Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

ДОСЛІДЖЕННЯ ТА РЕАЛІЗАЦІЯ МЕТОДУ РОЗПІЗНАВАННЯ ГОЛОСОВИХ КОМАНД ЗА ДОПОМОГОЮ НЕЙРОМЕРЕЖІ
(тема)

Виконав:
студент 2 курсу, групи ІНФм-23-2
Касумов А.І.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Шафроненко А.Ю.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Касумову Артуру Імрановичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження та реалізація методу розпізнавання голосових команд за допомогою нейромережі

затверджена наказом по університету від 25 листопада 2024 року № 1246Ст

2. Термін подання студентом роботи до екзаменаційної комісії 1 січня 2025 р.

3. Вихідні дані до роботи аудіозапис, математичні моделі обробки аудіосигналу.

4. Перелік питань, що потрібно опрацювати в роботі

1. Проведення аналізу існуючих методів розпізнавання голосу.

2. Розробка алгоритма виділення ознак голосу на основі перетворення звукового сигналу в спектрограму.

3. Реалізування методу нейронних мереж для розпізнавання голосових команд з використанням отриманих спектральних образів.

4. Розробка комп'ютерної моделі для аналізу і розпізнавання голосу.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) актуальність проблеми обробки аудіосигналу, аналіз предметної області, постановка задачі, модель розпізнавання аудіосигналу, тестові зображення, аналіз отриманих результатів, висновки.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	25.11.2024	
2	Аналіз завдання, підбір літератури	26.11.24-28.11.24	
3	Аналіз літератури з досліджуваної проблеми	28.11.24-30.11.24	
4	Аналіз сучасних методі обробки аудіосигналу	01.12.24-05.12.24	
5	Розробка методу обробки аудіосигналу	06.12.24-10.12.24	
6	Програмна реалізація	11.12.24-21.12.24	
7	Оформлення пояснювальної записки	22.12.24-26.12.24	
8	Перевірка на плагіат	27.12.2024	
9	Рецензування	28.12.2024	
10	Підготовка презентації та доповіді	05.01.2025	
11	Занесення роботи в електронний архів	08.01.2025	
12	Попередній захист кваліфікаційної роботи	09.01.2025	

Дата видачі завдання 25 листопада 2024 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Шафроненко А.Ю.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 74 с., 3 табл., 26 рис., 42 джерела.

РОЗПІЗНАВАННЯ ГОЛОСОВИХ КОМАНД, НЕЙРОННІ МЕРЕЖІ, ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ, ГЛИБИННЕ НАВЧАННЯ, ОБРОБКА МОВЛЕННЕВИХ СИГНАЛІВ, КЛАСИФІКАЦІЯ.

Об’єктом дослідження є розпізнавання голосових команд за допомогою нейронних мереж.

Метою дослідження є розробка ефективного методу розпізнавання голосових команд з використанням MFCC для виділення ознак звуку. Це дозволяє вилучити ключові елементи з аудіосигналу та формувати вектори ознак, які потім використовуються для класифікації команд.

Використано методи обробки аудіосигналу та гібридного розпізнавання голосу. Проведено дослідження методу MFCC, який перетворює сигнал на компактний набір числових характеристик, що відображають його особливості, а також акустичної моделі, яка пов’язує вектори спостережень із фонемами.

У результаті дослідження здійснена програмна реалізація системи для розпізнавання голосових команд.

RECOGNITION OF VOICE COMMANDS, NEURAL NETWORK, CONVOLUTIONAL NEURAL NETWORK, DEEP LEARNING, PROCESSING OF WORD SIGNALS, CLASSIFICATION.

The object of the research is voice command recognition using neural networks.

The aim of the research is to develop an efficient method for recognizing voice commands using MFCC to extract sound features. This method allows extracting key elements from the audio signal and forming feature vectors that are then used for command classification.

Audio signal processing methods and hybrid voice recognition techniques were employed. The MFCC method was studied, which transforms the signal into a compact set of numerical characteristics reflecting its specific features, as well as the acoustic model that links observation vectors to phonemes.

As a result of the research, a software implementation of a voice command recognition system was developed.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

HMM – Hidden Markov Model (прихована модель Маркова)

MFCC – Mel Frequency Cepstral Coefficients (мел-кепстральні коефіцієнти)

NN – Neural Network (нейронна мережа)

RNN – Recurrent Neural Network (рекурентна нейронна мережа)

LSTM – Long Short-Term Memory (довга короткострокова пам’ять)

DCT – Discrete Cosine Transform (зворотне косинусне перетворення)

DTW – Dynamic Time Warping (динамічне вирівнювання часової шкали)

WER – Word Error Rate (коефіцієнт помилок у словах)

HTML – Hypertext Markup Language (мова розмітки гіпертексту)

CSS – Cascading Style Sheets (таблиця каскадних стилів)

ВСТУП

Розпізнавання мовлення – це технологія, яка дозволяє комп’ютеру або програмі перетворювати усне мовлення людини в текст або інші зрозумілі для системи команди.

Розпізнавання голосових команд є важливою частиною сучасних технологій взаємодії людини з комп’ютером. Воно застосовується в різних сферах, таких як мобільні пристрої, розумні будинки, автомобільні системи, а також у сферах медицини та освіти. Розпізнавання голосу дозволяє користувачам взаємодіяти з системами без використання клавіатур чи інших фізичних інтерфейсів, що значно підвищує зручність та швидкість виконання завдань.

Основною метою розпізнавання голосу є ідентифікація та класифікація сказаних команд, перетворення їх у зрозумілі для машини інструкції. Цей процес включає виділення ознак з аудіосигналу, його попередню обробку та класифікацію за допомогою алгоритмів машинного навчання, зокрема нейронних мереж. Нейронні мережі демонструють високі результати в обробці природної мови та аудіо через свою здатність вивчати складні залежності в даних та адаптуватися до різних акцентів, шуму та інших зовнішніх факторів.

Актуальність дослідження полягає у необхідності розробки ефективних методів та алгоритмів, які забезпечать точне і швидке розпізнавання голосових команд у реальному часі, що є ключовою вимогою для сучасних інтерактивних систем.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Актуальність задачі розпізнавання мови

Задача розпізнавання мови на сьогоднішній день є однією з найбільш актуальних в галузі інформаційних технологій та штучного інтелекту. Її важливість пояснюється швидким розвитком технологій автоматизації та зростаючим попитом на зручні і природні інтерфейси взаємодії між людиною та машиною. Розпізнавання голосових команд дозволяє значно підвищити ефективність і комфорт роботи з різними пристроями та сервісами.

Технології розпізнавання мови використовуються в найрізноманітніших сферах: від мобільних застосунків і голосових асистентів до систем управління автомобілями та побутовою технікою. Наприклад, розумні колонки, такі як Google Home або Amazon Alexa, забезпечують інтерактивний спосіб управління будинком через голосові команди. Голосові системи також застосовуються в автомобілях для управління навігацією, програванням музики або телефонними дзвінками.

Окремо варто відзначити важливість технологій розпізнавання мови для людей з обмеженими можливостями. Голосові команди можуть забезпечити більш доступну і зручну форму взаємодії з пристроями для людей з порушеннями зору, моторики або інших обмежень. Це підвищує якість життя і розширює можливості інтеграції таких людей у суспільство.

Системи розпізнавання голосових команд базуються на технологіях машинного навчання та штучного інтелекту. Удосконалення алгоритмів розпізнавання мови дозволяє підвищити точність і швидкість обробки голосових команд, що робить ці системи більш надійними і зручними для користувачів. Технології на основі нейронних мереж здатні розуміти не лише окремі слова, але й контекст, що суттєво покращує взаємодію між користувачем та пристроєм [1].

Ще одним викликом в області розпізнавання мови є адаптація систем до різних мов і діалектів. Важливо, щоб технології могли ефективно працювати з різними акцентами, інтонаціями і навіть різними мовними особливостями. Це дозволяє підвищити доступність технологій на глобальному рівні і забезпечує їхню інтеграцію в різних країнах та культурах.

Задача розпізнавання мови є надзвичайно актуальною в сучасному світі, оскільки вона відкриває нові горизонти для розвитку технологій взаємодії між людиною і машиною. Використання голосових команд спрощує взаємодію з пристроями, робить їх доступнішими для всіх категорій населення, а також стимулює подальший розвиток штучного інтелекту. Тому розробка ефективних систем розпізнавання мови є важливим завданням для інноваційних проєктів.

1.1.1 Історія розвитку систем розпізнавання мови

Розробки в галузі розпізнавання мовлення почалися понад півстоліття тому. Однією з перших успішних систем була «Одрі», створена в Bell Laboratories. Ця система могла з точністю до дев’яносто відсотків розпізнавати цифри. З роками застосування технологій розширювалося, а кількість розпізнаних слів зростала. Спочатку системи використовували лише звукові елементи (фонеми), тому користувачам потрібно було говорити повільно та робити паузи, щоб пристрій правильно інтерпретував звуковий потік.

Згодом інженери додали до акустичного підходу лінгвістичний компонент, завдяки чому системи почали застосовувати мовні правила. Це дозволяло системам робити припущення щодо незрозумілих слів, спираючись на семантичні та синтаксичні правила.

У 1997 році був представлений перший розпізнавач безперервної мови – Dragon Dictate. Він міг розпізнавати до 100 слів на хвилину без необхідності робити паузи між словами.

Справжній прорив у цій галузі відбувся завдяки машинному навчанню. З появою глибинного навчання точність розпізнавання значно покращилася. У 2010 році Google об’єднав алгоритми машинного навчання з хмарними технологіями, створивши Google Voice Search, який використовував персоналізацію для підвищення точності. Ці напрацювання лягли в основу Google Assistant, встановленого на більш ніж половині смартфонів світу.

У 2011 році Apple випустила свого голосового асистента Siri, який зміг точно інтерпретувати природну мову. Успіх Siri стимулював розвиток інших голосових асистентів від Amazon, Microsoft та інших компаній.

1.1.2 Приклади використання систем розпізнавання мови

Системи розпізнавання мови знайшли широке застосування у різних сферах, значно спрощуючи взаємодію людини з технікою. Ось кілька прикладів їхнього використання:

– голосові асистенти. Вони дозволяють користувачам давати команди для пошуку інформації в інтернеті, керувати пристроями, надсилати повідомлення або нагадування, а також взаємодіяти з застосунками за допомогою голосу;

– розумні будинки. У розумних будинках системи розпізнавання мови використовуються для управління побутовою технікою, освітленням, опаленням або системами безпеки. Наприклад, за допомогою голосу можна ввімкнути світло, змінити температуру або керувати камерами спостереження;

– автомобільні системи. Сучасні автомобілі обладнані системами голосового управління, які дозволяють водіям здійснювати телефонні

дзвінки, керувати навігацією, включати музику чи навіть налаштовувати клімат-контроль без відволікання від дороги. Це підвищує безпеку під час водіння;

– медичні та освітні програми. У медицині системи розпізнавання мови допомагають лікарям швидко диктувати записи про пацієнтів, а в освіті – використовуються для створення текстових матеріалів на основі лекцій або диктантів. Це підвищує ефективність роботи та зменшує час, витрачений на ручний запис;

– підтримка для людей з обмеженими можливостями. Системи розпізнавання мови роблять технології доступнішими для людей з порушеннями зору або моторики, дозволяючи їм керувати пристроями і системами за допомогою голосу.

Таким чином, системи розпізнавання мови стають важливим інструментом для полегшення щоденних задач і підвищення зручності в різних аспектах життя [2].

1.2 Існуючі підходи до вирішення задачі розпізнавання мови

1.2.1 Класифікація систем розпізнавання мови

Системи розпізнавання мови можна поділити за наступними критеріями:

– за типом входу. Системи розпізнавання мовлення можуть класифікуватися за типом вхідних мовленнєвих даних.

Перший тип це розпізнавання континуального мовлення. Це системи, які здатні обробляти мовлення в реальному часі без явних пауз між словами. Такий тип розпізнавання використовують, наприклад, у системах диктування або голосових асистентах. Основна складність полягає в тому, що слова можуть бути «злиті» одне з одним, і система повинна мати потужні алгоритми для їхнього правильного розподілу і обробки. Наприклад, якщо

людина говорить швидко, без пауз, система повинна не тільки ідентифікувати кожне слово, але й враховувати контекст для кращого розуміння. Серед переваг цього типу можна відокремити наступне: підходить для реального використання в природному діалозі та підтримує високий рівень автоматизації, дозволяючи системам розпізнавати безперервний потік слів, але має також свої недоліки: висока обчислювальна складність та потреба в потужних мовних і акустичних моделях.

Другий тип це розпізнавання ізольованих слів. Система розпізнає кожне слово або висловлювання окремо. Мовленнєві паузи між словами дозволяють системі чітко визначати початок і кінець кожного слова. Такі системи простіші у реалізації, оскільки немає потреби аналізувати контекст і злиття слів. Вони зазвичай використовуються в системах команд, де кожне слово є окремою командою, наприклад, «відкрити», «закрити» тощо. Серед переваг можна виділити простоту у реалізації і навчанні та високу точність розпізнавання ізольованих слів. Але має свої недоліки, а саме: не підходить для природного діалогу та користувачеві потрібно говорити з явними паузами між словами;

– за обсягом лексики. Цей параметр стосується кількості слів, яку система здатна розпізнавати. Чим більша лексика системи, тим більше можливостей вона має для виконання складніших задач.

Існують системи з малою лексикою, які містять від кількох до кількох сотень слів. Зазвичай використовуються в спеціалізованих застосунках, наприклад, для управління пристроями або для розпізнавання голосових команд у системах контролю. Прикладом можуть бути голосові команди в смарт-пристроях типу «включити світло», «зупинити музику» тощо. Цей підхід дає висока точність при малому наборі слів та прост у реалізації і потребує невеликих ресурсів для обробки, але має обмежені можливості в складних сценаріях і не підходять для застосування в діалогових системах.

Також бувають системи з великою лексикою. Містять тисячі і більше слів. Використовуються в системах загального призначення, таких як

голосові асистенти (наприклад, Google Assistant, Siri). Такі системи повинні мати більш складні мовні моделі для коректного розпізнавання слів у різних контекстах. Цей підхід дає можливість працювати з великим числом запитів та підтримує складні діалоги і широкий спектру команд, але потребує більших ресурсах для обробки;

– за залежністю від мовця. Системи розпізнавання мовлення можуть класифікуватися за тим, наскільки вони чутливі до особливостей голосу конкретного користувача.

