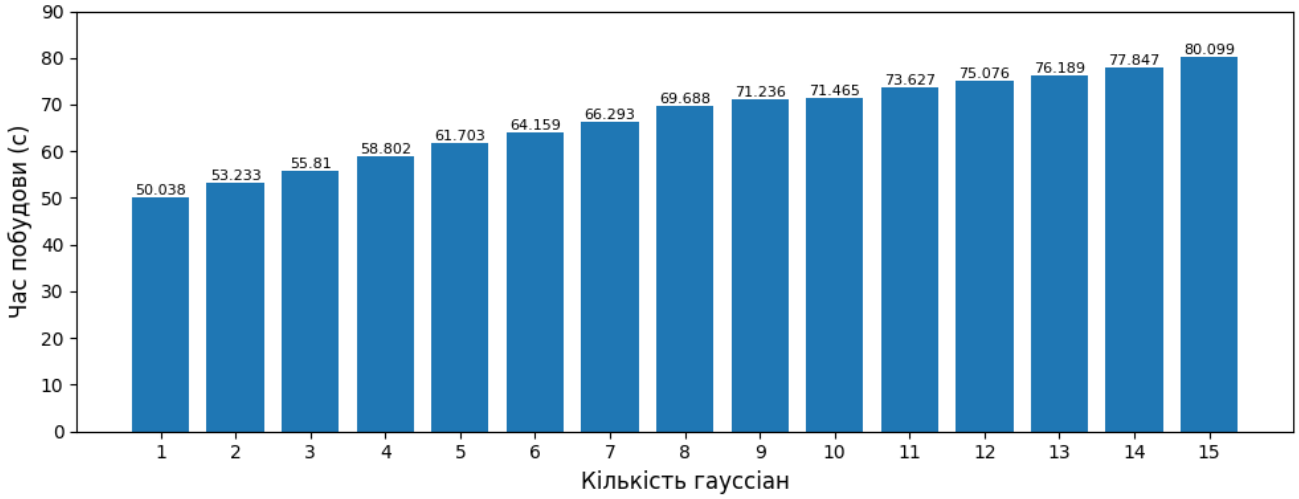

Рисунок 3.2 – Графік залежності часу побудови бази даних від кількості гауссіан для груп з 3-х аудіофайлів довжиною 2 секунди

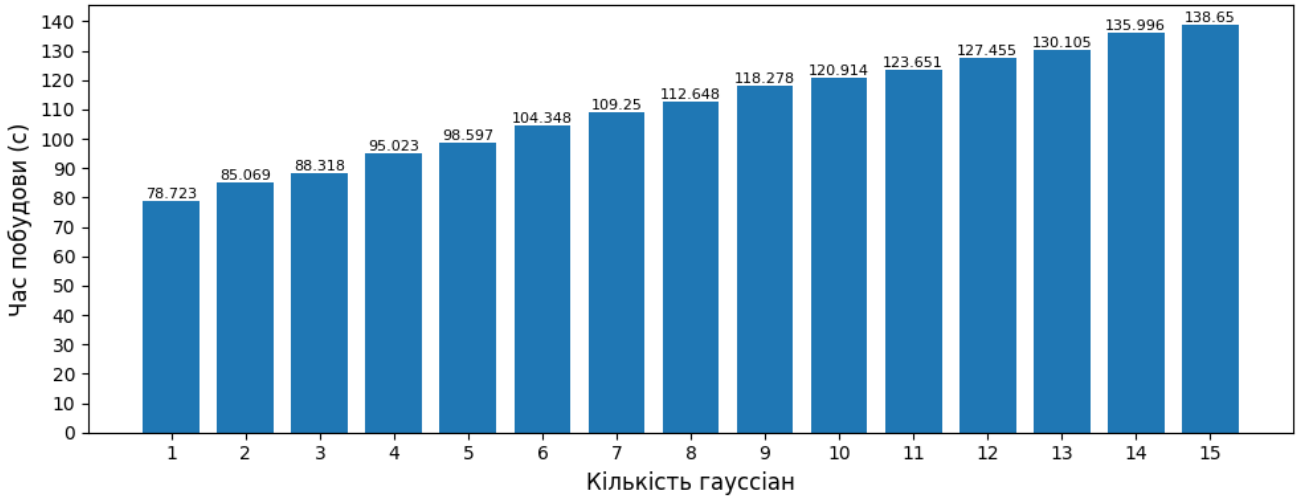

Рисунок 3.3 – Графік залежності часу побудови бази даних від кількості гауссіан для груп з 5-и аудіофайлів довжиною 2 секунди

Час побудови баз даних для груп, які містять 3 та 5 аудіфайлів довжиною 5 секунд наведено нижче (відповідно, Рисунок 3.4 та Рисунок 3.5).

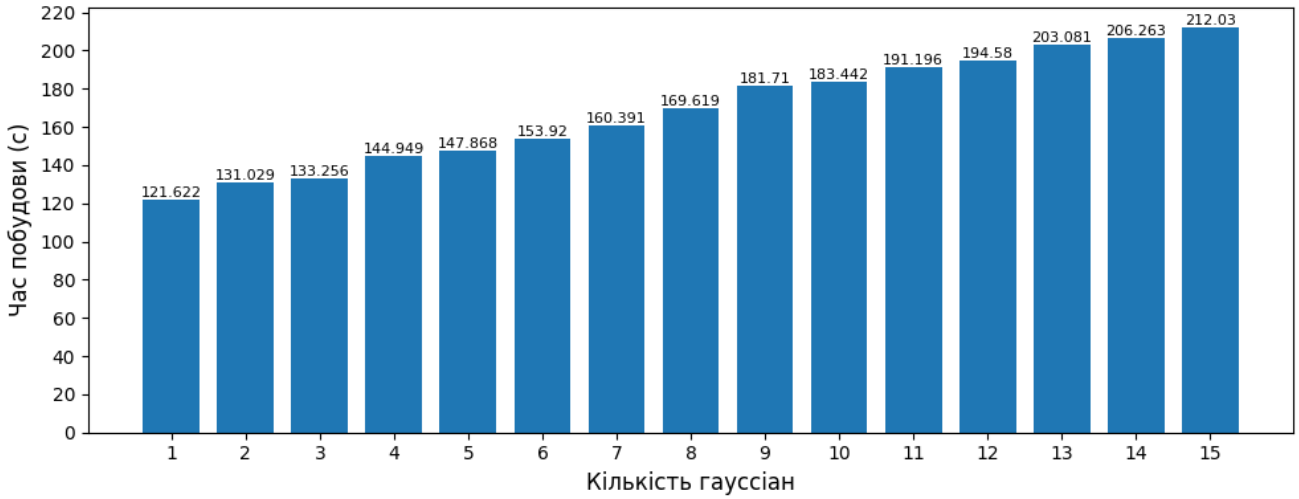

Рисунок 3.4 – Графік залежності часу побудови бази даних від кількості гауссіан для груп з 3-х аудіофайлів довжиною 5 секунд

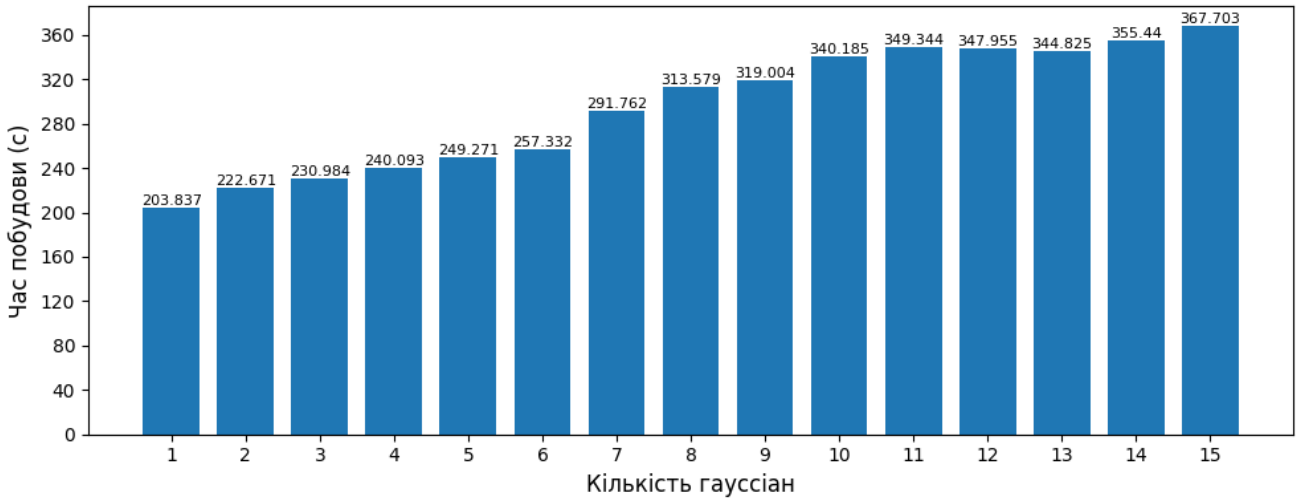

Рисунок 3.5 – Графік залежності часу побудови бази даних від кількості гауссіан для груп з 5-и аудіофайлів довжиною 5 секунд

Час побудови баз даних для груп, які містять 3 та 5 аудіфайлів довжиною 10 секунд наведено нижче (відповідно, Рисунок 3.6 та Рисунок 3.7).

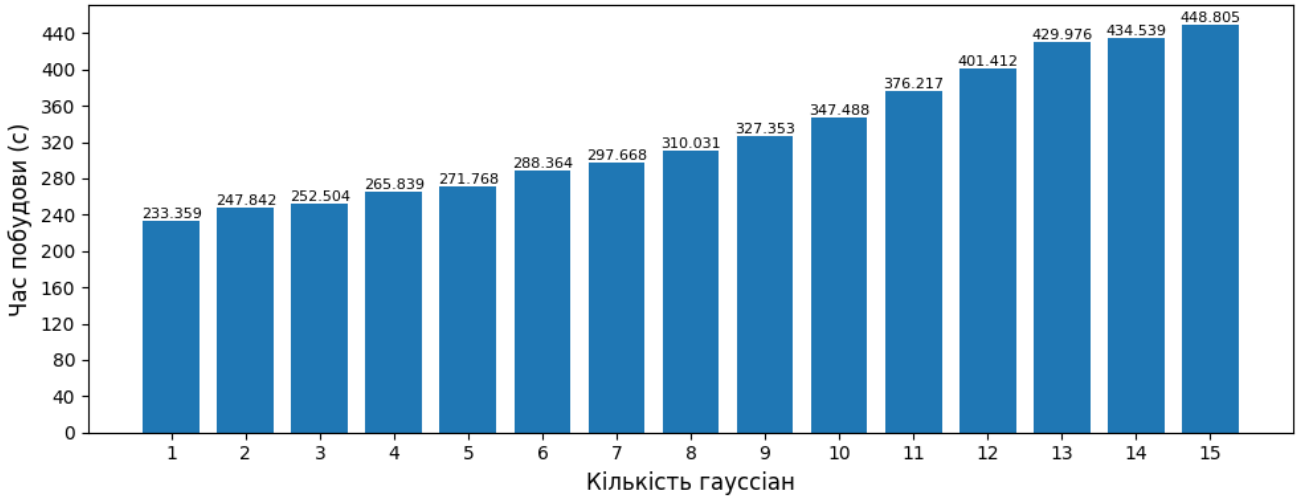

Рисунок 3.6 – Графік залежності часу побудови бази даних від кількості гауссіан для груп з 3-х аудіофайлів довжиною 10 секунд

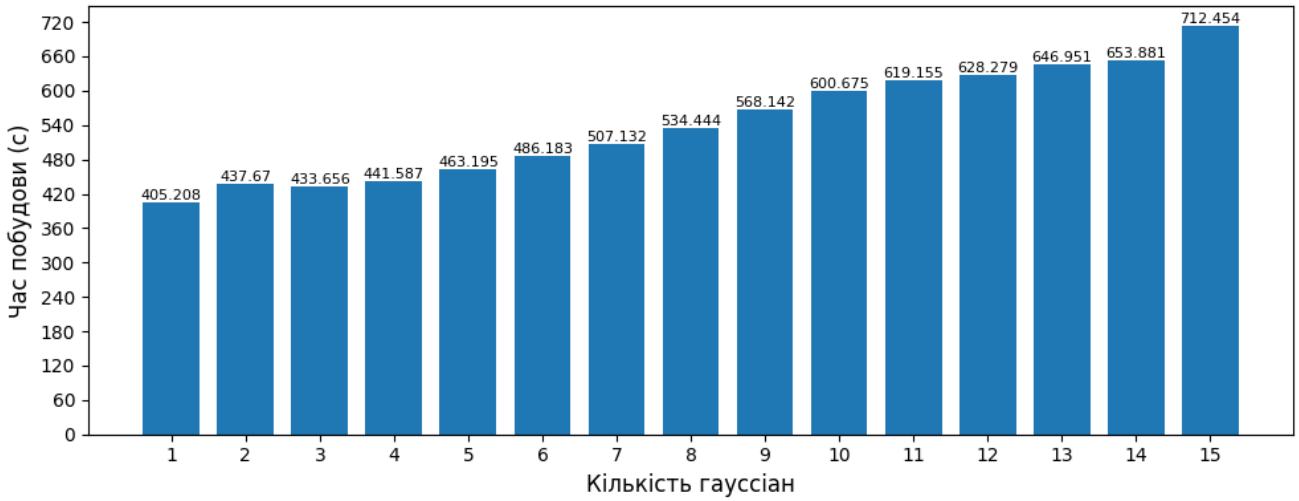

Рисунок 3.7 – Графік залежності часу побудови бази даних від кількості гауссіан для груп з 5-и аудіофайлів довжиною 10 секунд

Проаналізувавши отримані результати, можна стверджувати, що збільшення кількості гауссіан призводить до лінійного збільшення часу, необхідного для побудови бази даних. Також чим довша загальна тривалість аудіофайлів групи, тим більше часу потрібно на побудову БД (база даних).

Таким чином, найшвидше було побудовано базу даних для груп з 3-х аудіофайлів довжиною 2 секунди зі встановленою кількістю гауссіан 1. Побудова зайняла 50,038 секунд і її розмір складає 3,7 МБ.

Відповідно до попередньо зроблених висновків, очевидно, що найбільше часу знадобилось для побудови бази даних для груп з 5-и аудіофайлів довжиною

10 секунд і зі встановленою кількістю гауссіан 15. Побудова зайняла 712,454 секунд і її розмір складає 54,2 МБ.

Слід зауважити, що розміри вихідних баз даних є набагато меншими за розміри вхідних аудіоданих. Наприклад, розмір найбільшої бази даних складає 54,2 МБ в той час, як розмір відповідних вхідних даних дорівнює 865,2 МБ.

3.2 Розпізнавання особи

Після побудови бази даних осіб система є готовою до розпізнавання. яке проходить за схемою, що зображена далі (Рисунок 3.8).

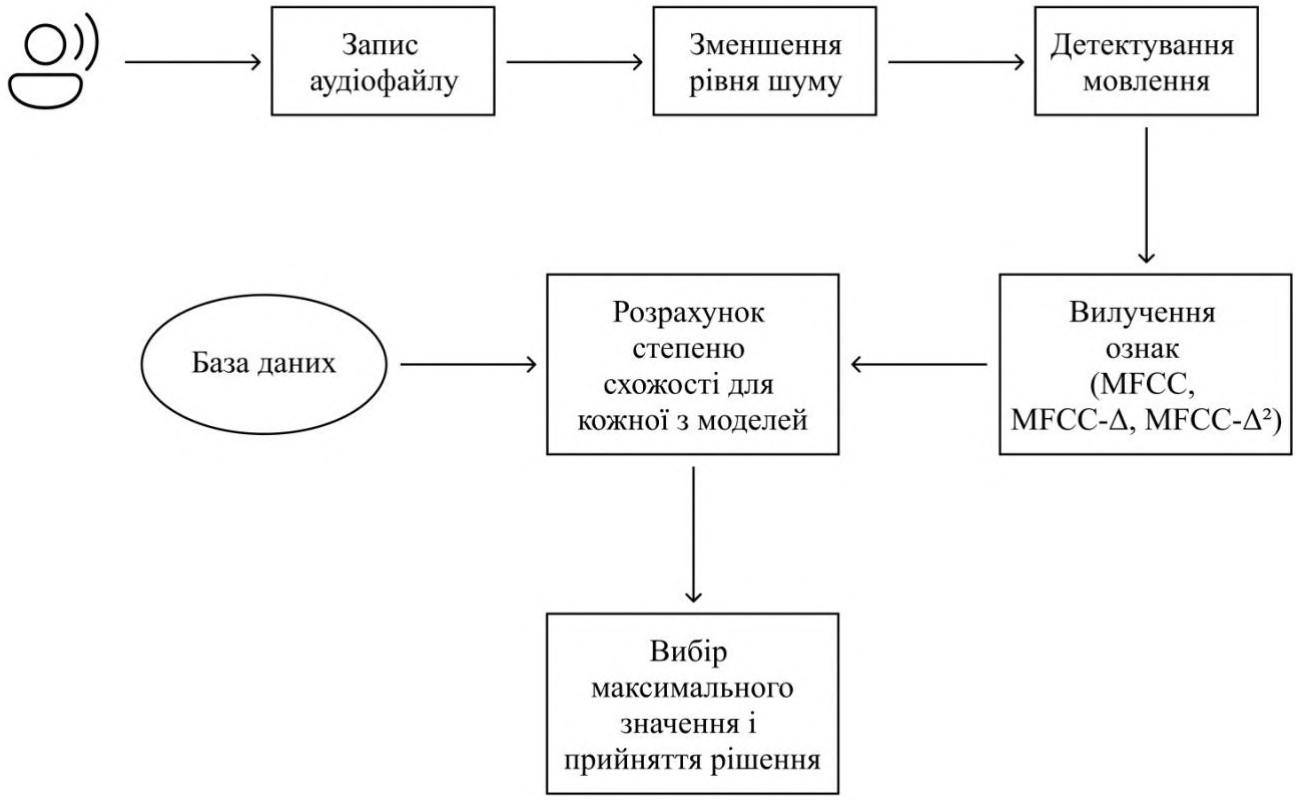

Рисунок 3.8 – Покрокова схема розпізнавання особи

Для отримання чисельних показників ефективності розпізнавання було проведено тестування з використанням кожної із попередньо створених баз даних.

Для прикладу розглянемо спосіб визначення ймовірності правильного розпізнавання особи при використанні груп перевірочних аудіофрагментів тривалістю 2 секунди і бази даних, що була побудована за допомогою груп із 3-х фрагментів тривалістю 2 секунди і кількістю гауссіан 1.

Кожен з перевірочних аудіофайлів певної особи використовується для розпізнавання за допомогою вищеописаного алгоритму (Рисунок 3.8). Для отримання чисельних показників схожості вилучених ознак з кожною із моделей, потрібно обчислити значення функції правдоподібності, використавши у якості параметрів ознаки і модель. Для цього використовується формула (1.20).

Таким чином, маємо 5 результатів розпізнавання. На основі цих даних обчислюється ймовірність правильного розпізнавання цієї особи. Обчисливши ці значення для кожної з осіб і розрахувавши середнє значення, отримаємо ймовірність правильного розпізнавання при використанні перевірочного аудіофрагменту довжиною 2 секунди.

Аналогічно проводяться розрахунки для інших груп перевірочних аудіофрагментів та кожної з баз даних. Результати обчислень для кожної з баз даних наведено нижче (Рисунки 3.9-3.14).

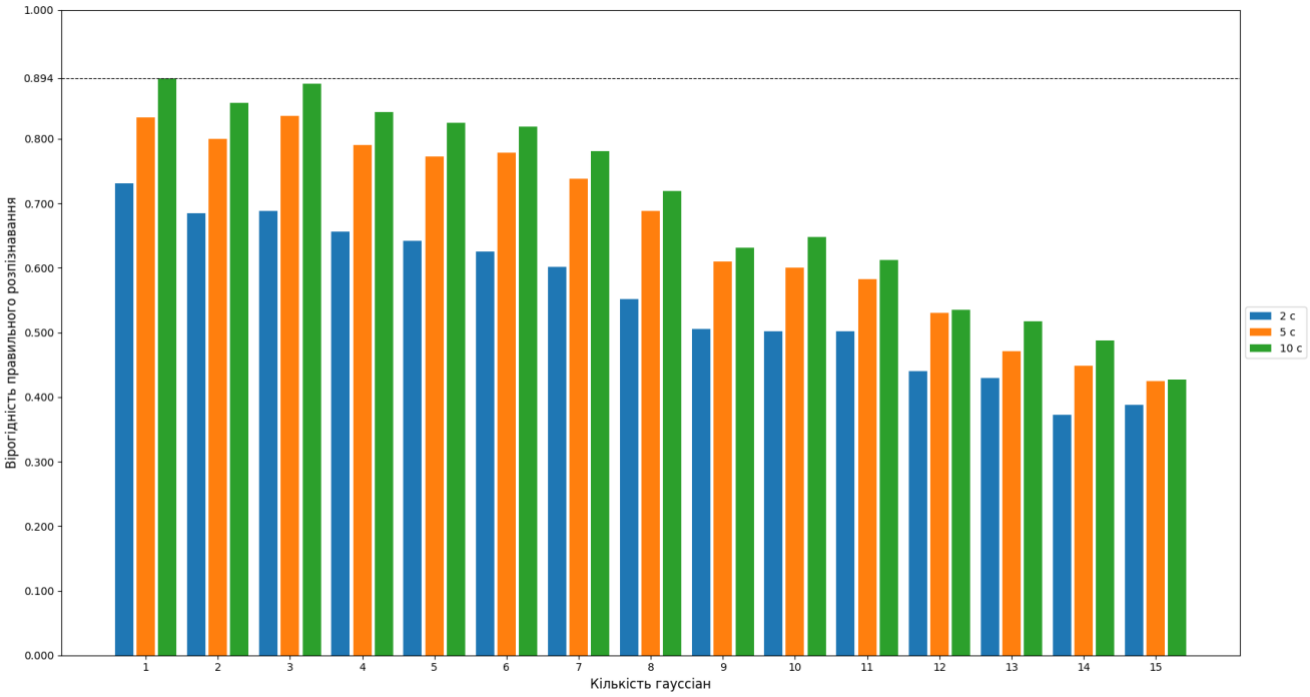

Рисунок 3.9 – Ймовірності правильного розпізнавання при використанні бази даних, побудованої на основі груп з 3-х аудіофрагментів тривалістю 2 секунди