Деякі системи залежні від мовця, бо вони розробляються для конкретного користувача. Такі системи зазвичай потребують попереднього налаштування або навчання для кожного диктора, що дозволяє адаптувати систему під конкретні фізіологічні характеристики голосу. Вони використовуються, наприклад, у деяких системах безпеки або в індивідуальних голосових асистентах. Серед переваг можна виділити високу точність для конкретного диктора та простоту у навчанні. Але такі системи неможливо використовувати з іншими користувачами без додаткового налаштування.

Також існують незалежні від мовця системи, які можуть розпізнавати мовлення будь-якої людини без попередньої адаптації. Вони широко використовуються в застосунках загального призначення, таких як системи автоматичної відповіді, голосові асистенти, інтерактивні системи. Ці системи підходять для широкого використання, але їх точність нижча порівняно з залежними від мовця системами та розробка таких систем складніша і дорожча;

– за мовною моделлю. Мовна модель визначає, як система аналізує мовленнєві одиниці (фонеми) для їх подальшого розпізнавання.

Монофонічні моделі працюють з окремими фонемами (найменшими одиницями звукової мови) без врахування їхнього контексту. Вони є простішими в реалізації, але можуть допускати більше помилок, оскільки не враховують вплив сусідніх звуків. Серед переваг можна зазначити, що у них

менша складність реалізації, а також краще підходять для простих застосунків із чіткими мовленнєвими сигналами. Але через це такі системи менш точні, оскільки не враховується контекст фонем та вразливі до шуму і спотворень.

Трифонічні моделі враховують не лише кожну фонему, але й її оточення, тобто сусідні фонеми, що значно покращує точність розпізнавання. Це дозволяє системі краще розуміти звуки у складних контекстах. У цих моделей підвищена точність і стійкість до варіацій мовлення та краще розпізнавання в умовах шуму, але більш складні алгоритми обробки та вища потреба в більших обчислювальних ресурсах.

1.2.2 Існуючі підходи для розпізнавання мови

У сучасній науці та індустрії використовуються різні підходи до вирішення задачі розпізнавання мови. Вони еволюціонували від простих алгоритмів до складних моделей, заснованих на машинному навчанні та нейронних мережах. Основні підходи до розпізнавання мови можна поділити на кілька категорій [3]:

– акустико-лінгвістичні моделі. Цей підхід базується на використанні прихованих марковських моделей (HMM) для обробки мовного сигналу. Акустичні моделі поєднують фонетичну інформацію зі звукового потоку з лінгвістичними правилами, що дозволяє програмі будувати імовірнісні припущення про те, яке слово було сказане. Лінгвістичний компонент враховує граматичні та синтаксичні правила мови, що дозволяє системі «вгадувати» слова навіть за наявності шуму або неточностей у вимові;

– метод на основі шаблонів. Цей підхід порівнює вхідні звукові сигнали зі збереженими шаблонами мовних одиниць (наприклад, слів або фонем). Метод використовувався на ранніх етапах розвитку технології розпізнавання мови, оскільки він потребував лише мінімального

обчислювального ресурсу. Проте його ефективність значно обмежується великою варіативністю вимови та умов запису звуку;

– нейронні мережі та глибоке навчання. Сьогодні нейронні мережі, особливо глибокі нейронні мережі (DNN) та рекурентні нейронні мережі (RNN), є основою сучасних систем розпізнавання мови. Ці моделі використовують багатошарові мережі, що здатні навчатися на величезних масивах даних. Вони не тільки вивчають акустичні властивості мовлення, але й можуть моделювати контекст мовлення, що дозволяє розпізнавати навіть складні фрази з високою точністю. Архітектури на основі трансформерів, такі як BERT і GPT, дозволили значно покращити розуміння природної мови. Вони застосовуються в сучасних голосових асистентах (Siri, Google Assistant) і здатні обробляти як окремі команди, так і тривалі діалоги;

– моделі на основі end-to-end навчання. Технології end-to-end навчання, такі як рекурентні нейронні мережі (RNN) або згорткові нейронні мережі (CNN), пропонують підхід, де система вчиться перетворювати мовні сигнали безпосередньо в текст без розбивки на окремі етапи (акустичний, лінгвістичний). Це спрощує процес і зменшує кількість помилок на кожному з етапів. Важливою перевагою є те, що ці системи навчаються з даних автоматично, що дозволяє їх адаптувати до різних мов та акцентів;

– комбіновані підходи. У сучасних системах часто використовуються комбіновані підходи, що об’єднують кілька методів для підвищення точності. Наприклад, нейронні мережі можуть використовуватися для попередньої обробки мовних сигналів, а потім застосовуються алгоритми на основі прихованих марковських моделей для детальнішого аналізу.

1.2.3 Основні проблеми розпізнавання мовлення

За останнє десятиліття в розвитку розпізнавання відбувся різкий ривок вгору. Але тим не менш, машина ще не досягла тієї точності розпізнавання,

яку має звичайна людина. Існує декілька складнощів, над якими ще треба працювати та добиватись покращень [4].

Системи часто мають труднощі з розумінням людей із різними акцентами або діалектами. Мовні моделі, які навчаються на стандартній вимові, можуть погано працювати для регіональних варіантів мови, що знижує точність розпізнавання.

Розпізнавання мовлення в умовах шуму (наприклад, на вулиці або в натовпі) значно ускладнюється. Фонові звуки, такі як музика або розмови інших людей, можуть спотворювати вхідний сигнал, що призводить до помилок.

Інша проблема – це семантичні помилки, коли система неправильно розуміє значення фрази. За даними досліджень Microsoft, багато помилок машин схожі на людські, але люди використовують контекст, тему розмови, візуальні підказки (міміку, жести), щоб правильно інтерпретувати слова.

Також важливими залишаються такі завдання, як розподіл багатоголосого потоку, стійкість до різних форматів запису та шуму, зменшення затримки між промовою та транскрипцією, а також врахування контексту у процесі розпізнавання [5].

1.3 Аналіз літературних джерел щодо апробації результатів застосування методів розпізнавання мови

Для кращого рівня обізнаності даної теми було проведено дослідження наукового матеріалу. У цьому підрозділі наведено результати дослідження літературних джерел.

У джерелі [6] розглянуто основи застосування нейронних мереж для розпізнавання мовлення, зокрема рекурентних нейронних мереж (RNN), які спеціально розроблені для обробки послідовностей даних. Автори звертають особливу увагу на вдосконалені варіанти RNN, такі як LSTM (Long Short-

Term Memory), які здатні ефективно зберігати інформацію про попередні стани у довгих послідовностях. Це робить LSTM ідеальними для обробки мовлення, оскільки голосові сигнали мають сильну залежність від попереднього контексту. Основною метою дослідження є підвищення точності розпізнавання голосу через оптимізацію архітектури нейромереж та покращення методів обробки аудіосигналів.

У статті детально описується, як LSTM вирішує проблему «забування» попередньої інформації, властиву класичним RNN, і яким чином це підвищує ефективність розпізнавання мовлення. Використання таких моделей дозволяє системі правильно інтерпретувати мовленнєві команди навіть у випадках, коли вони супроводжуються шумом чи акцентами.

У джерелі [7] обговорюються можливості застосування моделей типу трансформерів у розпізнаванні голосу. Трансформери, на відміну від RNN, не залежать від попередніх станів послідовності, а використовують механізм уваги для одночасного аналізу всієї послідовності. Це дозволяє моделі враховувати різні частини вхідного сигналу незалежно від їх позиції у часі. У цьому дослідженні автори підкреслюють важливість використання трансформерів у завданнях, пов’язаних з великими обсягами даних, оскільки такі моделі не лише підвищують точність розпізнавання, але й значно зменшують час обробки.

Дослідження демонструє, що трансформери не просто здатні швидко обробляти голосові команди, але також ефективно розпізнають складні мовні структури завдяки паралельній обробці інформації, що дає їм перевагу перед класичними RNN.

У джерелі [8] розглядаються методи попередньої обробки звукових сигналів, зокрема виділення мовних ознак через спектрограму мел-частот (Mel-spectrogram). Автори стверджують, що спектрограми мел-частот дозволяють краще зберігати важливу інформацію про частотні складові голосу, які є критичними для точного розпізнавання. Спектрограми

конвертують аудіосигнал у візуальні ознаки, які можуть бути подані як вхідні дані до нейронних мереж.

Дослідження підкреслює, що застосування таких методів перед обробкою сигналів дозволяє нейронним мережам краще аналізувати звукові дані і виділяти ключові особливості, що суттєво підвищує точність розпізнавання мовлення, особливо у складних акустичних умовах.

У джерелі [9] представлено модель Deep Speech, яка використовує підхід end-to-end, що означає пряме перетворення звукових сигналів у текст без потреби у складних попередніх алгоритмах обробки. Глибинні нейронні мережі (DNN) навчаються розпізнавати мову безпосередньо на великих наборах даних. Основний акцент дослідження зроблено на масштабованості моделі та її здатності працювати з великими обсягами даних.

Автори демонструють, що модель Deep Speech дозволяє значно спростити процес розпізнавання мовлення, оскільки вона не потребує ручної інженерії ознак і може навчатися безпосередньо на сирих аудіоданих, що робить систему більш гнучкою і придатною для різних умов мовлення.

У джерелі [10] розглянуто вдосконалений варіант Deep Speech – Deep Speech 2, який здатен розпізнавати мовлення на декількох мовах, включаючи англійську та китайську. Дослідники показали, що використання такої моделі з новими техніками обробки аудіосигналів дозволяє досягти вищої точності в умовах різноманітних акустичних середовищ.

Модель Deep Speech 2 підкреслює можливість створення універсальних систем розпізнавання мовлення, здатних працювати в умовах різної якості запису, з різними мовами та діалектами. Це досягається завдяки покращенню обробки акустичних ознак та адаптації моделі до різних мовних контекстів.

У джерелі [11] досліджується застосування моделей з механізмом уваги для розпізнавання мовлення. Автори акцентують увагу на тому, що такі моделі можуть динамічно зосереджуватись на певних частинах аудіо під час розпізнавання, що дозволяє підвищити точність особливо у складних умовах (наприклад, у шумі чи при нерівномірній тривалості сигналів).

Механізм уваги забезпечує можливість фокусування на важливих частинах сигналу, що підвищує якість обробки складних мовних структур і дозволяє досягти високої точності розпізнавання у реальних умовах.

У джерелі [12] досліджено використання двонаправлених LSTM (BiLSTM) для класифікації фонем у мовленні. Автори показують, що двонаправлені моделі здатні одночасно враховувати як попередні, так і наступні стани сигналу, що дозволяє точніше аналізувати мовні команди, зберігаючи контекст на всіх етапах аналізу.

Двонаправлені LSTM дозволяють покращити якість розпізнавання мови завдяки врахуванню повного контексту, що є особливо корисним у випадках, коли поточний звук залежить від того, що було сказано раніше або що буде сказано далі.

У джерелі [13] проведено детальний аналіз роботи чотирьох дослідницьких груп, які працюють над використанням глибинних нейронних мереж (DNN) для акустичного моделювання у системах розпізнавання мовлення. Дослідження демонструє, що DNN значно перевершують традиційні моделі, такі як приховані марковські моделі (HMM) і гаусові суміші (GMM), які довгий час використовувалися в розпізнаванні мовлення. Глибинні нейронні мережі відкрили нові можливості для покращення якості розпізнавання голосу та розширили можливості обробки звукових сигналів.

У джерелі [14] обговорюється застосування sequence-to-sequence моделей, що є інноваційними підходами до розпізнавання мовлення. Sequence-to-sequence моделі (Seq2Seq) використовують нейронні мережі для прямого перетворення мовних сигналів у текстові дані, що дозволяє значно спростити процес обробки і покращити точність. Основна перевага цих моделей полягає в тому, що вони здатні працювати із цілими послідовностями даних, не розбиваючи мовлення на окремі елементи, як це робилося раніше у традиційних моделях.

У джерелі [15] описано систему розпізнавання мовлення, розроблену компанією Microsoft у 2017 році. Ця система поєднує новітні досягнення в

галузі нейронних мереж та машинного навчання для вирішення завдань розпізнавання мовлення, особливо в реальних умовах. Основна увага приділяється розпізнаванню розмовного мовлення, яке, на відміну від формального, має багато особливостей, таких як фонові шуми, нерівномірні паузи, швидка зміна темпу мовлення та інтонації.

1.4 Постановка задачі дослідження

Сучасні технології розпізнавання голосу є важливим напрямком в області обробки та аналізу мовних сигналів. Точне і швидке розпізнавання голосових команд на основі індивідуальних особливостей голосу потребує застосування сучасних методів машинного навчання і нейронних мереж. Тому постає завдання розробки алгоритмів, здатних ідентифікувати голосові команди [16].

Об’єктом дослідження є розпізнавання голосових команд за допомогою нейронних мереж.

Метою дослідження є розробка ефективного методу розпізнавання голосових команд з використанням MFCC для виділення ознак звуку. Це дозволяє вилучити ключові елементи з аудіосигналу та формувати вектори ознак, які потім використовуються для класифікації команд.

Для досягнення мети необхідно вирішити такі завдання:

– провести аналіз існуючих методів розпізнавання голосу на основі спектральних ознак;

– розробити алгоритм виділення ознак голосу на основі перетворення звукового сигналу в спектрограму;

– реалізувати метод нейронних мереж для розпізнавання голосових команд з використанням отриманих спектральних образів;

– розробити комп’ютерну модель для аналізу і розпізнавання голосу на основі індивідуальних ознак.

2 МАТЕМАТИЧНІ МОДЕЛІ РОЗПІЗНАВАННЯ ГОЛОСОВИХ КОМАНД

2.1 Запис і цифрування звука

Мовний сигнал є неперервним аналоговим сигналом, який потрібно оцифрувати для подальшої обробки. З даними такого типу система розпізнавання працювати не може. Тому перед безпосередньою обробкою звуку, його необхідно перевести у послідовність цифрових сигналів. Це робиться шляхом оцифрування – процесу, який розбиває мовний сигнал на маленькі частини (так звані відліки), кожна з яких має своє числове значення. Частота дискретизації – це те, наскільки часто беруться ці частини сигналу. Чим вища частота, тим більше деталей голосу зберігається.