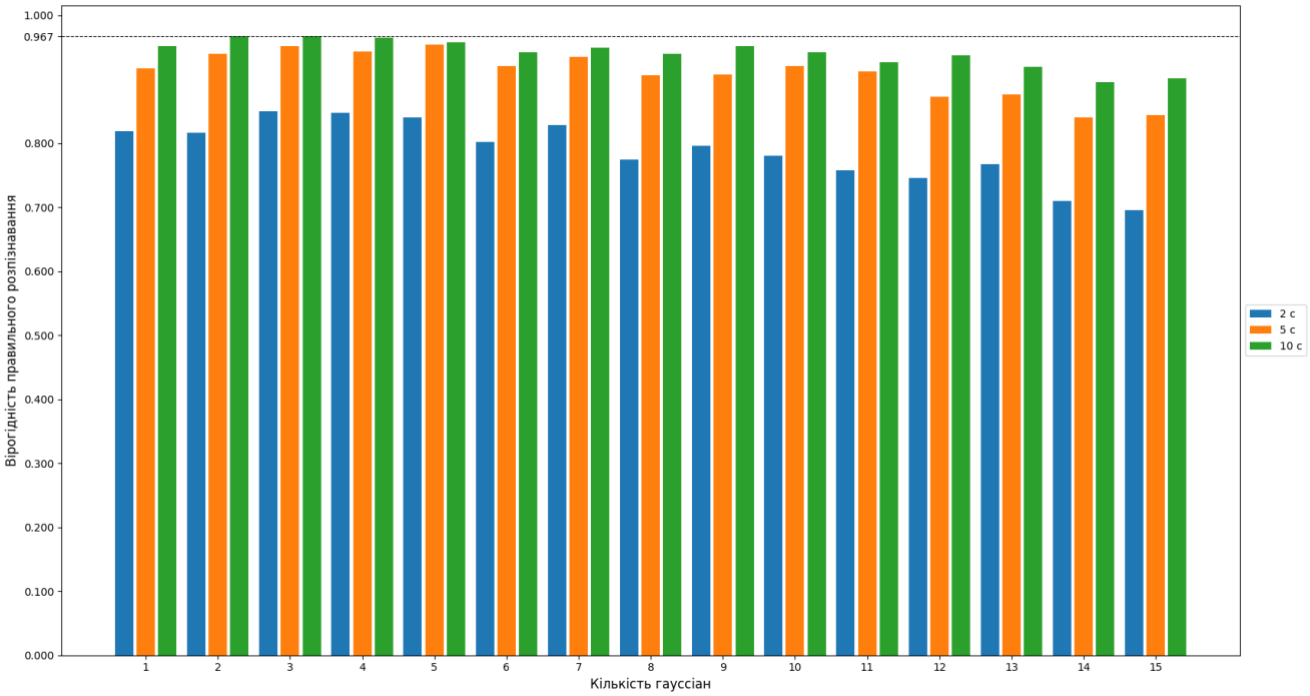

Рисунок 3.10 – Ймовірності правильного розпізнавання при використанні бази даних, побудованої на основі груп з 5-и аудіофрагментів тривалістю 2 секунди

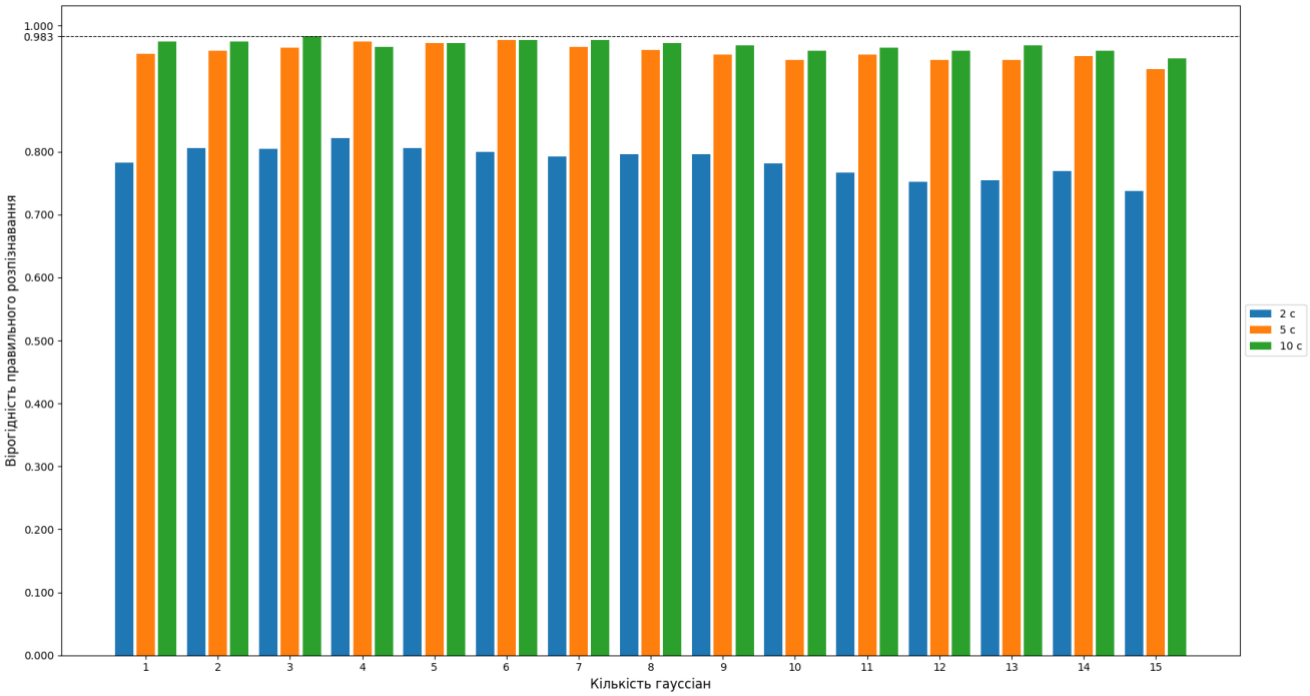

Рисунок 3.11 – Ймовірності правильного розпізнавання при використанні бази даних, побудованої на основі груп з 3-х аудіофрагментів тривалістю 5 секунд

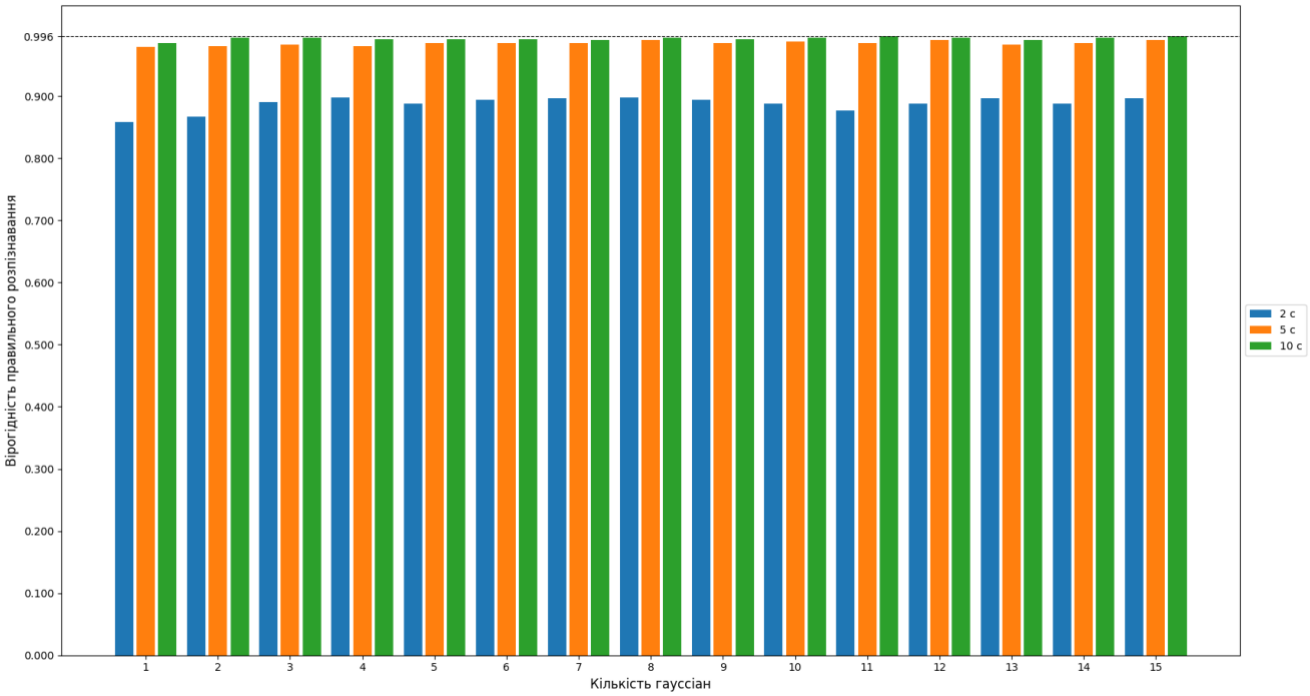

Рисунок 3.12 – Ймовірності правильного розпізнавання при використанні бази даних, побудованої на основі груп з 5-и аудіофрагментів тривалістю 5 секунд

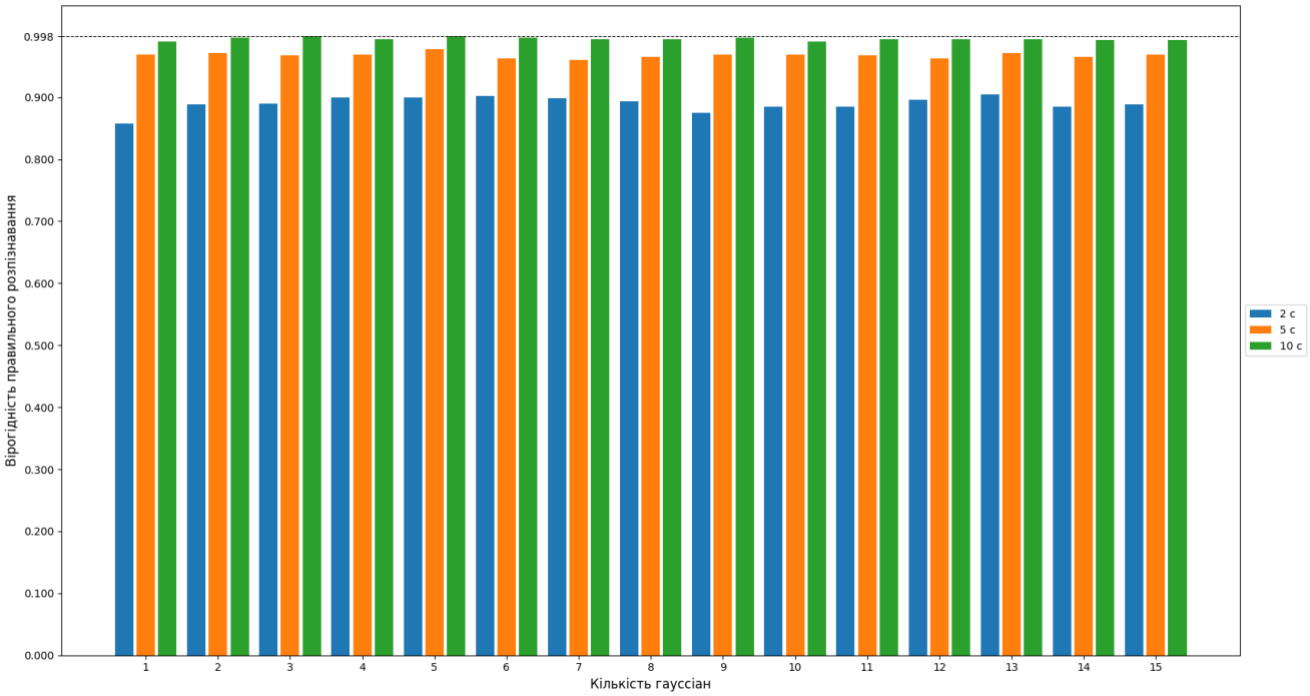

Рисунок 3.13 – Ймовірності правильного розпізнавання при використанні бази даних, побудованої на основі груп з 3-х аудіофрагментів тривалістю 10 секунд

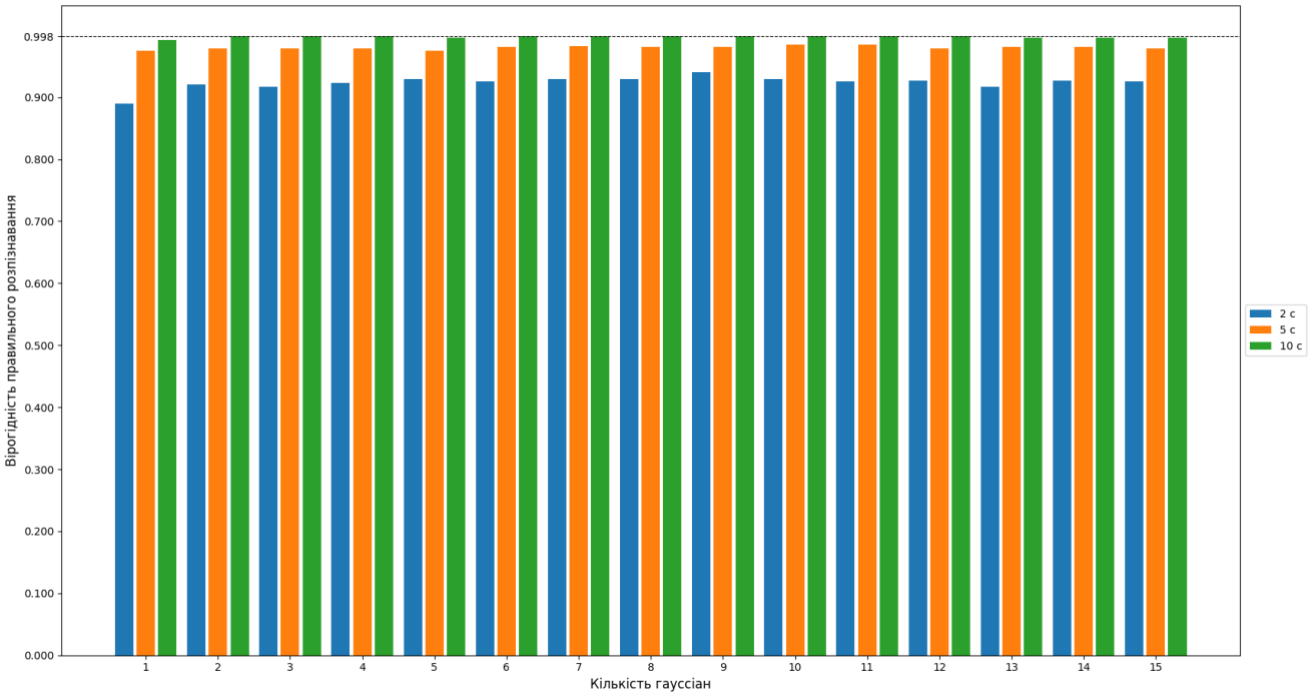

Рисунок 3.14 – Ймовірності правильного розпізнавання при використанні бази даних, побудованої на основі груп з 5-и аудіофрагментів тривалістю 10 секунд

Очевидно, що використання груп перевірочних аудіофрагментів тривалістю 10 секунд забезпечує отримання більшої ймовірності правильного розпізнавання. Найкращі результати для кожної з баз даних наведено нижче (Таблиця 3.1).

Таблиця 3.1 – Найкращі показники ймовірності правильного розпізнавання для кожної з баз даних

Тип бази даних	Кількість гауссіан	Тривалість перевірочного фрагменту, с	Ймовірність правильного розпізнавання
3 аудіофрагменти тривалістю 2 с.	1	10	0,89375
5 аудіофрагментів тривалістю 2 с.	3	10	0,96666
3 аудіофрагменти тривалістю 5 с.	3	10	0,98333

Продовження таблиці 3.1 – Найкращі показники ймовірності правильного розпізнавання для кожної з баз даних

Тип бази даних	Кількість гауссіан	Тривалість перевірочного фрагменту, с	Ймовірність правильного розпізнавання
5 аудіофрагментів тривалістю 5 с.	3	10	0,99583
3 аудіофрагменти тривалістю 10 с.	3	10	0,99791
5 аудіофрагментів тривалістю 10 с.	3	10	0,99791

Проаналізувавши ці дані та ті, що були отримані в попередньому розділі, можна стверджувати, що для того, щоб досягнути найкращих показників ефективності системи, слід використовувати аудіофрагменти тривалістю не менше ніж 10 секунд як на етапі навчання, так і на етапі верифікації осіб. Варто зауважити, що при використанні аудіофайлів меншої довжини швидкодія системи зросте, але відповідно зросте кількість хибних спрацювань, що є недопустимим для систем біометричної верифікації.

Відповідно до даних, наведених у таблиці, найбільша ймовірність правильного розпізнавання досягається при використанні груп перевірочних фрагментів тривалістю 10 секунд і баз даних, побудованих на групах з 3-х та 5-и аудіофрагментів тривалістю 10 секунд та кількості гауссіан 3. Так як значення вірогідності є однаковими в цих двох випадках, більш доцільним є використання бази даних, побудованої на групах з 3-х аудіофрагментів, оскільки це дасть змогу зекономити час і ресурси, необхідні для її побудови і при тому ніяк не вплинувши на ефективність роботи системи.

4 ПЕРЕВІРКА СТІЙКОСТІ РОЗРОБЛЕНОЇ СИСТЕМИ ДО ВИКОРИСТАННЯ ЗГЕНЕРОВАНОГО МОВЛЕННЯ

4.1 Тестування розробленого алгоритму з використанням згенерованого мовлення

У сучасну епоху досягнення в галузі штучного інтелекту та нейронних мереж призвели до значних результатів у створенні більш реалістичного типу синтезованого аудіо та мовлення. Однак ці досягнення також призвели до можливості зловживань цією технологією.

Технологія синтезу мовлення розвинулась з концепції простого конвертування тексту в звук до безпосереднього навчання та імітації голосу користувача. У минулому для генерації мовлення записувались часто вживані слова та звуки і потім озвучувався потрібний текст. Через недостатню природність та точність дослідники звертались до таких наук як фонетика, лінгвістика, статистика, а тепер і до технології штучного інтелекту [15]. Робота, опублікована у 2019 році, описує алгоритм клонування голосу, який спроможний генерувати реалістичні зразки мовлення довільної довжини при тому, що для навчання потрібний аудіофрагмент тривалістю хоча б 5 секунд [16]. Можливість зробити клон голосу є доволі перспективною і вже використовується, наприклад, для запису підкастів або генерації голосу померлих людей. Також ця технологія дає можливість синтезувати мовлення людини на іншій мові навіть без її знання [17]. Проте якщо існує можливість створення точної копії голосу людини, то очевидно, що це можна розцінювати як загрозу безпеки.

В наш час існують сервіси, які надають можливість створити клон власного голосу, щоб в подальшому мати змогу синтезувати мовлення з його допомогою. Наприклад, RESEMBLE.AI, Descript, BeyondWords, MURF.AI, Respeecher.

Для подальшого тестування поведінки системи при використанні згенерованого мовлення був використаний сервіс Descript, який є провідним у

галузі клонування голосів. Цей сервіс впроваджує етичні норми для запобігання несанкціонованого клонування голосу.

Descript використовує процес навчання мовних моделей, який залежить від перевірки усної згоди особи. Таким чином, користувачі можуть створювати тільки ті моделі перетворення тексту в мовлення, які були авторизовані власником голосу. Після створення голосу власник голосу має контроль над тим, коли і як він буде використовуватися [18]. Варто зауважити, що тема клонування голосу є активно досліджуваною, тому з’являється велика кількість подібних сервісів. На жаль, деякі з них не мають ніяких обмежень щодо можливості клонування чужого голосу, що може призвести до зловживання та використання цього функціоналу у неправомірних цілях.

Алгоритм клонування голосу у сервісі Descript є наступним:

1) Завантажити або записати аудіофайл тривалістю від 30 хвилин, який міститиме мовлення тільки однієї персони;
2) Дати усну згоду на обробку та клонування голосу;
3) Очікувати побудову моделі голосу, що може тривати від 2 до 24 годин.

Для проведення тестування розробленої системи було записано два аудіофрагменти мовлення різних осіб для клонування. Процес створення моделей у сервісі Descript зайняв приблизно 21 годину для кожного. Після цього сервіс дає можливість згенерувати довільний фрагмент мовлення на основі введеного тексту і обрати одну із готових моделей голосу (Рисунок 4.1). Наступним кроком для кожної особи було згенеровано аудіофайли довжиною 500 секунд і розділено на фрагменти тривалістю 10 секунд.