Потім, щоб аналізувати цей цифровий сигнал, використовується математичний метод під назвою дискретне перетворення Фур'є (2.1). Перетворення Фур'є перетворює цифровий звуковий сигнал з його часової форми (де видно, як змінюється гучність або амплітуда сигналу з плином часу) в частотну форму, яка показує, з яких частот складається цей сигнал. Перетворення Фур'є «розкладає» цю звукову хвилю на складові частоти. Будь-який звук можна описати як набір хвиль різної частоти і гучності. Перетворення Фур'є дозволяє побачити, які частоти присутні в цьому звуці і з якою гучністю [17].

В результаті, замість звичайного звукового сигналу в часі, отримуємо частотний спектр – це графік, на якому по одній осі вказані частоти (низькі, середні, високі), а по іншій – їх амплітуди (наскільки сильно представлена кожна частота в сигналі).

Після перетворення сигналу в частотний вигляд його можна зобразити у вигляді спектрограми – графіка, який показує, які частоти є в голосі і як вони змінюються з часом.

Дискретне перетворення Фур’є:

$$X(k) = \sum_{n=0}^{N-1} x(n) \times e^{-j\frac{2\pi}{N}kn}, \quad (2.1)$$

де $X(k)$ – спектральний компонент на частоті k;

$x(n)$ – дискретний сигнал;

N – кількість відліків сигналу;

$e^{-j\frac{2\pi}{N}kn}$ – основна гармоніка.

2.2 Виділення ознак голосу

На цьому етапі з цифрового сигналу виділяються частини, що містять мовну інформацію, а також визначаються параметри ознак мовлення. Основна мета цього етапу – зберегти корисні дані і відсіяти зайву інформацію.

Спочатку дані розбиваються на фрейми – невеликі часові інтервали, які частково перекривають один одного. Це означає, що кінець одного фрейму збігається з початком наступного. Такий підхід дозволяє більш ефективно аналізувати звукові дані, ніж при використанні окремих точок цифрового сигналу. Перекриття фреймів допомагає згладити результати аналізу. На практиці встановлено, що оптимально розбивати звук на фрейми тривалістю 25 мс з кроком 10 мс [18].

Для більш інформативного представлення з фреймів виділяють ознаки. Існує кілька методів представлення таких ознак, серед яких найпопулярніші: мел-частотні кепстральні коефіцієнти (MFCC), лінійні прогнозовані кепстральні коефіцієнти, коефіцієнти лінійного прогнозування, вейвлет-коефіцієнти.

2.3 MFCC ознаки

MFCC – це спосіб перетворення звукового сигналу на набір коефіцієнтів, які описують його спектральні характеристики з урахуванням сприйняття людиною.

Нехай фрейм – це вектор $x[k], 0 \le k \le N$, де N – розмір фрейму.

Алгоритм знаходження MFCC-коефіцієнтів для заданого фрейму виглядає наступним чином:

Крок 1. Розраховується спектр сигналу за допомогою дискретного перетворення Фур'є (2.1).

Крок 2. Обчислюється mel-фільтри. Mel – це «психофізична одиниця висоти звуку», заснована на суб'єктивному сприйнятті людиною. В першу чергу вона залежить від частоти звуку (а також від гучності і тембру). Ця величина показує, на скільки звук певної частоти «значущий» для людського вуха. Графік залежності mel-одиниць від частот показано на рисунку 2.1. Перетворення частоти в mel-одиницю виконується наступним чином [18]:

$$M = 2595 \times \log_{10}(1 + \frac{f}{700}), \tag{2.2}$$

де M – частота у шкалі Мела;

f – частота у Гц.

Обернене перетворення має наступний вигляд:

$$F = 700 \times (e^{\frac{M}{2595}} - 1), \tag{2.3}$$

де M – частота у шкалі Мела;

f – частота у Гц.

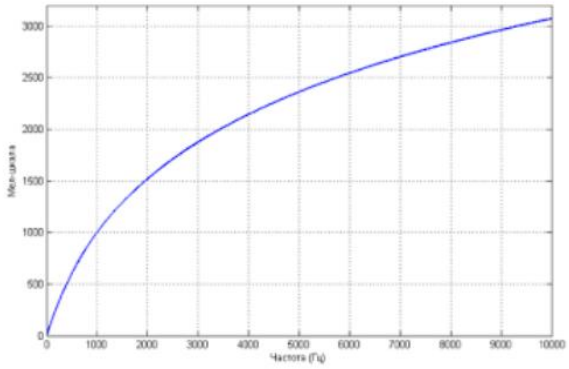

Рисунок 2.1 – Графік залежності mel-одиниць від частот

Крок 3. Застосування фільтрів на шкалі Мелла. Щоб розкласти спектр сигналу по мел-шкалі, необхідно створити мел-фільтри. Кожен мел-фільтр – трикутна віконна функція, яка визначає кількість енергії в певному частотному діапазоні, що дозволяє обчислити мел-коефіцієнт. Знаючи кількість мел-коефіцієнтів і частотний діапазон, можна сформувати наступний набір фільтрів (рис. 2.2) [19].

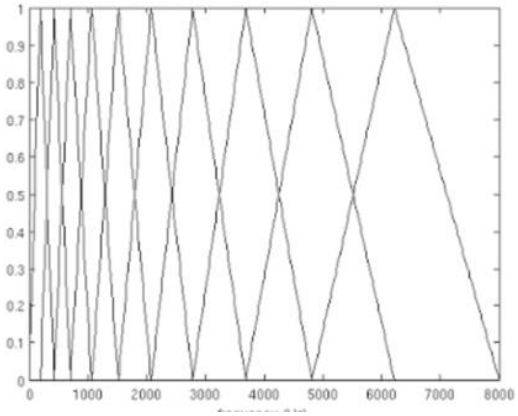

Рисунок 2.2 – Графік mel-фільтрів

Крок 4. Застосовується логарифмування результатів, щоб перетворити енергію на децибели та зменшити вплив високоенергетичних частот.

Крок 5. Застосовується зворотне косинусне перетворення (DCT). За допомогою DCT можна обчислити кепстральні коефіцієнти. Його основна мета полягає в «стисненні» отриманих даних, при цьому збільшуючи значущість перших коефіцієнтів і зменшуючи важливість останніх [19].

DCT рахується наступним чином:

$$C(k) = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}(n+\frac{1}{2})k\right], \quad (2.4)$$

де $C(k)$ – коефіцієнт DCT для частоти k;

x_n – вхідний сигнал довжини N;

N – загальна кількість точок сигналу;

k – [0,1,…,N-1].

У результаті для кожного фрейму отримуємо набір MFCC коефіцієнтів. Надалі їх можна буде використати у якості вхідних даних для акустичної моделі.

2.4 Акустичне моделювання

Акустичне моделювання є ключовим компонентом системи розпізнавання мовлення, оскільки воно відповідає на питання, яка фонема була вимовлена у заданому фреймі. Найчастіше відповідь не може бути однозначною, тому використовуються ймовірнісні підходи: для певного сигналу деякі фонеми можуть мати вищу ймовірність, тоді як інші можна відкинути. По суті, акустична модель – це функція, яка отримує на вхід певну ділянку звукового сигналу (фрейм) і повертає розподіл ймовірностей фонем для цього фрейму.

Сучасні методи розпізнавання мовлення можна поділити на кілька основних класів:

– динамічне програмування. Це метод динамічного вирівнювання часової шкали (DTW);

– приховані марковські моделі (HMM);

– нейронні мережі;

– гібридні методи. Методи, які поєднують переваги різних підходів. Популярним є використання гібридних моделей, що комбінують приховані марковські моделі з нейронними мережами (HMM-NN). У таких методах нейронна мережа обчислює ймовірності спостережень для HMM, що дозволяє підвищити точність розпізнавання за рахунок гнучкості нейронних мереж і стійкості HMM до шумів та інших варіацій у даних.

2.4.1 DWT алгоритм

Алгоритм DWT використовується для обробки цифрових сигналів мовлення, дозволяючи визначати слова шляхом порівняння числових представлень цих сигналів або їх спектрограм. Основною задачею є компенсація різної тривалості звукових послідовностей та нелінійності звуку, що досягається шляхом знаходження оптимальної деформації між рядами різної довжини.

Існує два основних підходи до застосування алгоритму:

– пряме порівняння числових форм сигналів. Для кожної послідовності створюється зменшена числова підпослідовність. Оригінальна послідовність може складатись із кількох тисяч значень, тоді як підпослідовність може мати кілька сотень. Зменшення кількості значень виконується видаленням проміжних точок, не змінюючи загальної форми сигналу. Хоча це зменшує точність, швидкість розпізнавання підвищується завдяки збільшенню обсягу словника [20];

– подання сигналів у вигляді спектрограм і застосування DTW для порівняння. Цей метод передбачає розділення цифрового сигналу на інтервали, які накладаються один на одного. Для кожного інтервалу розраховується швидке перетворення Фур’є (FFT), яке зберігається у матриці спектрограми. Параметри залишаються однаковими для всіх обчислень: тривалість імпульсу, довжина FFT та рівень перекриття імпульсів. Отримана

спектрограма, що представляє матрицю енергії, дозволяє застосувати DTW для порівняння сигналів [21].

DTW застосовується до різних типів даних таких як відео, аудіо та графіки та підходить для аналізу будь-яких лінійно представлених даних. Історично DTW широко використовувався для розпізнавання мови, але зараз він частково витіснений більш сучасними методами, такими як моделі на основі прихованих марковських процесів (HMM).

2.4.2 Приховані марковські моделі (HMM)

При розпізнаванні мовлення HMM використовується для моделювання послідовності фонем, які відповідають вимовленому слову. Звуковий сигнал розбивається на фрейми, і кожен фрейм описується набором ознак (наприклад, кепстральних коефіцієнтів). Далі ці фрейми подаються на вхід HMM, яка оцінює ймовірність належності кожного фрейму до певної фонеми, і на основі цього обирається послідовність фонем, яка найбільш ймовірно відповідає вхідному сигналу [22].

Основні поняття HMM:

– марковський процес. Це стохастичний процес, який має властивість «відсутності пам’яті», тобто майбутній стан залежить лише від поточного стану і не залежить від попередніх станів. Наприклад, перехід від одного стану до іншого залежить тільки від того, в якому стані система зараз перебуває;

– приховані стани. В моделі HMM є набір станів, які називаються прихованими, оскільки вони не спостерігаються безпосередньо. Натомість спостерігається послідовність сигналів або спостережень, які залежать від цих прихованих станів;

– ймовірності переходів. Визначають ймовірності переходу від одного стану до іншого. Це задається матрицею переходів *A*, де елемент $a_{i,j}$ відповідає ймовірності переходу зі стану *i* до стану *j*;

– ймовірності спостережень. Визначають ймовірності появи спостереження (наприклад, конкретного звуку) у кожному зі станів. Це описується матрицею спостережень *B*, де $b_j(o_t)$ – це ймовірність того, що спостереження o_t відбувається у стані *j*;

– початкові ймовірності станів. Задаються вектором π, де π_i – це ймовірність того, що початковий стан системи дорівнює *i*.

Модель HMM можна описати наступним чинном [22]:

$$\lambda = (A, B, \pi) \ , \tag{2.5}$$

де *A* – матриця ймовірностей переходів між станами;

B – матриця ймовірностей спостережень;

π – початкові ймовірності станів.

Для визначення прихованої марковської моделі необхідно ввести такі основні елементи:

– кількість станів моделі (*N*). Це кількість різних станів, які може приймати система. Набір усіх можливих станів позначається як $S = \{S_1, S_2, ..., S_N\}$, де кожен S_i відповідає певному стану. Поточний стан моделі в момент часу *t* позначається як q_t [23];

– кількість символів спостережень (*M*). Це розмір дискретного алфавіту, тобто кількість різних спостережуваних символів, які може генерувати модель. Ці символи відповідають реальним спостереженням системи. Набір спостережуваних символів позначається як $V = \{V_1, V_2, ..., V_N\}$;

– матриця ймовірностей переходів між станами (*A*). Ця матриця визначає ймовірності переходу зі стану *i* у стан *j*. Елемент матриці позначається як $a_{i,j}$:

$$a_{i,j} = P(q_{t+1} = S_j \mid q_t = S_i), 1 \le i, j \le N, \tag{2.6}$$

де *P* – ймовірність;

q_t – стан в момент часу *t*;

S_i, S_j – стани;

– матриця ймовірностей спостережень (*B*). Визначає ймовірність отримання певного спостереження у кожному зі станів. Елемент матриці позначається як $b_j(k)$:

$$b_j(k) = P(O_t = v_k \mid q_t = S_j), 1 \le j \le N, 1 \le k \le M, \tag{2.7}$$

де O_t – спостереження в момент часу *t*;

P – ймовірність;

q_t – стан в момент часу *t*;

S_j – стан;

v_k – можливий символ з алфавіту *V*, яке це спостереження може приймати;

– початковий розподіл ймовірностей станів (π). Визначає ймовірність того, що система перебуває у стані S_i на початку, тобто в момент часу $t = 1$. Початковий розподіл позначається як π_i:

$$\pi_i = P(q_1 = S_i), \tag{2.8}$$

де S_i – стан;

q_1 – стан на початку.

Вибравши значення для *N*, *M*, *A*, *B* та π, можна використовувати HMM для генерації певної послідовності спостережень $O = \{O_1, O_2, ..., O_T\}$, де кожне спостереження O_t є символом з алфавіту *V*, а T – загальна кількість спостережень у послідовності [23]. Процес генерації здійснюється за такою схемою:

Крок 1. Обирається початковий стан $q_1 = S_i$ відповідно до розподілу π.

Крок 2. Встановлюється $t = 1$.

Крок 3. Генеруються спостереження $O_t = v_k$ згідно з ймовірністю $b_i(k)$ у стані S_i.

Крок 4. Перехід до нового стану $q_{t+1} = S_j$ згідно з ймовірністю переходу a_{ij}.

Крок 5. Збільшується t на 1. Якщо $t < T$, то перехід на Крок 3, інакше процедура завершається.

Для підвищення точності розпізнавання голосових команд часто використовуються гібридні моделі, які поєднують HMM з іншими методами, такими як нейронні мережі. Наприклад, нейронні мережі можуть використовуватися для обчислення ймовірностей спостережень, які потім передаються в HMM. Це дозволяє поєднати здатність HMM працювати з часовими даними та високу точність нейронних мереж в обробці нелінійних залежностей [24].

2.4.3 Рекурентні нейроні мережі

Нейронні мережі в гібридних моделях виконують функцію прогнозування ймовірностей фонем або інших елементів мовлення. Класична гібридна архітектура зазвичай складається з декількох шарів штучних нейронів, які можуть бути згруповані у три основні компоненти:

– вхідний шар, який отримує акустичні ознаки мовного сигналу. Це можуть бути мел-кепстральні коефіцієнти, перетворені спектри або інші представлення звуку, які забезпечують мережу необхідними характеристиками для розпізнавання [25];

– приховані шари, які складаються з одного або декількох шарів нейронів, які обробляють вхідну інформацію, витягуючи ознаки більш високого рівня. Рекурентні нейронні мережі (RNN) особливо добре підходять для обробки послідовностей, оскільки здатні зберігати контекст попередніх фреймів [25];

– вихідний шар, який прогнозує ймовірності різних фонем, які передаються на приховану марковську модель (HMM), що виконує обробку послідовностей та визначає кінцеві результати на основі ймовірнісного розподілу фонем.

Під час звичайного спілкування людина запам’ятовує не тільки те слово, яке вимовлене зараз, а й ті, що були сказані раніше. Мозок не розглядає кожен момент як абсолютно новий, а будує словесні та контекстуальні зв’язки, часто підсвідомо. На відміну від цього, звичайні нейронні мережі та приховані Марковські моделі позбавлені такої здатності, що ускладнює розпізнавання довгих фраз і речень. Щоб вирішити подібні динамічні задачі, у 1997 році були створені рекурентні нейронні мережі – тип штучних нейронних мереж, де зв’язки між вузлами утворюють спрямований у часі граф, що дозволяє зберігати інформацію про попередні стани [25].

Рекурентні нейронні мережі (RNN) – це клас нейронних мереж, які особливо добре підходять для обробки послідовних даних, таких як мова, текст або часові ряди, завдяки здатності зберігати інформацію про попередні кроки. На відміну від класичних нейронних мереж, де зв’язки йдуть лише в одному напрямку (від входу до виходу), RNN мають зворотні зв’язки, що дозволяє їм підтримувати стан пам’яті та враховувати попередні значення під час прийняття рішень [26].

Рекурентна нейронна мережа складається з шарів нейронів, які на кожному часовому кроці приймають вхід і генерують вихід, враховуючи поточний стан пам’яті. На кожному кроці часового ряду *t* мережа отримує на вхід поточний вектор ознак $x^{(t)}$ і стан пам’яті попереднього кроку $h^{(t-1)}$. Поточний стан h^t обчислюється як [27]:

$$h^{(t)} = \sigma(W_h x^{(t)} + U_h h^{(t-1)} + b_h) \ , \tag{2.9}$$

де W_h, U_h – вагові матриці, відповідаючі за обробку входу та попереднього стану;

b_h – вектор зміщення;

σ – нелінійна активаційна функція.

На основі стану $h^{(t)}$ формується вихідний сигнал мережі $y^{(t)}$, що представляє собою прогноз на поточному часовому кроці [27]:

$$y^{(t)} = \phi(W_y h^{(t)} + b_y) \ , \tag{2.10}$$

де W_y – матриця ваг для переходу з прихованого шару до вихідного;

b_y – вектор зміщення;

ϕ – нелінійна активаційна функція.

У ситуаціях, коли відстань між актуальною інформацією і місцем, де вона потрібна, невелика, RNN можуть навчитися використовувати дані з попередніх ітерацій. Однак у випадках, коли для розуміння контексту потрібно більше інформації, якість роботи RNN знижується зі збільшенням цієї відстані. Для таких завдань розроблено спеціальний тип архітектури RNN, відомий як Long Short-Term Memory (LSTM), здатний запам’ятовувати довготривалі залежності. Основним компонентом LSTM є клітинка та її стан [28]. Рисунок 2.3 [29] детально ілюструє її вигляд.

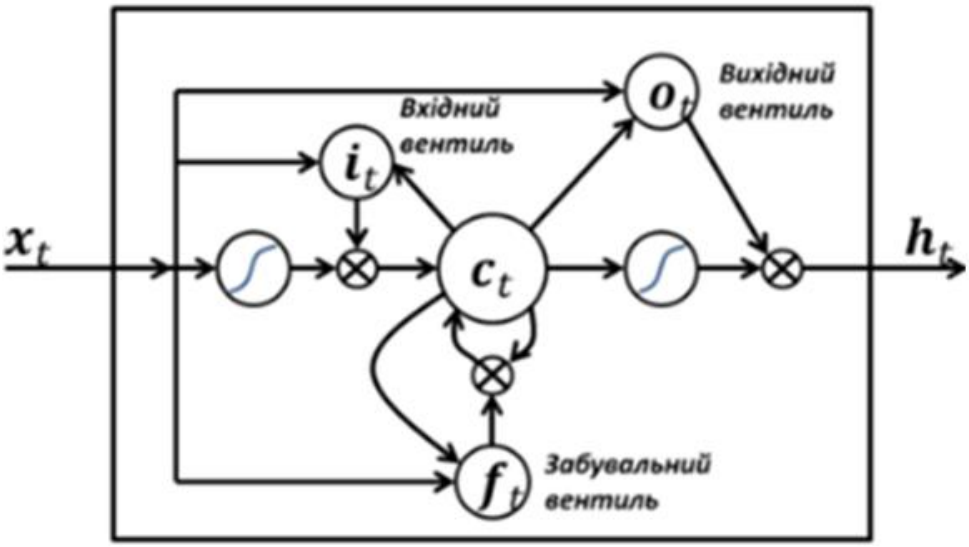

Рисунок 2.3 – Клітинка LSTM

Клітинка це основний елемент LSTM, який зберігає інформацію протягом тривалого часу. Клітинка пам’яті може зберігати або забувати дані, ґрунтуючись на входах, які вона отримує. Саме вона робить LSTM здатною запам’ятовувати важливі аспекти вхідної інформації.

Стан у LSTM представляє собою інформацію, що зберігається і передається між часовими кроками. Існує звичайний стан комірки (довготривалий стан), який переносить накопичену інформацію з попередніх часових кроків. Він постійно оновлюється під впливом гейтів. Також є прихований стан (короткостроковий стан), що передається між часовими кроками і визначає вихід поточної комірки [28].

Гейт – це механізм у LSTM, що вирішує, яку частину інформації оновити, зберегти або видалити з комірки. LSTM використовує три типи гейтів:

– гейт запису (2.11) вирішує, що додати в стан комірки;

– гейт забування (2.12) визначає, що забути зі стану комірки;

– гейт виходу (2.13) визначає, яка інформація з комірки піде на вихід та у прихований стан.

$$i(t) = \sigma(W_i x(t) + U_i h(t-1) + b_i) \ , \tag{2.11}$$

де W_i – ваги для гейту запису;

$x(t)$ – вхідний вектор у момент часу *t*;

U_i – ваги для прихованого стану *h(t−1)* від попереднього часу;

$h(t-1)$ – прихований стан на попередньому кроці;

b_i – зсув для гейту запису;

σ – сигмоїдна функція активації, яка обмежує значення між 0 і 1.

$$f(t) = \sigma(W_f x(t) + U_f h(t-1) + b_f) \ , \tag{2.12}$$

де W_f – ваги для гейту забування;

b_f – зсув для гейту забування.

$$o(t) = \sigma(W_o x(t) + U_o h(t-1) + b_o) \ , \tag{2.13}$$

де W_o – ваги для вихідного гейту;

b_o – зсув для вихідного гейту.

Оновлення стану комірки пам’яті, яка зберігає основну інформацію, накопичену мережею має наступний вигляд [28]:

$$c(t) = f(t) \otimes c(t-1) + i(t) \otimes \tanh(W_c x(t) + U_c h(t-1) + b_c) \ , \tag{2.14}$$

де $c(t-1)$ – стан комірки пам’яті на попередньому кроці;

W_c – ваги для стану комірки;

U_c – ваги для попереднього прихованого стану;

b_c – зсув для комірки пам’яті;

tanh – гіперболічна тангенс-функція активації, яка обмежує значення між -1 і 1;

$\otimes$ – поелементне множення.

Прихований стан *h(t)*, який визначає вихід LSTM-комірки на поточному кроці має наступний вигляд:

$$h(t) = o(t) \otimes \tanh(c(t)) \ , \tag{2.15}$$

де $c(t)$ – оновлений стан комірки пам'яті.

Процес навчання нейронної мережі для гібридного методу виконується на основі великої кількості попередньо розмічених акустичних даних. Нейронна мережа поступово вчиться асоціювати певні акустичні ознаки з відповідними фонемами. Найчастіше для цього використовується алгоритм зворотного поширення помилки разом з градієнтним спуском для оптимізації параметрів мережі.

2.5 Мовна модель

Мовна модель допомагає вибрати найбільш вірогідну послідовність слів серед усіх можливих варіантів, які можуть бути інтерпретовані на основі акустичних характеристик. Це здійснюється шляхом комбінування інформації від акустичної моделі, яка визначає ймовірності певних звукових характеристик для різних фонем, та мовної моделі, яка визначає ймовірність певних послідовностей слів [30].

Мовна модель визначає ймовірність послідовності слів $W = w_1, w_2, ..., w_T$. Завдання полягає у знаходженні ймовірності *P(W)*, яка може бути виражена через правило умовної ймовірності:

$$P(W) = P(w_1) \times P(w_2 \mid w_1) \times ... \times P(w_T \mid w_1, w_2, ..., w_{T-1}), \quad (2.16)$$

де w_i – окреме слово;

T – довжина послідовності.

2.6 Критерій якості роботи системи розпізнавання голосу

Критерій якості роботи системи розпізнавання голосу оцінює, наскільки точно і ефективно система розпізнає мовні команди або текст з голосового сигналу. Існує кілька метрик та підходів, які використовуються для вимірювання якості таких систем.

World Error Rate (WER) є загальноприйнятою метрикою ефективності системи розпізнавання мовлення. Щоб обчислити WER, потрібно вирівняти два текстових рядки: перший – це результат розпізнавання, а другий – еталонний текст, який повинен був бути сказаний. Це вирівнювання здійснюється з використанням алгоритму динамічного програмування для обчислення відстані Левенштейна. Відстань Левенштейна визначає «вартість» редагування, тобто мінімальну кількість або зважену суму операцій редагування, необхідних для перетворення одного рядка в інший за допомогою найменшої кількості замін (S), видалень (D) і вставок (I) слів.

$$WER = \frac{S + D + I}{N} \times 100\% , \quad (2.17)$$

де S – кількість замін;

D – кількість пропусків;

I – кількість вставок;

N – загальна кількість слів у тексті.

2.7 Відстань Левенштейна

Відстань Левенштейна (редакційна відстань) – метрика, яка вимірює по модулю різницю між двома рядками. Вона визначається як мінімальна кількість односимвольних операцій (вставки, видалення, заміни), необхідних для перетворення одного рядка на інший. Кожній операції (вставка, заміна, видалення) можна поставити у відповідність вартість (числовий еквівалент витрат за її виконання):

- $w(a,b)$ – вартість заміни символу a на символ b;
- $w(e,b)$ – вартість вставки символу b;
- $w(a,e)$ – вартість видалення символу a.

Зазвичай при розрахуванні відстань Левенштейна використовує наступна вартість операцій:

- $w(a,b) = 1$;
- $w(e,b) = 1$;
- $w(a,e) = 1$.

Знаходження відстань Левенштейна зводиться до знаходження такої послідовності операцій сума вартості яких буде мінімальною.

Для знаходження відстані Левенштейна існує декілька алгоритмів, одним з них є алгоритм Вагнера-Фішера (2.18), суть якого зводиться до побудови матриці розміру N+1, M+1 та заповнення її елементів на основі рекурентного виразу [31]:

$$D(i,j)=\begin{cases}0, i=0,\ j=0,\\ i,\ j=0, i>0,\\ j, i=0,\ j>0,\\ \min=\begin{cases}D(i-1,j)+1,\\ D(i,j-1)+1,\\ D(i-1,j-1)+m(S_1[i],S_2[j]),\end{cases} & i,j>0\ ,\end{cases} \qquad (2.18)$$

де S_1, S_2 – строки, що порівняються;

$S_1[i], S_2[j]$ – символи, що відповідають елементам матриці;

$m(S_1[i], S_2[j])$ – результат порівняння $S_1[i], S_2[j]$: 0 якщо різні символи та 1, якщо однакові.

Значення у нижньому правому куті матриці $D(m,n)$ буде дорівнювати редакційній відстані між рядками S_1 та S_2.

Іншим методом є алгоритм Дамерау-Левенштейна. Цей алгоритм використовується коли треба виявляти та коригувати помилки, пов'язані з порядком символів в рядках. До нього ще додається операція транспозиції (обмін місцями двох сусідніх символів) У інших випадках можна використовувати алгоритм Вагнера-Фішера, так як він більш швидкий за рахунок меншого обсягу обчислень [31].

Нехай S_1 і S_2 – два рядки, де S_1 має довжину *M*, а S_2 – довжину *N*. Алгоритм Дамерау-Левенштейна обчислює мінімальну кількість операцій (заміни, вставки, видалення або транспозиції), необхідних для перетворення рядка S_1 у рядок S_2.

Для реалізації алгоритму використовується матриця *D* розміром $(M+1)\times(N+1)$, де елемент $D[i][j]$ представляє мінімальну кількість операцій для перетворення перших *i* символів рядка S_1 у перші *j* символів рядка S_2.

Алгоритм складається з наступних кроків:

Крок 1. Ініціалізація матриці:

– заповнити перший рядок $D[i][0]=i$, де $0\le i\le M$, що відповідає видаленню *i* символів;

– заповнити перший стовпець $D[0][j]=j$, де $0\le j\le N$, що відповідає вставці *j* символів.

Крок 2. Рекурсивне обчислення елементів матриці. Для кожного $1\le i\le M$ і $1\le j\le N$:

$$D[i][j] = \min \begin{cases} d[i-1][j]+1, \\ d[i][j-1]+1, \\ d[i-1][j-1]+\cos t \end{cases}, \quad (2.19)$$

де *cost* – дорівнює 0, якщо $S_1[i] = S_2[j]$, або 1, якщо символи різні.