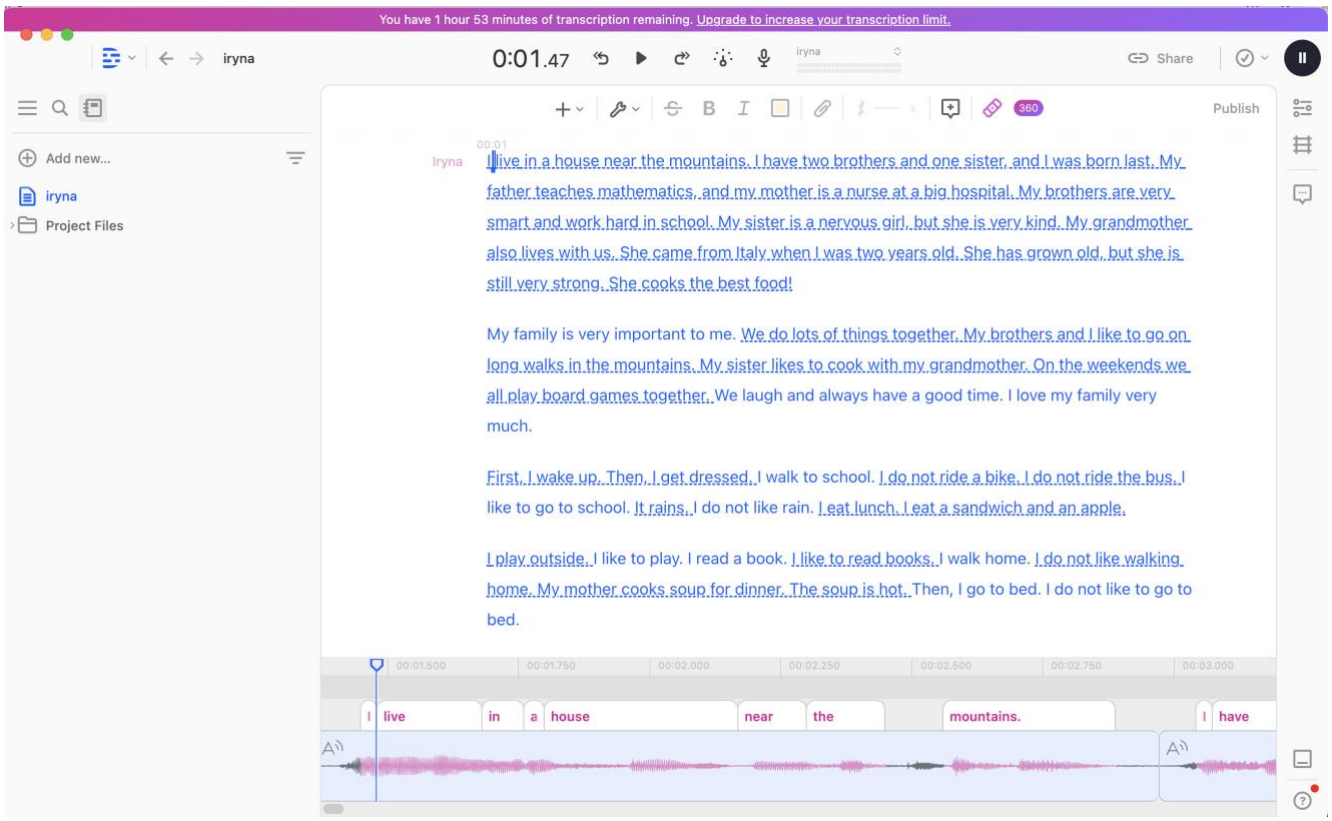

Рисунок 4.1 – Процес перетворення тексту у мовлення у сервісі Descript

Для порівняння характеристик оригінального та згенерованого мовлень були створені два однакові за змістом зразки: перший був записаний власником голосу, інший згенерований за допомогою готової моделі клонованого голосу. На слух фрагменти здаються доволі схожими. Як видно з даних, представлених нижче (Рисунок 4.2 та 4.3), ці аудіофрагменти мають схожі спектрограми та АЧХ, проте зразок згенерованого мовлення несе у собі більше інформації та деталей.

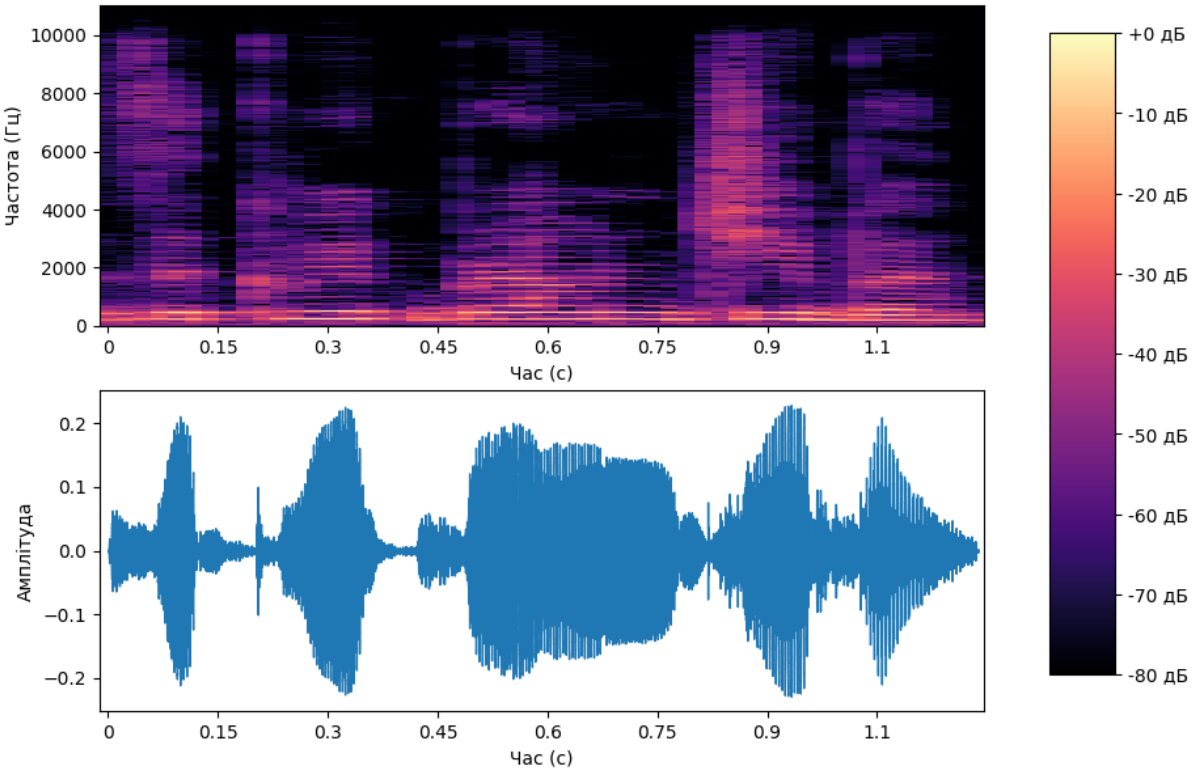

Рисунок 4.2 – Спектрограма та АЧХ запису оригінального голосу

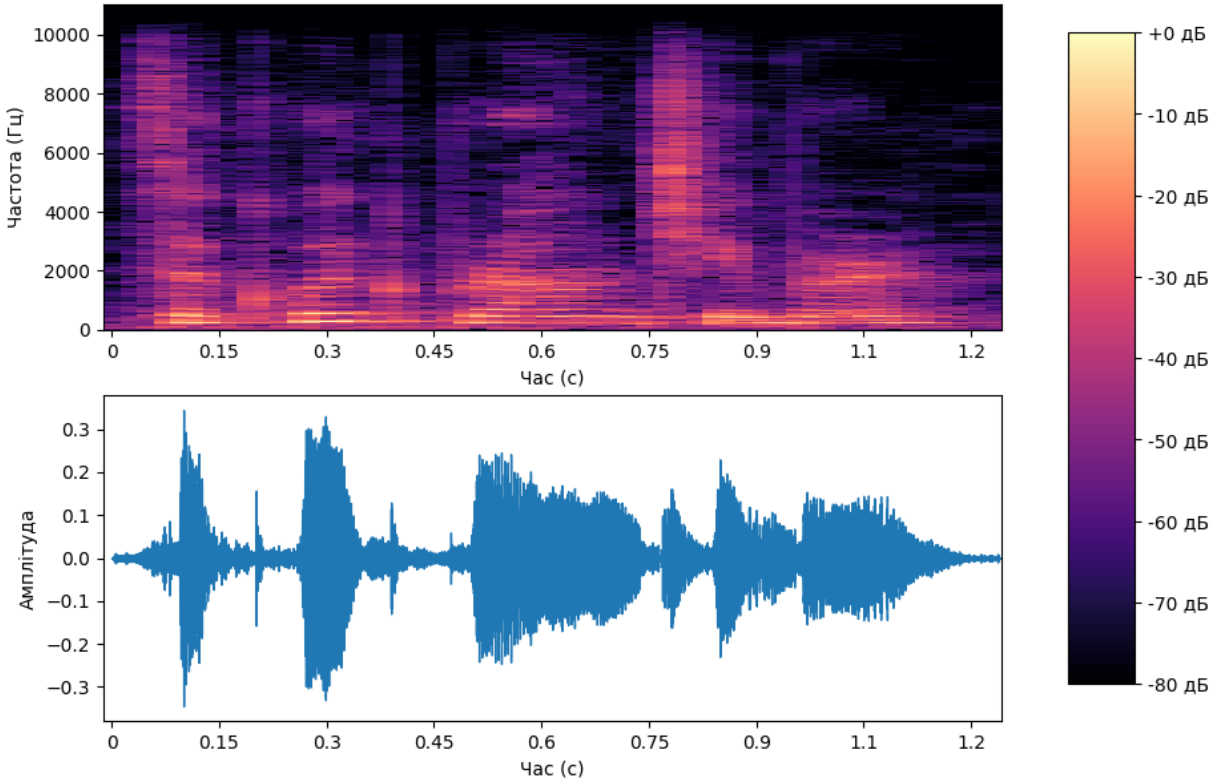

Рисунок 4.3 – Спектрограма та АЧХ згенерованого голосу

Основуючись на висновках із попереднього розділу для тестування було зібрано базу даних 100 осіб на основі груп з 3-х аудіофайлів тривалістю 10 секунд та з кількістю гауссіан 3. В цій базі даних також містились дані двох тестових осіб. Алгоритм тестування для кожної з осіб був наступним:

1) Взяти 50 аудіофрагментів тривалістю 10 секунд з оригінального запису мовлення особи;
2) Для кожного фрагменту отримати результат верифікації системою;
3) Розрахувати ймовірність правильного розпізнавання системою;
4) Взяти 50 аудіофрагментів тривалістю 10 секунд зі згенерованого мовлення сервісом Descript;
5) Для кожного фрагменту отримати результат верифікації системою;
6) Розрахувати ймовірність правильного розпізнавання системою;
7) Порівняти значення, отримані в пунктах 3 та 6.

Чисельні результати проведення тестування наведено нижче (Рисунок 4.2).

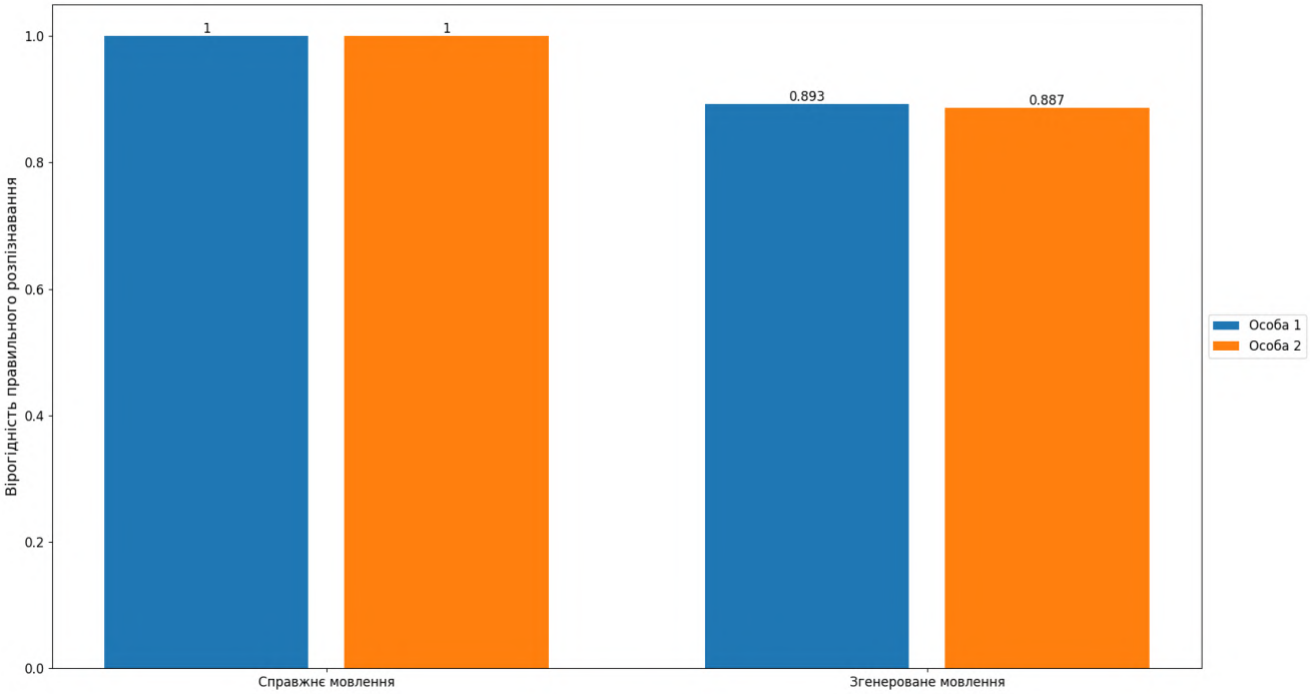

Рисунок 4.4 – Ймовірності правильного розпізнавання особи при використанні справжнього мовлення і згенерованого

Як видно з графіку, для кожної з осіб при використанні фрагментів справжнього мовлення система не дала жодного помилкового результату. У порівнянні зі справжнім мовленням використання згенерованого призводило до помилкових результатів.

Незважаючи на наявність помилкових результатів вірогідність правильного розпізнавання при умові використання згенерованого мовлення залишається доволі високою, що є доволі непередбаченим та вказує на рівень сучасних систем клонування голосу. На жаль, на цей момент ще не сформувались загальні методики та способи детектування згенерованого мовлення для подальшого відбракування. Існуючі методи є обмеженими та неуніверсальними. Більшість з них націлені на конкретну мову, різні довжини файлів, а також є доволі чутливими до шумів [19].

На жаль, вихідний код сервісу Descript не є відкритим і жодних описів методик та алгоритмів немає, тому це ще більше ускладнює розробку методів протидії такому виду атак та детектування згенерованого мовлення.

4.2 Методи детектування згенерованого мовлення

Незважаючи на те, що технології клонування голосу були впроваджені для допомоги людям, їх зловживання призвело до необхідності розробки методів детектування підробок. Люди можуть використовувати техніку підробки голосу для маніпулювання громадською думкою, наклепу або навіть тероризму. Відомі випадки, коли зловмисники, використовуючи подібне програмне забезпечення, видавали себе за інших осіб та, таким чином, виманювали великі суми грошей.

Всього можна виділити три категорії технологій аудіопідробок: ті, що базуються на імітації, синтетичні та відтворювальні. Голос може бути імітований, наприклад, за участі особи, яка має схожий до потрібного голос. Синтетичні (Text-To-Speech) технології мають на меті перетворити текст від прийнятне мовлення в реальному часі. Для генерації синтетичних підробок потрібно виконати два важливі кроки: потрібно зібрати чіткий структурований аудіоматеріал з транскриптом тексту та навчити модель з використанням зібраних даних для генерації звуку. Після навчання модуль аналізу тексту обробляє вхідний текст і як результат видає лінгвістичні характеристики. Далі акустичний модуль вилучає

параметри цільового диктора з набору даних залежно від попередньо отриманих лінгвістичних характеристик. Потім вокодер навчається створювати форми мовних хвиль на основі параметрів. Після цього відбувається генерація кінцевого аудіофайлу. Відтворювальні методи мають на меті відтворення запису голосу цільового спікера. Такі методи можуть використовуватись у текстово-залежній системі, оскільки можна записати речення, необхідне для перевірки.

З попереднього пункту очевидно, що сучасні технології клонування голосу є доволі успішними, чого, на жаль, не можна сказати про методи їх детектування. Важливо розглянути існуючі методи детектування згенерованого мовлення, щоб зрозуміти які з них є найбільш успішними, перспективними та придатними для використання в реальних умовах. Загалом існуючі методи можна поділити на дві категорії: методи машинного навчання та глибинного навчання.

Дослідження у напрямку машинного навчання мали суттєвий недолік – неможливість автоматизувати процес вилучення ознак та необхідність інтенсивної попередньої обробки. Оскільки це займає багато часу, дослідження були направлені на методи глибинного навчання.

Методи глибинного навчання хоч і дозволяють автоматизувати процес вилучення ознак та надмірного навчання, проте вимагають спеціальних перетворень для аудіоданих.

Характеристики деяких існуючих методів наведено нижче (Таблиця 4.1).

Таблиця 4.1 – Характеристики існуючих методів детектування згенерованого мовлення

Рік	Автори	Мова мовлення	Ознаки, що використовуються	Недоліки
2020	Wang та ін. [20]	Англійська, китайська	MFCC	Сильно залежить від оточуючих шумів
2020	Subramani, Rao [21]	Англійська	Спектрограма	Використовується підхід на основі зображень, який потребує спеціального перетворення аудіоданих в зображення
2020	Kumar-Singh, Singh [22]	Англійська	MFCC, MEL-спектрограма	Ознаки вилучаються вручну, що вимагає більше часу
2021	Borrelli та ін. [23]	Англійська	STLT	
2021	Liu та ін. [24]	Англійська	MFCC	
2020	Zhenchun Lei та ін. [25]	Англійська	CQCC, LFCC	Модель не є стійкою до різних ознак, працює тільки з LFCC
2021	S. Camanco та ін. [26]	Англійська	Точкові діаграми	Модель потребує додаткового навчання

Як видно з даних у таблиці, існуючі методи детектування клонованого мовлення не є універсальними та достатньо ефективними. Практично всі дослідження фокусуються на розробці методів детектування тільки при використанні англійської мови. Наприклад, немає досліджень, присвячених арабській мові, яка є четвертою за вживаністю у світі. Причина виділення саме арабської мови полягає в тому, що цей алфавіт має такі правила промови, з якими традиційні методи обробки звуків не можуть впоратись. Тому важливо розуміти, що навіть якщо буде створено успішний алгоритм для детектування англійського згенерованого мовлення, він може не бути таким продуктивним для інших мов.

Не дивлячись на те, що деякі методи можуть забезпечити високу точність визначення, на теперішній момент вони йдуть на компроміс між ефективністю та масштабованістю.

Важливою проблемою також є відсутність уваги акцентам. Дослідження з цього питання досі не проводяться, тому навіть незрозуміло чи впливає цей фактор на точність детектування. Необхідні подальші дослідження в цьому напрямку, оскільки при збільшенні кількості акцентів ймовірність того, що модель буде більш загальною та універсальною, збільшиться.

Залежність від шуму також є важливим питанням. Так як зазвичай у будь-якому аудіозаписі міститься шум, зловмисники можуть ввести його для обману детекторів. На жаль, проводилось лише одне дослідження з цього питання. У результаті не вдалось досягти потрібних результатів та уникнути впливу шумів [27].

Зрозуміло, що запропоновані методи вимагають спеціальної обробки даних для належної роботи. Класичні методи машинного навчання потребують великої кількості ручної праці для підготовки даних, тоді як глибинне навчання використовує методи на основі зображень. Такий підхід може вплинути на продуктивність методу. З усього вищесказаного можна зробити висновок, що цій області потрібна велика кількість нових досліджень для розробки метода, який зможе детектувати згенероване мовлення.

5 ПЕРСПЕКТИВИ ПОДАЛЬШОГО ВДОСКОНАЛЕННЯ РОЗРОБЛЕНОГО АЛГОРИТМУ

Ймовірнісні показники розробленого алгоритму дозволяють назвати його ефективним та придатним до використання в реальних умовах. Проте потрібно вказати деякі рекомендації та варіанти вдосконалення для забезпечення його стійкості.

Головна загроза при використанні голосової біометрії – це генерація голосу за допомогою сторонніх сервісів для несанкціонованого доступу. Зараз існує велика кількість методів детектування згенерованого голосу, проте всі вони не є достатньо ефективними, а лише вказують на майбутній напрямок дослідження. Так як точно визначити чи є голос згенерованим майже неможливо, можна ввести деякі додаткові вимоги при розпізнаванні, які дозволять мінімізувати можливість використання генерації голосу:

1) Використання випадкового паролю.

Під час розпізнавання можна попросити особу промовити певну послідовність слів, яка генерується на місці випадковим чином. У такому випадку розпізнавання особи проводиться не тільки за рахунок правильності її ознак, а й за відповідністю вхідного мовлення парольній фразі.

2) Встановлення часового ліміту для вимовляння парольної фрази.

Якщо при розпізнаванні встановити ліміт, впродовж якого повинний бути промовлений пароль, це ускладнить або навіть зробить неможливим генерацію мовлення, оскільки виділеного часу вистачить лише на промову встановленого паролю.