Крок 3. Додавання обробки транспозиції. Якщо $i > 1$, $j > 1$, $S_1[i] = S_2[j-1]$ і $S_1[i-1] = S_2[j]$, то розраховується додаткове значення:

$$D[i][j] = \min(D[i][j], D[i-2][j-2]+1) \ . \quad (2.20)$$

Крок 4. Значення у нижньому правому куті матриці $D(m,n)$ буде відповідати відстані Дамерау-Левенштейна між рядками S_1 і S_2.

Редакційна відстань є важливим інструментом у задачах, де потрібно порівнювати рядки або послідовності, особливо в умовах неточності або варіативності даних [32]. Редакційна відстань широко використовується в різних задачах, таких як:

– розпізнавання тексту та мовлення. Порівняння результату розпізнавання з еталонним текстом для оцінки точності (наприклад, обчислення WER);

– корекція орфографії. Визначення найвірогідніших варіантів виправлення помилок у словах.

3 ДОСЛІДЖЕННЯ КОМП'ЮТЕРНОЇ МОДЕЛІ РОЗПІЗНАВАННЯ ГОЛОСОВИХ КОМАНД

3.1 Обґрунтування вибору середовища програмної реалізації

У рамках кваліфікаційної роботи був розроблений застосунок для розпізнавання голосових команд за допомогою нейронних мережей. Для реалізації системи було обрано мову програмування Python разом з інтегрованим середовищем розробки PyCharm. Вибір зумовлений тим, що Python – одна з найпопулярніших мов у сучасній індустрії, має високий попит серед розробників та відмінно підходить для задач штучного інтелекту й машинного навчання. Зокрема, ця мова завоювала популярність завдяки своєму лаконічному синтаксису, великому вибору бібліотек та гнучкості в реалізації різноманітних проєктів [33].

PyCharm є одним із провідних кросплатформових інтегрованих середовищ для розробки на Python, пропонуючи розробникам широкий набір інструментів, що значно полегшують процес написання, налагодження і тестування коду. Python, як мова високого рівня загального призначення, має динамічну сувору типізацію та автоматичне управління пам'яттю, що дозволяє розробникам ефективно зосереджуватись на логіці програми. Особливість Python полягає також у повній об'єктно-орієнтованій моделі, де будь-яка сутність виступає у формі об'єкта [34].

Мова Python знайшла широке застосування у безлічі галузей. Ось декілька прикладів:

– аналіз даних. Дані сьогодні стали невід'ємною складовою будь-якого бізнесу чи наукового дослідження, і компанії по всьому світу активно займаються збором, обробкою і аналізом інформації для прийняття ефективних рішень. Python перевершує інші мови у цьому напрямку завдяки багатству бібліотек. Наприклад, pandas і NumPy є найбільш популярними

інструментами для обробки та аналізу даних, що використовуються і в розробці застосунку [35];

– візуалізація даних. Для представлення обробленої інформації у зручному та інтуїтивно зрозумілому вигляді Python пропонує безліч потужних бібліотек, таких як matplotlib. Це дає змогу наочно демонструвати результати досліджень чи роботи програм, що також було впроваджено у нашій розробці [36];

– машинне навчання. Цей напрямок, який є основною частиною штучного інтелекту, також активно використовує Python. Завдяки можливостям створення й тренування моделей, машина може аналізувати минулі дані, вивчати закономірності та передбачати майбутні події;

– веброзробка. Мова Python підходить для створення бекенд-систем завдяки своїм фреймворкам, таким як Django і Flask;

– розробка програмного забезпечення. Python дозволяє створювати як прості, так і комплексні програмні продукти.

Python має низку переваг, які сприяють його популярності серед розробників:

– легкість у навчанні та розумінні. Синтаксис мови простий, що робить її ідеальною для новачків. Python дозволяє писати код коротко і ясно, часто з меншою кількістю рядків порівняно з іншими мовами програмування, такими як C++ чи Java [37];

– інтерпретована природа. Python виконує код рядок за рядком, що спрощує пошук помилок. При виявленні помилки виконання одразу припиняється, а розробник отримує точне повідомлення, яке допомагає у відстеженні проблеми;

– динамічна типізація. У Python типи даних змінних не визначаються заздалегідь, і їх можна присвоювати автоматично під час виконання програми. Це звільняє розробника від необхідності жорстко вказувати типи даних;

– велика бібліотека. Стандартна бібліотека Python надзвичайно обширна і включає практично всі функції, необхідні для реалізації різних завдань. Це робить мову дуже привабливою для швидкої розробки;

– портативність. Код, написаний на Python, є переносимим між різними операційними системами. Це означає, що програми можуть легко працювати на Windows, macOS чи Linux без значних змін у коді.

Таким чином, Python є одним із найкращих варіантів для реалізації проєктів у сфері штучного інтелекту, машинного навчання та аналізу даних, забезпечуючи зручність та ефективність роботи розробника.

3.1.1 NumPy

NumPy – це одна з ключових бібліотек Python, яка надає потужний інструментарій для виконання широкого спектру математичних обчислень. Вона охоплює як базові функції, так і складні операції лінійної алгебри, забезпечуючи можливість ефективної роботи з масивами та матрицями різних розмірностей. NumPy додає до Python високопродуктивні функції для обробки багатовимірних структур даних, що робить її незамінною для наукових обчислень та чисельних задач [38].

Головною перевагою NumPy є її здатність забезпечувати швидкі й оптимізовані операції з масивами. Бібліотека відіграє центральну роль у багатьох сучасних проєктах, що потребують обробки великих обсягів числових даних, таких як аналіз даних і машинне навчання. Завдяки своїй універсальності NumPy широко застосовується у різних галузях науки та інженерії. Її підтримка високорівневих математичних функцій дозволяє легко маніпулювати векторами, матрицями та багатовимірними масивами.

Сьогодні NumPy є основою для інших популярних бібліотек у Python екосистемі, які будуються на її потужностях. Серед них варто відзначити scikit-learn, що використовується для машинного навчання, SciPy – для

наукових і технічних обчислень, pandas – для обробки та аналізу даних, а також TensorFlow – для створення нейронних мереж. Всі ці інструменти успішно інтегруються з NumPy, забезпечуючи швидке і ефективне виконання складних завдань.

NumPy має широкий спектр використання:

– наукові обчислення. Учені з різних сфер такі як математики, фізики, біоінформатики, обчислювальної хімії чи когнітивної психології активно застосовують NumPy для розв'язання багатовимірних задач, що потребують значної обчислювальної потужності;

– аналіз даних. NumPy лежить в основі екосистеми для роботи з даними у Python. Її функції використовуються на кожному етапі обробки даних, починаючи з вилучення і перетворення, і закінчуючи моделюванням, аналізом та візуалізацією результатів. Бібліотека забезпечує швидкий і ефективний аналіз великих обсягів інформації, що має важливе значення для сучасного бізнесу і досліджень [38];

– машинне навчання. Бібліотеки машинного навчання, такі як scikit-learn та SciPy, базуються на функціоналі NumPy для виконання обчислювальних операцій. Завдяки NumPy, ці інструменти можуть обробляти великі масиви даних та виконувати навчання моделей з високою ефективністю.

3.1.2 Pandas

Pandas – це популярна програмна бібліотека, написана мовою Python, призначена для зручної обробки та аналізу даних. Її архітектура побудована поверх бібліотеки NumPy, яка забезпечує числові операції низького рівня. Завдяки Pandas, робота з даними у Python стає значно простішою, оскільки вона пропонує високорівневі структури даних, що дозволяють зручно маніпулювати як простими, так і складними інформаційними наборами.

Назва бібліотеки Pandas походить від терміна «panel data», що означає панельні дані. Панельні дані – це структурована у вигляді таблиць інформація, яка зазвичай збирається у процесі тривалих досліджень. Pandas дозволяє працювати з даними в різних форматах, зокрема, в таблицях, подібних до тих, що використовуються у програмах на кшталт Microsoft Excel. Ця бібліотека надає потужний функціонал для маніпулювання великими обсягами даних, аналізу та навіть інтеграції з іншими джерелами, такими як бази даних або файли CSV.

Головними структурами даних у Pandas є об’єкти Series і DataFrame:

– Series – це одновимірний масив, подібний до списку або стовпця в електронній таблиці. Кожен елемент у Series має індекс, що дозволяє легко посилатися на потрібні значення. Виглядає Series як колонка, де зліва розташовані індекси, а справа – дані;

– DataFrame – це більш складна структура, яка являє собою двовимірний масив, організований як таблиця з рядками і стовпцями. DataFrame можна порівняти з листом в Excel, оскільки в ньому можна зберігати дані у вигляді таблиці, де кожен стовпець є об’єктом Series. За допомогою DataFrame зручно об’єднувати різні джерела даних, сортувати інформацію за певними параметрами, фільтрувати записи та виконувати обчислення.

Однією з ключових переваг Pandas є її здатність працювати з даними різних форматів. Наприклад, бібліотека дозволяє зчитувати та зберігати таблиці у форматах CSV, Excel, JSON, а також інтегруватися з базами даних SQL. Це забезпечує гнучкість у роботі з інформацією, спрощуючи процеси попередньої обробки та аналізу даних.

Бібліотека Pandas надає розробникам великий набір інструментів для обробки даних: групування, сортування, фільтрацію та виконання математичних операцій. Завдяки цьому Pandas широко використовується в різних сферах, включаючи дослідження, бізнес-аналітику, фінансове моделювання, машинне навчання та багато іншого. Вона значно спрощує

виконання завдань, пов’язаних з обробкою та аналізом великих масивів інформації [39].

3.1.3 Matlotlib

Matplotlib – це популярна бібліотека Python, призначена для візуалізації даних у двовимірних і тривимірних форматах. Метою розробки Matplotlib було покращити представлення наукових даних за допомогою графіків, що могли б допомогти у вивченні складних закономірностей, зокрема тих, що спостерігалися в мозковій активності.

З моменту випуску Matplotlib перетворився на один із найпопулярніших інструментів для створення графіків у Python. Він надає розробникам і дослідникам широкий набір можливостей для візуалізації, дозволяючи легко створювати різноманітні види графіків, що підходять для представлення як простих, так і складних наборів даних. Бібліотека Matplotlib часто використовується в поєднанні з іншими бібліотеками, такими як NumPy та pandas, для створення інформативних та якісних графічних презентацій даних [40].

Matplotlib підтримує велику кількість видів графіків і діаграм, що дозволяють повноцінно візуалізувати інформацію:

– діаграми розсіювання. Використовуються для демонстрації залежності між двома змінними, зображуючи кожне спостереження у вигляді точки на площині;

– стовпчасті діаграми. Допомагають відображати категоріальні дані та частотний розподіл числових значень у вигляді стовпців;

– кругові діаграми. Застосовуються для показу часткових значень відносно загальної суми, наприклад, для демонстрації відсотків;

– діаграми стебло-листя. Специфічний формат, який показує розподіл даних;

– контурні графіки. Використовуються для візуалізації тривимірних даних у вигляді ізоліній, що показують точки однакового значення на площині;

– поля градієнтів. Відображають зміни значень за допомогою градієнтів кольору, що наочно демонструє розподіл даних у просторі;

– спектральні діаграми. Застосовуються для аналізу та представлення частотних компонент сигналів.

3.1.4 Scikit-learn

Scikit-learn – це один із найвідоміших та найбільш широко використовуваних пакетів Python, призначених для аналізу даних і реалізації алгоритмів машинного навчання. Цей інструмент є важливим елементом Python екосистеми у сфері науки про дані, надаючи можливості для вирішення різноманітних задач, починаючи від класифікації і закінчуючи кластеризацією [41].

Scikit-learn надає великий набір алгоритмів машинного навчання, а також інструменти для попередньої обробки даних, оцінки моделей та їх налаштування. Пакет відзначається доступною документацією, що детально описує методи та функції, а також пояснює принципи роботи використовуваних алгоритмів. Завдяки цьому scikit-learn став одним із найпопулярніших інструментів серед розробників і дослідників даних, оскільки дозволяє швидко і просто почати роботу з машинним навчанням.

Основою scikit-learn є бібліотеки NumPy та SciPy, які забезпечують числові обчислення та наукову обробку даних. Тому для ефективного використання цього інструмента варто мати хоча б базове уявлення про роботу з масивами та математичними функціями цих бібліотек. Пакет scikit-learn безкоштовний і може застосовуватись як для навчальних, так і для

комерційних проєктів, що робить його доступним для широкого кола користувачів.

Scikit-learn охоплює широкий спектр алгоритмів, які використовуються для різних завдань машинного навчання [41]:

– класифікація. Пакет включає алгоритми, які дозволяють вирішувати задачі розпізнавання та класифікації, наприклад, метод опорних векторів, логістичну регресію, наївний баєсів класифікатор, дерева рішень і метод випадкового лісу;

– регресія. Для прогнозування числових значень доступні різні регресійні моделі, такі як лінійна регресія, регресія Lasso та метод посилення градієнта;

– кластеризація. Scikit-learn пропонує інструменти для сегментації даних на групи без використання міток, наприклад, алгоритм *k*-середніх, ієрархічну кластеризацію та метод DBSCAN;

– зменшення розмірності. Для зменшення кількості змінних, що описують дані, доступні методи, такі як головні компоненти та аналіз дискримінантів Фішера;

– попередня обробка даних. Пакет забезпечує функції для нормалізації, масштабування, перетворення категоріальних змінних і розбиття даних на навчальні та тестові набори.

3.2 Аналіз вимог користувача до програмного продукту

Для того щоб користувачі могли без труднощів користуватися системою голосового управління, вона має відповідати кільком основним вимогам.

По-перше, система повинна розпізнавати неперервне мовлення, подібне до повсякденної мови. Це завдання є складнішим, ніж просте розпізнавання окремих слів.

По-друге, система має бути оптимізована для швидкої роботи. Сказаний текст повинен майже одразу виконувати відповідну системну команду. Адже головна мета голосового управління – автоматизоване та швидше виконання завдань порівняно з ручним керуванням.

Важливою вимогою є універсальність. Система має розпізнавати будь-який голос, незалежно від тембру, статі, віку, інтонації чи швидкості мовлення.

Отже застосунок має швидко й точно обробляти голосові команди, визначати сказані команди та одразу їх виконувати.