Особливість розробленої системи полягає в тому, що навіть у випадку, коли особи, яка намагається пройти ідентифікацію, немає в базі даних, система її розпізнає як того користувача, з яким голос найбільш подібний. Щоб уникнути такого сценарію, можна скористатись однією з наступних пропозицій:

1) Використовувати розроблену систему як додатковий етап верифікації особи після попередньої ідентифікації іншим методом;

2) Експериментальним шляхом розрахувати таке порогове значення подібності голосу під час розпізнавання, що за умови його недосягнення система не ідентифікує особу.

Варто зауважити, що перевірка працездатності розробленого алгоритму здійснювалась на базі даних зі 100 осіб. Це кількість працівників середнього за розміром підприємства. Тобто можна зробити висновок, що система разом із запропонованими вище покращеннями може використовуватись для керування доступом у середньому підприємстві.

Проте для розуміння універсальності алгоритму та можливості його використання на більших підприємствах потрібно провести тестування на більшій базі, що, у свою чергу, вимагає великої кількості ресурсів.

Окрім того, що розроблений алгоритм може бути частиною більш складної біометричної системи, він може самостійно використовуватись в таких випадках:

- Банківська система.

Запис голосу клієнта банку може використовуватись при його бажанні виконати певну операцію або під час входу до мобільного застосунку. Наприклад, зараз голосова біометрія використовується в ПриватБанку.

- Розумний дім.

Окрім того, що можна вмикати та вимикати певні пристрої, також можливо використовувати біометричне розпізнавання для керування доступом.

- Медицина.

Можливість ведення електронних карт пацієнтів.

З усього вищевказаного можна зробити висновок, що біометричні системи автоматичного розпізнавання по голосу є перспективними, але потребують більшої кількості досліджень задля забезпечення максимальної стійкості.

ВИСНОВКИ

1) У ході виконання роботи були проаналізовані основні теоретичні відомості з теми дослідження, а саме: текстово-залежні та текстово-незалежні системи автоматичного розпізнавання, різновиди ознак для вилучення, зменшення рівня стаціонарного та нестаціонарного шуму, алгоритм детектування активного мовлення, мел-частотні кепстральні коефіцієнти та модель суміші Гауса.

2) Було прийнято рішення про використання методу зменшення рівня стаціонарного шуму, так як при розробці системи робиться припущення, що під час запису голосів буде присутній тільки стаціонарний шум (наприклад, шум від мікрофону).

3) Проаналізовано алгоритм детектування активного мовлення WebRTC VAD з трьома рівнями агресивності. Для подальшої роботи був обраний режим агресивності 3, оскільки для реалізації системи не потрібна цілісність початкового мовлення.

4) Розроблена програмна імплементація алгоритму автоматичного розпізнавання довільного мовлення за допомогою мови програмування Python.

5) Для тестування алгоритму була зібрала база даних зі 100 осіб. Для кожного учасника було створено 6 груп аудіофайлів з 3-х та 5-и аудіофрагментів тривалістю 2 с, 5 с, 10 с. Проведений аналіз залежності часу, необхідного для побудови бази даних, від кількості гауссіан та тривалості аудіофайлів у групі.

6) Проведене тестування алгоритму для всіх створених баз даних та визначені значення ймовірностей правильного розпізнавання залежно від тривалості перевірочних фрагментів.

7) Результати тестування дозволяють назвати розроблену систему ефективною. Використання запропонованих значень параметрів системи дозволили отримати ймовірність правильного розпізнавання рівну 0,99791.

8) Проведений аналіз роботи системи при використанні згенерованого мовлення за допомогою сервісу Descript. Ймовірність правильного розпізнавання

при використанні згенерованого мовлення є високою, що говорить про необхідність створення засобів його детектування.

9) Проведений аналіз існуючих алгоритмів детектування згенерованого мовлення.

10) Наведено варіанти можливих досліджень та рекомендації щодо покращення алгоритму у майбутньому. Наприклад, використання випадково генерованої парольної фрази та встановлення часового ліміту.

11) Результати цієї роботи можуть стати частиною складних біометричних систем. Також запропонований алгоритм може використовуватись у таких напрямках, як: медицина, банківська система, при оподаткуванні та інше.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Pradeep. Ch, Text dependent speaker recognition using MFCC and LBG VQ, 2007. 61 p.
2. How Audacity Noise Reduction Works. URL: https://wiki.audacityteam.org/wiki/How_Audacity_Noise_Reduction_Works (дата звернення: 01.09.2022)
3. Tomi Kinnunen, Haizhou Li, An Overview of Text-Independent Speaker Recognition: from Features to Supervectors, North-Holland. 2009. 32 p.
4. Lostanlen, Vincent, Per-channel energy normalization: Why and how. *IEEE Signal Processing Letters 26.1*. 2018. P. 39-43.
5. Sainburg, Tim, Timothy Q. Gentner, Towards a computational neuroethology of vocal communication: from bioacoustics to neurophysiology, emerging tools and future direction. *Frontiers in Behavioral Neuroscience.* 2021. 330 p.
6. Voice Detection (VAD) Principles and Examples. URL: https://shichaog1.gitbooks.io/hand-book-of-speech-enhancement-and-recognition/content/chapter7.html (дата звернення: 10.09.2022)
7. WebRTC. URL: https://webrtc.org/ (дата звернення: 10.09.2022)
8. Mel Frequency Cepstral Coefficient: A Review / Shalbbya Ali and others. 2021. 10 p.
9. J.R. Deller, J.H. Hansen, J.G.Proakis, Discrete Time Processing of Speech Signals. *Prentice Hall*. 1993.
10. J. Benesty, M.M. Sondhi, Y.A. Huang, Hand book of Speech Processing. *Springer*, NewYork. 2008.
11. J.W. Picone, Signal modeling techniques in speech recognition. IEEE, 1993. P. 1215–1247.
12. Gaussian Mixture model. URL: https://brilliant.org/wiki/gaussian-mixture-model/ (дата звернення: 19.09.2022)

13. Gaussian Mixture Models: What are they & when to use? URL: https://vitalflux.com/gaussian-mixture-models-what-are-they-when-to-use/ (дата звернення: 30.09.2022)
14. Expectation–maximization algorithm. URL: https://en.wikipedia.org/wiki/Expectation%E2%80%93maximization_algorithm (дата звернення: 05.10.2022)
15. Multilingual Speech Synthesis for Voice Cloning. URL: https://ieeexplore.ieee.org/document/9373282 (дата звернення: 21.09.2022)
16. Transfer Learning from Speaker Verification to Multispeaker Text-To-Speech Synthesis / Ye Jia and others. Google Inc, 2019. 15 p.
17. Respeecher Calls on Celebrities to Speak Ukrainian Using AI Voice Generation Technology. URL: https://www.respeecher.com/blog/speak-ukrainian (дата звернення: 12.10.2022)
18. Descript Ethics Statement. URL: https://www.descript.com/ethics (дата звернення: 12.10.2022)
19. Zaynab Almutairi, Hebah Elgibreen, A Review of Modern Audio Deepfake Detection Methods: Challenges and Future Directions, Algorithms. 2022. 20 p.
20. Deepsonar: Towards effective and robust detection of ai-synthesized fake voices / Wang, R.; Juefei-Xu, F.; Huang, Y.; Guo, Q.; Xie, X.; Ma, L.; Liu, Y. New York, 2020. P. 1207–1216.
21. Subramani, N.; Rao, D. Learning efficient representations for fake speech detection. New York, 2020. P. 5859–5866.
22. Singh A.K., Singh P., Detection of ai-synthesized speech using cepstral & bispectral statistics. *In Proceedings of the 2021 IEEE 4th International Conference on Multimedia Information Processing and Retrieval (MIPR)*. Tokyo, 2021. P. 412–417.
23. Synthetic speech detection through short-term and long-term prediction traces / Borrelli and others. 2021.
24. Identification of fake stereo audio using SVM and CNN / Liu T. and others 2021. 12 p.

25. Siamese convolutional neural network using gaussian probability feature for spoofing speech detection / Lei Z. and others Shanghai, 2020; P. 1116–1120.
26. Fake speech recognition using deep learning / Camacho S. Cham, 2021. P. 38–48.
27. Deepsonar: Towards effective and robust detection of ai-synthesized fake voices / Wang R. and others New York, 2020. P. 1207–1216.

ДОДАТОК А

Вихідний код розробленої програмної імплементації

```
from enum import IntEnum, unique
from pydub import AudioSegment

_SECOND = 1000
_MINUTE = 60 * _SECOND


@unique
class Duration(IntEnum):
    TWO_SECONDS = _SECOND * 2
    FIVE_SECONDS = _SECOND * 5
    TEN_SECONDS = _SECOND * 10
    THIRTY_SECONDS = _SECOND * 30
    MINUTE = _MINUTE
    TWO_MINUTES = _MINUTE * 2

    def __str__(self):
        switch = {
            Duration.TWO_SECONDS: '2s',
            Duration.FIVE_SECONDS: '5s',
            Duration.TEN_SECONDS: '10s',
            Duration.THIRTY_SECONDS: '30s',
            Duration.MINUTE: '1m',
            Duration.TWO_MINUTES: '2m'
        }

        return switch[self]


@unique
class Chunks(IntEnum):
    ONE = 1
    TWO = 2
    THREE = 3
    FOUR = 4
    FIVE = 5
    SIX = 6
    SEVEN = 7
    EIGHT = 8
    NINE = 9
    TEN = 10
    FIFTY = 50
    HUNDRED = 100
    TWO_HUNDRED = 200


class Splitter:
    def __init__(self, duration: Duration, chunks: Chunks):
        self.duration = duration
        self.chunks = chunks

    def validate(self, audio: AudioSegment) -> None:
        if len(audio) < self.duration * self.chunks:
            raise AssertionError(
                'audio is short to process {} chunks of {} duration ({} ms)'.format(
                    self.chunks, self.duration, len(audio)
                )
            )

    def split(self, audio: AudioSegment) -> list[AudioSegment]:
        chunks = []

        for i in range(self.chunks):
            start = i * self.duration
            end = (i + 1) * self.duration
            chunk = audio[start:end]
            chunks.append(chunk)

        return chunks
```