3.3 Програмна реалізація

Розглянувши усі математичні моделі, можна побудувати систему, здатну розпізнавати людський голос. Така система включає в себе кілька етапів обробки сигналу, кожен з яких грає важливу роль у забезпеченні точності розпізнавання голосових команд.

Перший етап – покадрова обробка аудіосигналу, під час якого звуковий сигнал розбивається на фрейми тривалістю від 10 до 25 мілісекунд. Це дозволяє вловлювати важливі зміни в голосі з високою точністю.

Другий етап – виділення ключових акустичних ознак, які отримують з кожного фрейму за допомогою методу MFCC (мел-кепстральних коефіцієнтів). Цей метод забезпечує ефективне перетворення аудіосигналу в набір числових характеристик, які відображають особливості звуку та значно полегшують його подальшу обробку.

Останній етап полягає у використанні акустичної моделі, навченої на великій кількості даних, яка зіставляє отримані вектори ознак із фонемами мови. Це дозволяє точно ідентифікувати звукові сигнали і, як результат, отримати текст. Запропонована система показана на рисунку 3.1.

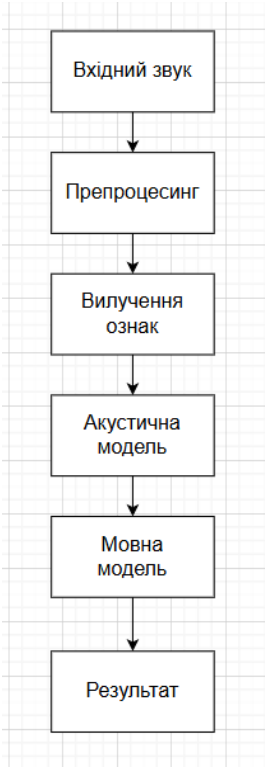

Рисунок 3.1 – Схема побудови системи розпізнавання мовлення

3.3.1 Розробка інтерфейсу

Для створення інтерфейсу користувача було обрано технологію Eel у поєднанні з HTML, CSS та Python. Eel є зручним інструментом для розробки Python застосунків з вебінтерфейсом. Eel обумовлений його зручністю для створення графічних інтерфейсів на Python із можливістю застосовувати сучасні вебтехнології для дизайну та інтерактивних елементів. Також він дозволяє легко зв’язати бекенд на Python з вебінтерфейсом, що значно спрощує розробку кросплатформного інтерфейсу.

Інтерфейс розроблено в стилі «Джарвіса» (рис. 3.2) – віртуального асистента, щоб додати застосунку сучасного та футуристичного вигляду. У цьому стилі інтерфейс складається з темної палітри кольорів, неонових відтінків синього та зеленого, а також анімацій, що створюють враження роботи високотехнологічної системи. Таке оформлення підвищує

привабливість застосунку, надаючи користувачеві відчуття взаємодії з сучасним штучним інтелектом.

Eel дозволяє легко інтегрувати інтерфейс на HTML і CSS із функціональністю на Python. Зазвичай для Python-інтерфейсів використовуються такі бібліотеки, як Tkinter або PyQt, але Eel має низку переваг.

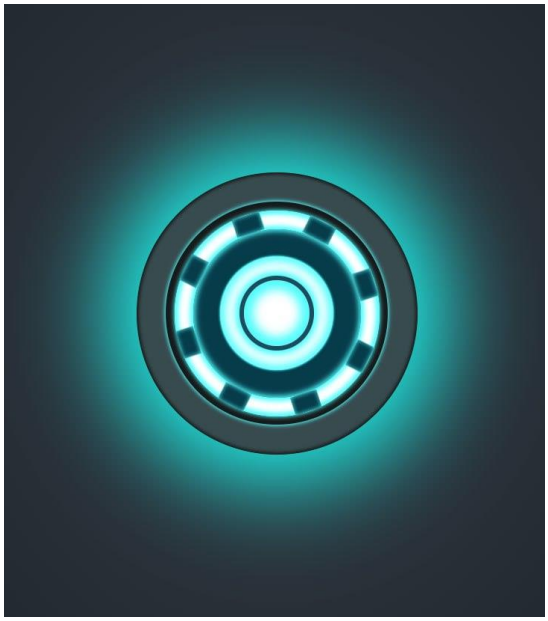

Рисунок 3.2 – Інтерфейс застосунку в стилі «Джарвіса»

По-перше, Eel підтримує вебтехнології, такі як HTML, CSS і JavaScript, що забезпечує широкі можливості для налаштування інтерфейсу та легкість створення адаптивного дизайну.

По-друге, через просту інтеграцію з Python Eel дає можливість викликати функції Python прямо з JavaScript, що спрощує передачу даних між інтерфейсом і функціональною частиною застосунку.

При завантаженні та обробці голосових команд використовуються анімації, які створюють ефект «живого» інтерфейсу, що реагує на дії користувача. Це робиться за допомогою CSS анімацій, які активуються під час запуску та розпізнавання команди. Вони допомагають користувачеві отримувати інформацію про те, коли система «слухає» або коли триває

обробка запиту. Наприклад, кнопка активації запису підсвічується, коли починається процес розпізнавання голосу, а індикатор стану відображає результат обробки запиту.

Реалізація інтерфейсу з Eel складається з декількох ключових елементів:

– HTML сторінка містить основну структуру інтерфейсу: розмітку кнопок, текстових полів, індикаторів (рис. 3.3);

```
<div id="reactor-container" class="reactor-container" onclick="ena
    <div class="reactor-container-inner circle abs-center"></div>
    <div class="tunnel circle abs-center"></div>
    <div class="core-wrapper circle abs-center"></div>
    <div class="core-outer circle abs-center"></div>
    <div class="core-inner circle abs-center"></div>
    <div id="coil-container" class="coil-container">
        <div class="coil coil-1"></div>
        <div class="coil coil-2"></div>
        <div class="coil coil-3"></div>
        <div class="coil coil-4"></div>
        <div class="coil coil-5"></div>
        <div class="coil coil-6"></div>
        <div class="coil coil-7"></div>
        <div class="coil coil-8"></div>
    </div>
</div>
```

Рисунок 3.3 – Приклад HTML сторінки для інтерфейсу

– CSS забезпечує оформлення: кольорову гаму, розмір елементів, анімації (рис. 3.4);

```
.reactor-container.active {
  width: 300px;
  height: 300px;
  margin: auto;
  position: relative;
  border-radius: 50%;
  background-color: #384c50;
  border: 1px solid rgb(18, 20, 20);
  box-shadow: 0px 0px 100px 30px rgb(28, 250, 242), 0px 0px 4px 1px rgb(18, 20, 20) inset;
}
```

Рисунок 3.4 – Приклад CSS сторінки для інтерфейсу

– JavaScript використовується для взаємодії з Python-бекендом (рис. 3.5). Через JavaScript відправляються запити до Python-скриптів для активації нейронної мережі та розпізнавання голосових команд. JavaScript також отримує дані від Python, щоб оновити інтерфейс за результатами розпізнавання.

```
async function enableOrDisableJarvis() : Promise<void>  {
    const coilContainer : HTMLElement  = document.getElementById( elementId: "coil-container");
    const reactorContainer : HTMLElement  = document.getElementById( elementId: "reactor-container");
    coilContainer.classList.toggle( token: 'active');
    reactorContainer.classList.toggle( token: 'active');
    const isActive : boolean  = coilContainer.classList.contains('active');
    if (!isActive) {
        await eel.disableJarvis()();
    } else {
        await eel.enableJarvis()();
    }
}
```

Рисунок 3.5 – Приклад JavaScript для реалізації інтерфейсу

3.3.2 Функціональна розробка

Процес розпізнавання розподілено на декілька основних етапів:

– запис голосового сигналу;
– попередня обробка голосу;
– розпізнавання команд за допомогою нейронної мережі;
– обробка результату.

Кожен із цих модулів взаємодіє з інтерфейсом, створеним за допомогою Eel, а також використовує Python для основної логіки застосунку.

Застосунок має два стани роботи: увімкнений та вимкнений. Для того щоб увімкнути застосунок потрібно натиснути на головний компонент системи, після чого застосунок програє голосову команду привітання. Якщо натиснути ще раз, то система вимкнеться та програє команду прощання.

Функція програвання аудіо застосунком реалізована за допомогою програвання аудіо файлів (рис. 3.6).

Після того, як користувач активував застосунок, він здатний обробляти звукові сигнали. Для забезпечення розпізнавання голосу в режимі реального часу функціональна архітектура системи передбачає безперервне прослуховування користувача. Застосунок постійно стежить за голосовим потоком і активує обробку аудіосигналу лише тоді, коли користувач перестає говорити. Такий підхід дозволяє забезпечити максимально природну

взаємодію, оскільки користувачеві не потрібно натискати кнопки для запуску запису.

```
def enable(self):
    self.isEnabled = True
    self.playSound('web/run.wav')

1 usage
def disable(self):
    self.isEnabled = False
    self.playSound('web/off.wav')

3 usages
def playSound(self, filename):
    pygame.mixer.music.load(filename)
    pygame.mixer.music.play()
```

Рисунок 3.6 – Програвання привітання та прощання застосунком

Для реалізації безперервного запису використовується pyaudio у Python, що дозволяє захоплювати аудіопотік безперервно і зберігати короткі інтервали в пам'яті (рис. 3.7). Цей підхід дозволяє зберігати декілька останніх секунд звуку для подальшої обробки.

```
recognizer = sr.Recognizer()
# Функція для розпізнавання голосу
1 usage
def recognize_speech():
    with sr.Microphone() as source:

        audio = recognizer.listen(source)
```

Рисунок 3.7 – Функція для розпізнавання голосу

Система слухає звук і відстежує паузи. Якщо мовлення припиняється, буфер відправляється на обробку. Це забезпечує режим реального часу, коли система автоматично переходить до розпізнавання, як тільки користувач закінчує говорити.

На етапі попередньої обробки записаний аудіобуфер фільтрується для підвищення якості та точності розпізнавання. Процес включає:

– фільтрацію шумів (рис. 3.8), що очищує сигнал від небажаних звуків , такі як фоновий гул, клацання або шум навколишнього середовища. Це досягається за допомогою методу фільтрація високих та низьких частот, який видаляє частоти поза межами необхідного для аналізу діапазону, що дозволяє позбутися низькочастотних шумів, таких як гул кондиціонера, та високочастотних перешкод;

```
# Фільтрація шумів
1 usage
def noise_reduction(audio_signal, sample_rate):
    # Застосовуємо фільтр високих частот для зменшення шумів низької частоти
    cutoff_freq = 300
    b, a = signal.butter( N: 6, cutoff_freq / (sample_rate / 2), btype='highpass')
    filtered_signal = signal.lfilter(b, a, audio_signal)
    return filtered_signal
```

Рисунок 3.8 – Функція для фільтрації шумів

– нормалізацію амплітуди (рис. 3.9), яка забезпечує стабільний рівень гучності аудіосигналу, що важливо для якісного розпізнавання. Цей процес дозволяє уникнути випадків, коли слабкі звуки можуть бути втрачені, а занадто гучні – перенасичені;

```
# Нормалізація амплітуди
1 usage
def normalize_amplitude(audio_signal):
    # Нормалізуємо амплітуду сигналу до діапазону [-1, 1]
    max_amplitude = np.max(np.abs(audio_signal))
    if max_amplitude == 0:
        return audio_signal
    normalized_signal = audio_signal / max_amplitude
    return normalized_signal
```

Рисунок 3.9 – Функція для нормалізації амплітуди

– перетворення Фур’є (рис. 3.10) перетворює аудіо з часової форми у частотну для подальшого аналізу нейронною мережею. Це особливо важливо для нейронних мереж, які обробляють спектральні ознаки, такі як частоти та їх інтенсивності, а не часові коливання.

```python
# Перетворення Фур'є
1 usage
def fourier_transform(audio_signal, sample_rate):
    stft_result = librosa.stft(audio_signal, n_fft=2048, hop_length=512)
    magnitude_spectrogram = np.abs(stft_result)
    return magnitude_spectrogram
```

Рисунок 3.10 – Функція для перетворення Фур'є

Оскільки застосунок працює в режимі реального часу, попередня обробка виконується для кожного буфера в потоці, щоб забезпечити мінімальну затримку між закінченням мовлення та початком розпізнавання.

Після перетворення Фур'є будується спектрограма (рис. 3.11) на основі даних перетворення Фур'є і представляє зміну частотного вмісту сигналу у часі. Вона є двовимірним зображенням, де одна вісь відповідає часу, інша – частотам, а інтенсивність кольору або яскравість вказує на силу певної частоти у конкретний момент.

```python
1 usage
def create_spectrogram(audio: AudioData):
    # Створення мел-спектрограми
    spectrogram = librosa.feature.melspectrogram(y=audio, sr=audio.sample_rate, n_mels=128)
    # Перетворення спектрограми у шкалу децибелів
    log_spectrogram = librosa.power_to_db(spectrogram, ref=np.max)
    # Відображення спектрограми
    plt.figure(figsize=(10, 4))
    librosa.display.specshow(log_spectrogram, sr=audio.sample_rate, x_axis='time', y_axis='mel')
    plt.colorbar(format='%+2.0f dB')
    plt.title('Мел-спектрограма')
    plt.xlabel('Час (с)')
    plt.ylabel('Частота (мел)')
    plt.tight_layout()
    plt.show()
```

Рисунок 3.11 – Функція для створення спектрограми

Приклад спектрограми показано на рисунку 3.12. Це візуальне представлення використовується як вхід для нейронної мережі, яка спеціалізується на аналізі мовних сигналів. Після отримання спектрограми вона розбивається на кадри, які подаються послідовно до спеціально навченої нейронної мережі, зазвичай рекурентні нейронні мережі (RNN). Мережа

навчається розпізнавати характерні особливості мовлення, які відповідають окремим звукам або фонемам.