Рисунок А.1 – Вихідний код класу Splitter для розділення аудіофайлу на фрагменти

```
from splitter import Splitter, Duration, Chunks

SPLITTERS_BY_ONE = [
    Splitter(duration=Duration.TWO_SECONDS, chunks=Chunks.ONE),
    Splitter(duration=Duration.FIVE_SECONDS, chunks=Chunks.ONE),
    Splitter(duration=Duration.TEN_SECONDS, chunks=Chunks.ONE),
    Splitter(duration=Duration.THIRTY_SECONDS, chunks=Chunks.ONE),
    Splitter(duration=Duration.MINUTE, chunks=Chunks.ONE),
]

SPLITTERS_BY_THREE = [
    Splitter(duration=Duration.TWO_SECONDS, chunks=Chunks.THREE),
    Splitter(duration=Duration.FIVE_SECONDS, chunks=Chunks.THREE),
    Splitter(duration=Duration.TEN_SECONDS, chunks=Chunks.THREE),
    Splitter(duration=Duration.THIRTY_SECONDS, chunks=Chunks.THREE),
    Splitter(duration=Duration.MINUTE, chunks=Chunks.THREE),
]

SPLITTERS_BY_FIVE = [
    Splitter(duration=Duration.TWO_SECONDS, chunks=Chunks.FIVE),
    Splitter(duration=Duration.FIVE_SECONDS, chunks=Chunks.FIVE),
    Splitter(duration=Duration.TEN_SECONDS, chunks=Chunks.FIVE),
    Splitter(duration=Duration.THIRTY_SECONDS, chunks=Chunks.FIVE),
    Splitter(duration=Duration.MINUTE, chunks=Chunks.FIVE),
]

SPLITTERS_BY_TEN = [
    Splitter(duration=Duration.TWO_SECONDS, chunks=Chunks.TEN),
    Splitter(duration=Duration.FIVE_SECONDS, chunks=Chunks.TEN),
    Splitter(duration=Duration.TEN_SECONDS, chunks=Chunks.TEN),
    Splitter(duration=Duration.THIRTY_SECONDS, chunks=Chunks.TEN),
    Splitter(duration=Duration.MINUTE, chunks=Chunks.TEN),
]

SPLITTERS_BY_FIFTY = [
    Splitter(duration=Duration.TWO_SECONDS, chunks=Chunks.FIFTY),
    Splitter(duration=Duration.FIVE_SECONDS, chunks=Chunks.FIFTY),
    Splitter(duration=Duration.TEN_SECONDS, chunks=Chunks.FIFTY),
]

SPLITTERS_BY_HUNDRED = [
    Splitter(duration=Duration.TWO_SECONDS, chunks=Chunks.HUNDRED),
    Splitter(duration=Duration.FIVE_SECONDS, chunks=Chunks.HUNDRED),
]
```

Рисунок А.2 – Конфігурація для розділення файлів

```
from pydub import AudioSegment
from os import walk
import time
import soundfile
import numpy as np
import noisereduce

MP3 = 'mp3'
WAV = 'wav'
FORMAT = MP3
OUT_FORMAT = WAV

INPUT_FOLDER = './in/mp3'
OUTPUT_FOLDER = './in/denoise'

STATIONARY = False


def denoise():
    files = []

    for (dir_path, dir_names, file_names) in walk(INPUT_FOLDER):
        files.extend(
            filter(lambda f: str(f).endswith(FORMAT), file_names)
        )
        break

    if len(files) == 0:
        print('No files')
        return

    for i, file in enumerate(files):
        file_name = str(file).replace('.' + FORMAT, '')
        file_path = '{}/{}'.format(INPUT_FOLDER, file)

        song = AudioSegment.from_mp3(file_path)
        y = song.get_array_of_samples()
        y = np.array(y)
        sr = song.frame_rate

        out = noisereduce.reduce_noise(y, sr, stationary=STATIONARY)

        path = '{}/{}.{}'.format(OUTPUT_FOLDER, file_name, OUT_FORMAT)
        soundfile.write(path, out, sr)

        print('Done with {} - {}'.format(i, path))


if __name__ == '__main__':
    start_time = time.time()
    denoise()
    print("--- %s seconds ---" % (time.time() - start_time))
```

Рисунок А.3 – Вихідний код методу зменшення рівня шуму аудіофайлів

```
from pydub import AudioSegment
from os import walk
import time
import soundfile
from pyvad import split
import numpy as np

MP3 = 'mp3'
WAV = 'wav'
FORMAT = MP3
OUT_FORMAT = WAV

INPUT_FOLDER = './in/mp3'
OUTPUT_FOLDER = './in/vad'


def vad():
    files = []

    for (dir_path, dir_names, file_names) in walk(INPUT_FOLDER):
        files.extend(
            filter(lambda f: str(f).endswith(FORMAT), file_names)
        )
        break

    if len(files) == 0:
        print('No files')
        return

    for i, file in enumerate(files):
        file_name = str(file).replace('.' + FORMAT, '')
        file_path = '{}/{}'.format(INPUT_FOLDER, file)

        song = AudioSegment.from_mp3(file_path)
        y = song.get_array_of_samples()
        y = np.array(y)
        sr = song.frame_rate

        edges = split(y, sr * 2, fs_vad=16000, hop_length=10, vad_mode=3)
        out = []
        for edge in edges:
            seg = y[edge[0]:edge[1]]
            out.extend(seg)
        out = np.array(out)

        path = '{}/{}.{}'.format(OUTPUT_FOLDER, file_name, OUT_FORMAT)
        soundfile.write(path, out, sr * 2)

        print('Done with {} - {}'.format(i, path))


if __name__ == '__main__':
    start_time = time.time()
    vad()
    print("--- %s seconds ---" % (time.time() - start_time))
```

Рисунок А.4 – Вихідний код методу детектування мовлення в аудіофайлах

```
from pydub import AudioSegment
from os import walk
from pathlib import Path
import time
from config import *

SPLITTERS = [
    *SPLITTERS_BY_ONE,
    *SPLITTERS_BY_THREE,
    *SPLITTERS_BY_FIVE,
    *SPLITTERS_BY_TEN,
    *SPLITTERS_BY_FIFTY,
    *SPLITTERS_BY_HUNDRED,
]

MP3 = 'mp3'
WAV = 'wav'
FORMAT = WAV
OUT_FORMAT = WAV

INPUT_FOLDER = './in/vad'
OUTPUT_FOLDER = './out'


def split():
    files = []

    for (dir_path, dir_names, file_names) in walk(INPUT_FOLDER):
        files.extend(
            filter(lambda f: str(f).endswith(FORMAT), file_names)
        )
        break

    if len(files) == 0:
        print('No files')
        return

    for file in files:
        file_name = str(file).replace('.' + FORMAT, '')
        file_path = '{}/{}'.format(INPUT_FOLDER, file)

        if FORMAT == MP3:
            song = AudioSegment.from_mp3(file_path)
        else:
            song = AudioSegment.from_wav(file_path)

        for splitter in SPLITTERS:
            dir_path = "{}/{}/{}/{}".format(OUTPUT_FOLDER, splitter.duration, splitter.chunks.value, file_name)
            Path(dir_path).mkdir(parents=True, exist_ok=True)

            splitter.validate(song)
            chunks = splitter.split(song)

            for i, chunk in enumerate(chunks):
                path = '{}/{}.{}'.format(dir_path, i, OUT_FORMAT)
                chunk.export(path, format=OUT_FORMAT)  # , bitrate=BITRATE)
                print('Done with {}'.format(path))


if __name__ == '__main__':
    start_time = time.time()
    split()
    print("--- %s seconds ---" % (time.time() - start_time))
```

Рисунок А.5 – Вихідний код для розділення аудіофайлів на фрагменти

```
import librosa
import numpy as np
from sklearn import preprocessing
from python_speech_features import mfcc


def extract_features(audio, rate):
    mfcc_features = mfcc(
        # The audio signal from which to compute features.
        audio,
        # The samplerate of the signal we are working with.
        rate,
        # The length of the analysis window in seconds.
        # Default is 0.025s (25 milliseconds)
        winlen=0.025,
        # The step between successive windows in seconds.
        # Default is 0.01s (10 milliseconds)
        winstep=0.01,
        # The number of cepstrum to return.
        # Default 13.
        numcep=13,
        # The number of filters in the filterbank.
        # Default is 26.
        nfilt=30,
        # The FFT size. Default is 512.
        nfft=1103,
        # If true, the zeroth cepstral coefficient is replaced
        # with the log of the total frame energy.
        appendEnergy=True)

    mfcc_features = preprocessing.scale(mfcc_features)
    mfcc_delta = librosa.feature.delta(mfcc_features)
    mfcc_delta2 = librosa.feature.delta(mfcc_features, order=2)
    combined = np.hstack((mfcc_features, mfcc_delta, mfcc_delta2))

    return combined
```

Рисунок А.6 – Вихідний код методу вилучення мел-частотних кепстральних коефіцієнтів

```
import os
import pickle
from sklearn.mixture import GaussianMixture
from main_functions import *
import time

DB_FOLDER = './db/10s3x1'
AUDIO_FOLDER = './out/10s/3'
GAUSSIANS_N = 1


def build_database():
    for user in os.listdir(AUDIO_FOLDER):
        user_path = os.path.join(AUDIO_FOLDER, user)

        features = np.array([])

        for audio_file in os.listdir(user_path):
            audio_path = os.path.join(user_path, audio_file)
            # reading audio files of speaker
            y, sr = librosa.load(audio_path)

            vector = extract_features(y, sr)

            if features.size == 0:
                features = vector
            else:
                features = np.vstack((features, vector))

        gmm = GaussianMixture(n_components=GAUSSIANS_N, max_iter=200, covariance_type='full', random_state=0)
        gmm.fit(features)

        # saving the trained gaussian model
        pickle.dump(
            gmm,
            open('{}/{}.gmm'.format(DB_FOLDER, user), 'wb')
        )

        print(user + ' added successfully')


if __name__ == '__main__':
    start_time = time.time()
    build_database()
    print("--- %s seconds ---" % (time.time() - start_time))
```

Рисунок А.7 – Вихідний код побудови бази даних з аудіофайлів

```
import os
import pickle
import time
from main_functions import *

DB_PATH = './db/10s3x3'
PATH = './out/10s/10/ann'
INDEX = 9


def recognize():
    gmm_files = [os.path.join(DB_PATH, fname) for fname in os.listdir(DB_PATH) if fname.endswith('.gmm')]
    models = [pickle.load(open(fname, 'rb')) for fname in gmm_files]
    speakers = [fname.split("/")[-1].split(".gmm")[0] for fname in gmm_files]

    if len(models) == 0:
        print("No Users in the Database!")
        return

    y, sr = librosa.load('{}/{}.wav'.format(PATH, INDEX))

    features = extract_features(y, sr)
    log_likelihood = np.zeros(len(models))

    start_time = time.time()

    # checking with each model one by one
    for i in range(len(models)):
        gmm = models[i]
        scores = np.array(gmm.score(features))
        log_likelihood[i] = scores.sum()

    pred = np.argmax(log_likelihood)

    print("Done in %s seconds" % (time.time() - start_time))

    identity = speakers[pred]

    print("Recognized as - ", identity)


if __name__ == '__main__':
    start_time = time.time()
    recognize()
    print("--- %s seconds ---" % (time.time() - start_time))
```

Рисунок А.8 – Вихідний код розпізнавання особи за аудіофайлом

ДОДАТОК В

Наукова публікація з теми дослідження

Рисунок В.1 – Сертифікат за участь у конференції