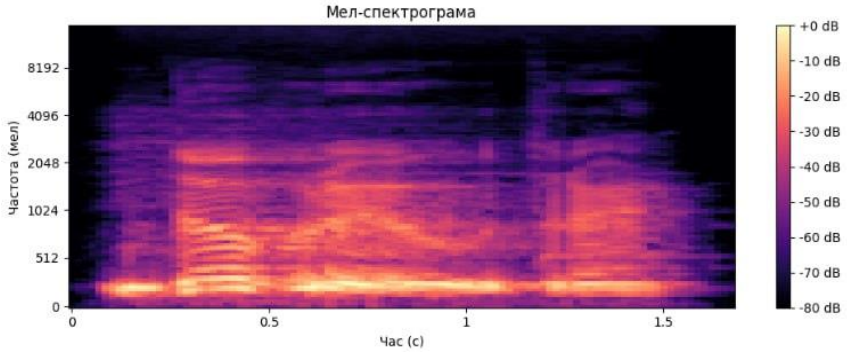

Рисунок 3.12 – Приклад спектрограми для аудіосигналу

RNN зберігає інформацію про попередні звуки, дозволяючи моделі враховувати, як одна фонема впливає на інші у контексті. У результаті система генерує послідовність ймовірностей для кожної фонеми, використовуючи контекстну інформацію. Архітектури нейронної мережі на основі RNN для розпізнавання фонем показано на рисунку 3.13.

```
import torch.nn as nn

1 usage
class SpeechRecognitionRNN(nn.Module):
    def init(self, input_size, hidden_size, output_size, num_layers=2):
        super(SpeechRecognitionRNN, self).init()
        self.rnn = nn.LSTM(input_size, hidden_size, num_layers, batch_first=True, bidirectional=True
        self.fc = nn.Linear(hidden_size * 2, output_size)

    def forward(self, x):
        rnn_out, _ = self.rnn(x)
        output = self.fc(rnn_out)
        return output
```

Рисунок 3.13 – Архітектура нейронної мережі на основі RNN

Під час розробки системи управління голосовими командами можна зіткнутися з викликом, як забезпечити точне розпізнавання та інтерпретацію команд навіть у випадках, коли є незначні похибки у розпізнаванні фонем чи шум у звуковому потоці. Щоб розв’язати це завдання, було створено два основних компоненти: мовна модель, яка враховує контекст слів, та алгоритм Левенштейна, який допомагає виправляти помилки у розпізнаванні.

Після того як мережа визначила послідовність фонем, завдання полягає у правильному складанні цих фонем у слова або фрази. Отримані фонеми порівнюються зі словником, у якому зберігаються всі можливі комбінації фонем і відповідні їм слова. Таким чином, система може реконструювати можливі слова з певною ймовірністю. Це дозволяє коригувати незначні помилки в розпізнаванні фонем, використовуючи знання про структуру мови. Система може використовувати мовні моделі для врахування ймовірності появи певних слів у поточному контексті.

Для того щоб система могла ефективно обробляти мовлення користувача, було створено мовну модель, яка покращує розпізнавання слів, враховуючи їх ймовірність у певному контексті. Мовна модель забезпечує, що система передбачає більш логічні та природні слова на основі попередніх слів у реченні. Був зібран певний набор тексту, що складається з численних прикладів природних фраз і команд, які користувач міг вимовити. На основі цього набору було розраховано частоти появи пар слів, що дозволило створити баграмну мовну модель. Баграми – це пари слів, де ймовірність другого слова залежить від першого. Щоб реалізувати це, було використано структуру defaultdict з Python, яка дозволяє зберігати частоти появи кожного слова після певного попереднього (рис. 3.14).

```python
from collections import defaultdict

# Ініціалізація словників для підрахунку частот баграм
bigram_counts = defaultdict(lambda: defaultdict(int))
total_counts = defaultdict(int)

# Приклад корпусу тексту
corpus = ["відкрий браузер", "яка погода", "запусти гру", "закрий програму"]

# Обчислення частот появи кожної баграми
for sentence in corpus:
    words = sentence.split()
    for i in range(len(words) - 1):
        bigram_counts[words[i]][words[i + 1]] += 1
        total_counts[words[i]] += 1

# Функція для обчислення ймовірності баграми
def bigram_probability(w1, w2):
    if total_counts[w1] == 0:
        return 0
    return bigram_counts[w1][w2] / total_counts[w1]
```

Рисунок 3.14 – Зберігання частоти появи кожного слова після попереднього

Мовна модель також дозволяє системі передбачати наступне слово. Це корисно для виправлення помилок, що виникають у процесі розпізнавання голосу. Ця мовна модель дозволяє системі не тільки розпізнавати слова, а й адаптуватися до помилок у розпізнаванні фонем, оскільки вона знає, які слова є більш ймовірними в даному контексті (рис. 3.15).

```
1 usage
def predict_next_word(word):
    if word not in bigram_counts:
        return None
    return max(bigram_counts[word], key=bigram_counts[word].get)

# Приклад використання мовної моделі
context_word = "відкрий"
predicted_word = predict_next_word(context_word)
print(f"Найбільш ймовірне наступне слово після '{context_word}': {predicted_word}")
```

Рисунок 3.15 – Розрахунок найбільш ймовірного наступного слова

Наступним важливим компонентом системи є алгоритм Левенштейна, який використовується для обробки незначних помилок у розпізнаних словах. Левенштейнова відстань вимірює, скільки операцій (вставок, видалень або замін) потрібно для перетворення одного слова в інше.

Коли система створила слово або фразу, виникає необхідність порівняти це слово з набором відомих команд. Система обчислює Левенштейнову відстань між розпізнаним словом та кожною відомою командою у списку (рис. 3.16). Щоб визначити, яка команда була розпізнана, система порівнює розпізнане слово з усіма відомими командами, використовуючи алгоритм Левенштейна. Якщо відстань між словами менша за певний поріг, команда вважається правильно розпізнаною. Наприклад, якщо користувач сказав «відкрий браузер», але через перешкоди мережа розпізнала це як «вткрий браузер», алгоритм Левенштейна дозволить правильно ідентифікувати команду «відкрий браузер».

Після вибору команди система негайно передає її в інтерфейс для виконання:

– виконання дії. Якщо команда є інструкцією на кшталт «відкрий браузер», програма відкриє веббраузер. Інші команди можуть запускати різні програми або виконувати інші дії на комп’ютері;

– подача інформації. Якщо команда пов’язана з отриманням інформації (наприклад, «яка погода»), програма може вивести інформацію на екран або озвучити її.

```
def levenshtein_distance(s1, s2):
    len_s1, len_s2 = len(s1), len(s2)
    dp = [[0] * (len_s2 + 1) for _ in range(len_s1 + 1)]

    # Ініціалізація базових випадків
    for i in range(len_s1 + 1):
        dp[i][0] = i
    for j in range(len_s2 + 1):
        dp[0][j] = j

    # Основний цикл для обчислення відстані
    for i in range(1, len_s1 + 1):
        for j in range(1, len_s2 + 1):
            cost = 0 if s1[i - 1] == s2[j - 1] else 1
            dp[i][j] = min(dp[i - 1][j] + 1,        # Видалення
                           dp[i][j - 1] + 1,        # Вставка
                           dp[i - 1][j - 1] + cost) # Заміна

    return dp[len_s1][len_s2]
```

Рисунок 3.16 – Обчислення відстані Левенштейна

Реалізована система розпізнавання голосових команд є складною і враховує безліч нюансів природної мови. Мовна модель допомагає передбачати найбільш ймовірні слова в певному контексті, що знижує кількість помилок, тоді як алгоритм Левенштейна виправляє незначні помилки у розпізнаванні, забезпечуючи високу точність розпізнавання навіть у складних умовах. Ці два компоненти працюють разом, щоб система могла надійно взаємодіяти з користувачем та швидко розпізнавати що саме було сказано.

Коли система завершила розпізнавання голосового вводу, вона переходить до фази порівняння розпізнаних слів або фраз з набором відомих команд, щоб визначити, яку дію необхідно виконати. У процесі виконання команд можуть виникати різні помилки, наприклад, якщо команда не

знайдена або зникло з’єднання з інтернетом. У таких випадках система обробляє винятки і надає користувачеві корисну зворотну інформацію. Після виконання команди система може надати зворотний зв’язок, щоб користувач знав, що команда виконана. Це може бути текстове повідомлення на екрані або голосова відповідь (рис. 3.17).

```
import pyttsx3

engine = pyttsx3.init()
engine.say("Команда виконана успішно")
engine.runAndWait()
```

Рисунок 3.17 – Повідомлення про успішність виконання команди

3.3.3 Реалізовані системні команди

Система розпізнавання голосових команд реалізована таким чином, що вона здатна виконувати різні дії на основі команд, які може надати користувач. Усі команди класифікуються за категоріями, і деякі з них можуть мати додаткові параметри. Нижче детально описано кожну команду та те, як вона може бути використана в застосунку:

– команда привітання. Застосунок програє аудіо привітання, коли користувач активує асистента (рис. 3.18);

– команда прощання. Застосунок програє аудіо прощання, коли користувач вимикає асистента (рис. 3.19);

– пошук відео. Виконує пошук відео на YouTube за вказаною назвою. Команда потребує додаткової обробки параметрів, що забезпечується за допомогою регулярних виразів. Це дозволяє витягувати параметри з команд користувача, щоб дізнатися, яке саме відео треба знайти (рис. 3.20);

```
def enable(self):
    self.isEnabled = True
    self.playSound('web/run.wav')
```

Рисунок 3.18 – Відтворення аудіо привітання

```
1 usage
def disable(self):
    self.isEnabled = False
    self.playSound('web/off.wav')
```

Рисунок 3.19 – Відтворення аудіо прощання

```
1 usage
def search_for_video_on_youtube(*args: tuple):
    os.system(f"start https://www.youtube.com/results?search_query={args[0]}")
```

Рисунок 3.20 – Пошук відео на YouTube

– пошук у Google. Система викнонує пошук у Google за бажаною темою користуавача;

– переклад. Система здатна перекласти сказаний текст на іншу мову, в залежності від того, на який просить користувач;

– прогноз погоди. Користувач може попросити систему дізнатися, яка зараз температура на вулиці (рис. 3.21);

– команди управління звуком. Система здатна регулювати гучність звука. Користувач може попросити систему вимкнути або увімкнути звук (рис. 3.22), а також зробити тихіше чи голосніше на певний відсоток (рис. 3.23);

– розгортання на весь екран. Система робить вікно застосунка розгорнутим на весь екран.

```
1 usage
def get_weather_forecast(city):
    url = f'https://api.openweathermap.org/data/2.5/weather?q={city}&units=metric&lang=ua&appid={api_key}

    response = requests.get(url)

    if response.status_code == 200:
        data = response.json()
        temp = round(data['main']['temp'])
        desc = data['weather'][0]['description']
        play_voice_assistant_speech(f'Температура {temp} градусів та {desc}')
    else:
        print('Error fetching weather data')
```

Рисунок 3.21 – Пошук інформації про погоду

```
1 usage
def decrease_volume(self, decrease_on_value):
    current_volume = self.volume.GetMasterVolumeLevelScalar()

    new_volume = max(0, current_volume - (int(decrease_on_value) / 100))
    self.volume.SetMasterVolumeLevelScalar(new_volume, None)

    print(f"Гучність зменшена на {decrease_on_value}%. Поточна гучність: {new_volume * 100}%")
```

Рисунок 3.22 – Регулювання гучності звуку

```
1 usage
def mute_volume(self):
    self.previous_volume = self.volume.GetMasterVolumeLevelScalar()
    self.volume.SetMasterVolumeLevelScalar(0.0, None)
```

Рисунок 3.23 – Вимкнення звуку

3.4 Дослідження практичних результатів

Для оцінки якості роботи застосунку було проведено всебічне тестування з урахуванням таких критеріїв, як дистанція до мікрофона, рівень навколишнього шуму та швидкість мовлення користувачів. Крім того, щоб отримати більш достовірні результати, система була протестована з використанням голосів декількох людей різного віку, статі та з різними тембрами голосу. Зокрема, у тестуванні брали участь три жінки у віці від 20 до 56 років і три чоловіки у віці від 14 до 57 року. Це дозволило перевірити, наскільки ефективно система розпізнає мову в різних умовах і з урахуванням

індивідуальних особливостей голосу. Результати показано у таблицях 3.1 – 3.3.

Таблиця 3.1 – Відсоток коректно розпізнаних слів за критеріями відстані та шуму

Відстань \ Шум	**Тихо**	**Шумно**
Біля мікрофона	100%	85%
1 м	100%	80%
3 м	80%	60%

Таблиця 3.2 – Відсоток коректно розпізнаних слів за критеріями швидкості та шуму

Швидкість \ Шум	**Тихо**	**Шумно**
Повільно	100%	85%
Нормально	100%	85%
Швидко	90%	75%

Таблиця 3.3 – Відсоток коректно розпізнаних слів за критеріями швидкості та відстані

Швидкість \ Відстань	**Біля мікрофона**	**1 м**	**3 м**
Повільно	100%	100%	100%
Нормально	100%	100%	100%
Швидко	95%	95%	85%

Загалом, результати тестування виявилися досить задовільними. Система продемонструвала здатність точно розпізнавати голоси людей з різними характеристиками, зберігаючи високу ефективність навіть при зміні тембру, статі або віку користувачів. За умов чіткої і роздільної вимови слів система досягає стовідсоткової точності розпізнавання. Однак навіть у

випадках швидкої, злитної мови, система змогла показати відносно високу ефективність, досягаючи коефіцієнта точності WER у вісімдесят відсодків.

Під час тестів у складніших умовах, таких як значний рівень фонових шумів або одночасна розмова кількох людей, точність розпізнавання може знижуватися до шістдесяти відскодків, що може означати два нерозпізнаних слова на речення. Це є досить типовою проблемою для більшості сучасних систем розпізнавання мови. Проте слід зазначити, що застосунок має потенціал для подальшого покращення, наприклад, шляхом впровадження більш досконалих алгоритмів фільтрації шумів і використання більш складних моделей нейронних мереж, що здатні краще адаптуватися до різних умов.

Крім того, були проведені експерименти з вимірювання часу затримки між закінченням вимови і розпізнаванням команди. Застосунок показав досить швидкий відгук, що забезпечує природну взаємодію з користувачем. Час відгуку залежить від складності фрази і рівня шуму, але в середньому становить не більше двох секунд, що є прийнятним для більшості завдань.

Висновки з тестування показують, що система є цілком придатною для використання в реальних умовах, проте залишаються області для вдосконалення. Надалі можливе застосування адаптивних мовних моделей і підвищення стійкості до шуму, що зробить систему ще більш надійною та універсальною.

3.5 Перспективи розвитку системи

Подальший розвиток відкриває величезні можливості для покращення взаємодії з користувачем і розширення функціоналу. Майбутні покращення стосуються як удосконалення існуючих алгоритмів розпізнавання, так і впровадження нових функцій, які зроблять систему ще більш зручною та

корисною для різноманітних користувачів, зокрема для тих, хто має особливі потреби.

Один із ключових напрямів розвитку системи – це покращення алгоритмів розпізнавання голосу. Сьогодні система вже добре справляється з розпізнаванням команд, проте існує потенціал для подальшого підвищення її точності та швидкості. Майбутні оновлення можуть включати:

– впровадження адаптивних мовних моделей. Система може навчатися на основі індивідуальних особливостей користувача, таких як акцент, швидкість і манера мовлення. Це дозволить зробити розпізнавання ще більш персоналізованим;

– застосування сучасних методів машинного навчання. Використання рекурентних нейронних мереж (RNN), трансформерів і моделей глибокого навчання допоможе покращити точність розпізнавання складних мовних конструкцій і скоротити кількість помилок при швидкому мовленні чи в умовах фонового шуму;

– інтеграція системи з базами даних фонетичних варіантів. Це дозволить системі коригувати розпізнавання, беручи до уваги особливості вимови різних мов і діалектів.

Інший напрямок розвитку системи – підтримка багатоканального розпізнавання. Це означає, що застосунок зможе одночасно обробляти голосові команди від декількох користувачів. Таке нововведення буде особливо корисним для корпоративного середовища, де одразу кілька людей можуть взаємодіяти з одним пристроєм.

Система може стати неймовірно корисною для людей з обмеженими можливостями, зокрема для тих, хто має проблеми зі слухом або мовленням. Плануються такі покращення:

– перетворення звуку на текст у режимі реального часу. Це допоможе глухим і слабочуючим користувачам сприймати аудіоінформацію з відео. Наприклад, при перегляді відео або прослуховуванні подкасту система

автоматично генеруватиме текстовий супровід, що дозволить краще зрозуміти вміст;

– виведення субтитрів із вбудованими контекстними підказками. Окрім перетворення звуку в текст, система зможе додавати підказки, наприклад, позначення емоційної інтонації (сміх, плач, здивування) або опис важливих звукових ефектів (грім, звук сигналу, тощо).

Ще однією перспективною можливістю є додавання функцій розпізнавання жестів. Завдяки сучасним алгоритмам комп'ютерного зору система зможе зчитувати жести користувача, дозволяючи виконувати команди не тільки голосом, але й рухами рук. Це відкриває шлях до інтеграції з пристроями типу розумних телевізорів, де користувач може перемикати канали, змінювати гучність або взаємодіяти з меню за допомогою жестів.

Система може розширюватися шляхом інтеграції з іншими розумними пристроями, такими як розумні колонки, термостати або автомобільні асистенти. Наприклад, користувачі зможуть використовувати голосові команди для керування розумним будинком: увімкнути світло, відрегулювати температуру в кімнаті або запустити робота-пилососа. Такі функції зроблять систему важливою частиною екосистеми розумного дому.

Щоб підвищити стійкість системи до зовнішніх шумів, буде впроваджено передові методи обробки аудіосигналів, такі як адаптивні фільтри шуму або методи використання просторового звуку для виділення голосу користувача. Завдяки цьому система зможе ефективно працювати навіть у шумних середовищах, таких як офіси, торгові центри чи транспорт.

Подальший розвиток системи управління голосовими командами відкриває великі перспективи для покращення зручності та ефективності її використання. Завдяки впровадженню нових технологій і функцій, таких як адаптивне навчання, підтримка жестів і оптимізація для людей з особливими потребами, ця система зможе забезпечити кращу якість взаємодії та стати важливим помічником у повсякденному житті.

ВИСНОВКИ

У рамках кваліфікаційної роботи була розроблена і реалізована система для розпізнавання голосових команд за допомогою нейронної мережі, що включає передові методи обробки звуку, аналізу мовлення та виконання команд. Результатом проведених досліджень і впровадження алгоритмів стала система, здатна розпізнавати та обробляти голосові команди в реальному часі, враховуючи різноманітні фактори, які можуть впливати на якість розпізнавання, зокрема фонові шуми, різницю в тембрах голосу та швидкість мовлення.

Науковою новизною даного дослідження є запропонований ефективний метод розпізнавання голосових команд з використанням MFCC для виділення ознак звуку, який на відміну від класичних методів розпізнавання голосових команд відрізняється можливістю вилучення ключових елементів з аудіосигналів, що подаються на обробку та дозволяє формувати вектори ознак, які потім використовуються для класифікації команд.

Особлива увага була приділена попередній обробці аудіосигналів, включаючи фільтрацію шумів і нормалізацію амплітуди, щоб забезпечити найвищу якість вхідних даних для системи. Було також розроблено ефективний механізм перетворення звуку у частотну форму за допомогою алгоритмів швидкого перетворення Фур’є, що дозволило виділяти ключові ознаки звуку для подальшого аналізу за допомогою нейронних мереж.

Реалізація системи включає продуману систему обробки фонем та аналізу слів, яка використовує методи корекції помилок, зокрема обчислення Левенштейнової відстані для підвищення точності визначення команд.

Було також протестовано різні аспекти системи, такі як точність розпізнавання у різних акустичних умовах, стійкість до шумів і швидкість виконання команд. Результати тестування показали, що застосунок демонструє високу ефективність, навіть при швидкій вимові та в присутності зовнішніх шумів.

Окрім базової функціональності, були передбачені і перспективи розвитку системи: інтеграція функцій для людей з особливими потребами, покращення адаптивності мовних моделей, додавання підтримки жестів та інтеграція з іншими смарт-пристроями. Дана система має величезний потенціал для розвитку і може бути використана у різних галузях.

Результати дослідження апробовано у вигляді тез доповіді під час Міжнародного молодіжного форуму «РАДІОЕЛЕКТРОНІКА І МОЛОДЬ У XXI СТОЛІТТІ» [42].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Hu, Z., Bodyanskiy, Y. V., Tyshchenko, O. K., & Shafronenko, A. (2019, July). Fuzzy clustering of incomplete data by means of similarity measures. In *2019 IEEE 2nd Ukraine Conference on Electrical and Computer Engineering (UKRCON)* (pp. 957-960). IEEE.

2. Bodyanskiy, Y., Shafronenko, A., & Pliss, I. (2022). Clusterization of vector and matrix data arrays using the combined evolutionary method of fish schools. *System research and information technologies*, (4), 79-87.

3. Shafronenko, A. Y., Kasatkina, N. V., Bodyanskiy, Y. V., & Shafronenko, Y. O. (2023). CREDIBILISTIC ROBUST ONLINE FUZZY CLUSTERING IN DATA STREAM MINING TASKS. *Radio Electronics, Computer Science, Control*, (3), 97-97.

4. Шафроненко, А. Ю., Бодянський, Є. В., & Руденко, Д. О. (2023). Модифікований рекурентний метод достовірної нечіткої кластеризації з використанням оптимізаційної процедури на основі косяків риб. *Системи обробки інформації*, (1 (172)), 92-96.

5. Шафроненко, А., Бодянський, Є., & Плісс, І. (2022). Нечіткі методи інтелектуального аналізу даних.

6. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems. *Advances in neural information processing systems*, *30*(2017).

7. Graves, A., Mohamed, A.-R., & Hinton, G. (2013). Speech recognition with deep recurrent neural networks. *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*.

8. Stevens, K. N. (2000). *Acoustic phonetics* (Vol. 30).

9. Hannun, A. (2014). Deep Speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*.

10. Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E.,

Case, C., ... & Zhu, Z. (2016, June). Deep speech 2: End-to-end speech recognition in english and mandarin. In *International conference on machine learning* (pp. 173-182). PMLR.

11. Chorowski, J. K., Bahdanau, D., Serdyuk, D., Cho, K., & Bengio, Y. (2015). Attention-based models for speech recognition. *Advances in neural information processing systems*, *28*.

12. Graves, A., & Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural networks*, *18*(5-6), 602-610.

13. Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A. R., Jaitly, N., ... & Kingsbury, B. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, *29*(6), 82-97.

14. Prabhavalkar, R., Rao, K., Sainath, T. N., Li, B., Johnson, L., & Jaitly, N. (2017, August). A Comparison of sequence-to-sequence models for speech recognition. In *Interspeech* (pp. 939-943).

15. Xiong, W., Wu, L., Alleva, F., Droppo, J., Huang, X., & Stolcke, A. (2018, April). The Microsoft 2017 conversational speech recognition system. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)* (pp. 5934-5938). IEEE.

16. Bodyanskiy, Y. V., Pliss, I. P., & Shafronenko, A. Y. (2022). Кластеризація масивів даних на основі комбінованої оптимізації функцій щільності розподілу та еволюційного методу котячих зграй. *Radio Electronics, Computer Science, Control*, (4), 61-61.

17. Тимощук, О., & Янчук, П. (2024). Аналіз роботи глибоких нейронних мереж методом швидкого перетворення Фур'є з використанням програмного забезпечення с. *Прикладні проблеми комп'ютерних наук, безпеки та математики*, (3), 23-31.

18. Wijaya, N. N., & Muslikh, A. R. (2024). Music-genre classification using Bidirectional long short-term memory and mel-frequency cepstral

coefficients. *Journal of Computing Theories and Applications*, *1*(3), 243-256.

19. Mashika, M., & van der Haar, D. (2023, July). Mel Frequency Cepstral Coefficients and Support Vector Machines for Cough Detection. In *International Conference on Human-Computer Interaction* (pp. 250-259). Cham: Springer Nature Switzerland.

20. Bringmann, K., Fischer, N., van der Hoog, I., Kipouridis, E., Kociumaka, T., & Rotenberg, E. (2024). Dynamic Dynamic Time Warping. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)* (pp. 208-242). Society for Industrial and Applied Mathematics.

21. Jiang, S., & Chen, Z. (2023). Application of dynamic time warping optimization algorithm in speech recognition of machine translation. *Heliyon*, *9*(11).

22. Manideep, N., & Mohana, J. (2023, November). Hidden Markov model to recognise the voice and increase the recognition rate compared with dynamic time warping method. In *AIP Conference Proceedings* (Vol. 2822, No. 1). AIP Publishing.

23. Isaac, S., Haruna, K., Ahmad, M. A., & Mustapha, R. (2023). Deep Reinforcement Learning with Hidden Markov Model for Speech Recognition. *Journal of Technology and Innovation*, 01-05.

24. Junling, Y. (2024). Online learning system for English speech automatic recognition based on hidden Markov model algorithm and conditional random field algorithm. *Entertainment Computing*, 100729.

25. Mienye, I. D., Swart, T. G., & Obaido, G. (2024). Recurrent neural networks: A comprehensive review of architectures, variants, and applications. *Information*, *15*(9), 517.

26. Shafronenko, A., Bodyanskiy, Y. V., & Pliss, I. (2023). Credibilistic Fuzzy Clustering Method Based on Evolutionary Approach of Crazy Wolfs in Online Mode. In *CMIS* (pp. 141-150).

27. Ackerson, J. M., Dave, R., & Seliya, N. (2021). Applications of recurrent neural network for biometric authentication & anomaly

detection. *Information*, *12*(7), 272.

28. Oruh, J., Viriri, S., & Adegun, A. (2022). Long short-term memory recurrent neural network for automatic speech recognition. *IEEE Access*, *10*, 30069-30079.

29. Довга короткочасна пам’ять. URL: https://uk.wikipedia.org/wiki/Довга_короткочасна_память (дата звернення 31.10.2024).

30. Bodyanskiy, Y. V., Pliss, I. P., & Shafronenko, A. Y. (2022). ШВИДКА НЕЧІТКА ПРАВДОПОДІБНА КЛАСТЕРИЗАЦІЯ НА ОСНОВІ АНАЛІЗУ ПІКІВ ЩІЛЬНОСТІ РОЗПОДІЛУ ДАНИХ. *Radio Electronics, Computer Science, Control*, (1), 76-76.

31. Minaylo, A. Y., & Turchina, V. A. (2015). Використання відстані Левенштейна для аналізу подібності даних. *Питання прикладної математики і математичного моделювання*.

32. Bodyanskiy, Y. V., Shafronenko, A. Y., & Klymova, I. N. (2021). Online fuzzy clustering of incomplete data using credibilistic approach and similarity measure of special type. *Radio Electronics, Computer Science, Control*, (1), 97-104.

33. Герасимова, С. С., & Шафроненко, А. Ю. (2024, April). РОЗРОБКА ЗАСТОСУНКУ ДЛЯ ОНЛАЙН РОЗПІЗНАВАННЯ ОБЛИЧ У ВІДЕОПОТОЦІ. In *The 5 th International scientific and practical conference “Science and society: modern trends in a changing world”(April 15-17, 2024) MDPC Publishing, Vienna, Austria. 2024. 492 p.* (p. 151).

34. Що таке Python та де він використовується. URL: https://dan-it.com.ua/blog/python-chto-jeto-za-jazyk-programmirovanija-i-gde-ego-ispolzujut/ (дата звернення 31.10.2024).

35. Sahoo, K., Samal, A. K., Pramanik, J., & Pani, S. K. (2019). Exploratory data analysis using Python. International Journal of Innovative Technology and Exploring Engineering (IJITEE), 8(12), 2019.

36. Heydt, M. (2017). Learning pandas. Packt Publishing Ltd.

37. Уроки Python. URL: https://itproger.com/ua/course/python (дата

звернення 31.10.2024).

38. Numpy Introduction. URL: https://www.w3schools.com/python/numpy/numpy_intro.asp (дата звернення 26.04.2023).

39. Pandas Tutoria. URL: https://www.w3schools.com/python/pandas/pandas_dataframes.asp (дата звернення 31.10.2024).

40. Introduction to Plotting with Matplotlib in Python. URL: https://www.datacamp.com/tutorial/matplotlib-tutorial-python (дата звернення 31.10.2024).

41. Scikit Learn Tutorial. URL: https://www.tutorialspoint.com/scikit_learn/index.htm (дата звернення 31.10.2024).

42. Касумов, А. І. (2024). Використання відстані Левенштейна при розробці голосового асистента. Радіоелектроніка та молодь у XXI столітті. Т. 7: Конференція "Комп’ютерний зір, системний аналіз та математичне моделювання": матеріали 28-го Міжнар. молодіж. форуму, 16–18 квіт. 2024 р., с. 54-56.